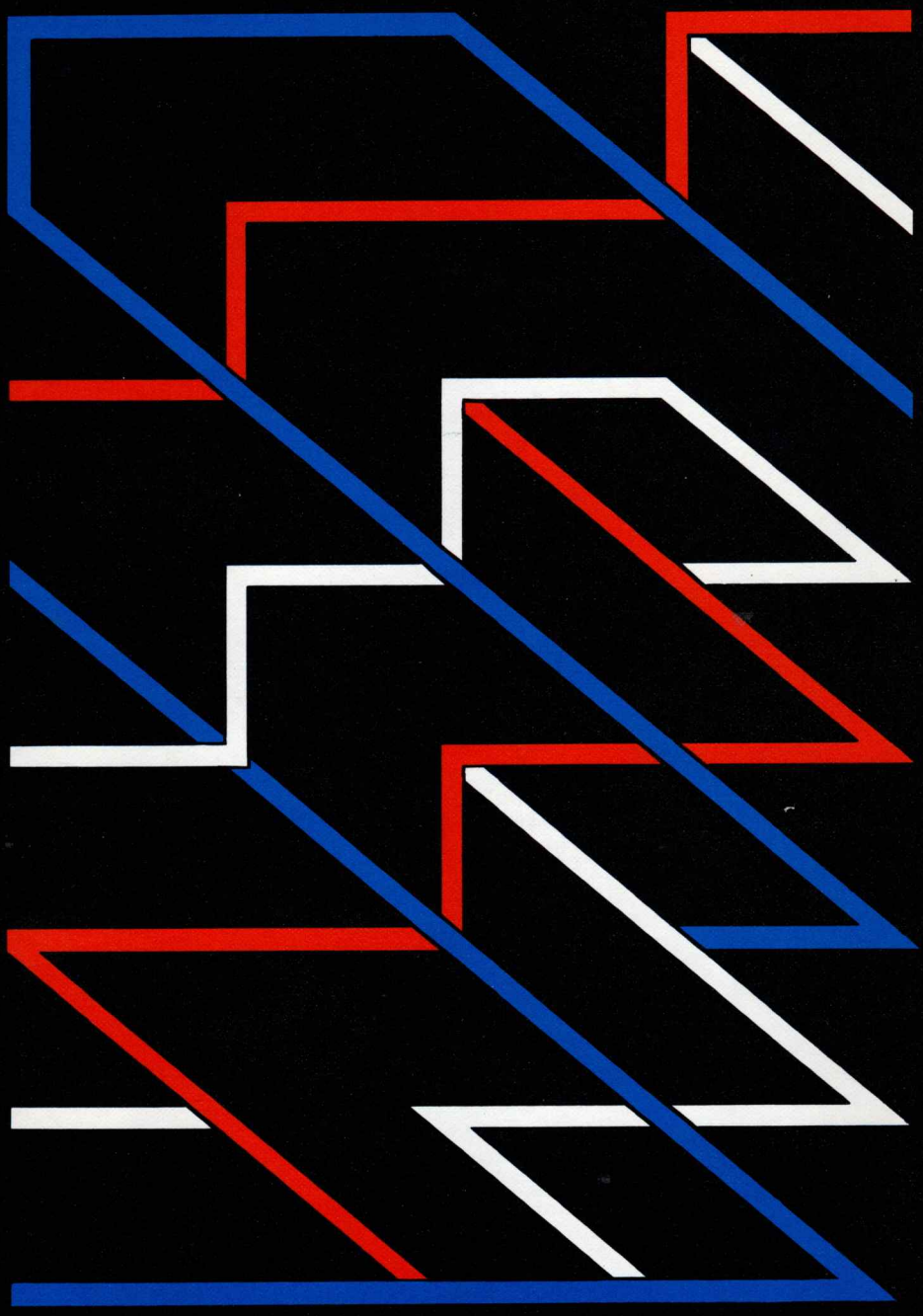
edited by

DAVID SANKOFF AND JOSEPH B. KRUSKAL

TIME WARPS, STRING EDITS, AND MACROMOLECULES: THE THEORY AND PRACTICE OF SEQUENCE COMPARISON



CHAPTER FIVE

USE OF DYNAMIC PROGRAMMING IN A SYLLABLE-BASED CONTINUOUS SPEECH RECOGNITION SYSTEM

Melvyn J. Hunt, Matthew Lennig, and Paul Mermelstein

1. INTRODUCTION

Speech is often said to be the fastest and most natural way for a person to communicate with a machine. Devices that can recognize isolated words have been available for some time now and are finding wide application (Bridle and Brown, 1974). However, the input required is often not a single word but a phrase or sentence. In that case, much of the speed and naturalness of voice input is lost if the words have to be spoken in isolation with periods of silence between them.

This chapter is concerned with the recognition of natural, continuously spoken sentences. The task is much more difficult than the isolated-word problem, and the few systems currently available are comparatively simple and not yet in widespread use. The problem becomes tractable only when the syntax and vocabulary of the phrases that the machine is expected to recognize are restricted. This may not be a serious handicap in practice: Most applications being considered are concerned with specific tasks and do not require free-ranging discussion. It is desirable, however, that the phrases that can be recognized should comprise a clean and natural subset of the user's language in order that he or she can easily understand which phrases are acceptable to the machine.

We are taking a syllable-based approach, where dynamic-programming methods are applied both to the recognition of syllables and to the evaluation of competing sentence hypotheses viewed as sequences of syllables. In this chapter we first point out some of the ways in which speech differs from many other sequence-matching problems. We look briefly at how some workers have addressed the problem and then try to explain why we feel that a syllable-based approach may be worth pursuing. In discussing our own system, we first describe the organization of the syntax and sentence-evaluation routines and then provide details of the time-warping algorithm used for syllable recognition. Results of experiments comparing variants of the syllable-recognition algorithm are presented.

2. SPEECH RECOGNITION AS A STRING-MATCHING PROBLEM

From an acoustical point of view, speech, like other sounds, consists of a sequence of small, rapidly varying perturbations in air pressure. These perturbations can be recorded as a waveform on an analogue medium such as a gramophone record.

The most direct way of making a digital recording is to sample the waveform at regular intervals and assign the pressure variation at each instant to the closest of a number of discrete levels. In order for good intelligibility and telephone-like quality to be preserved, the sampling rate needs to be at least 8000 times a second, and at least 128 levels are needed to represent the pressure changes. This is an enormously high information rate: The amount of speech that can be recorded in this way is severely limited and any digital processing applied to the waveform is computationally expensive. Moreover, because of the insensitivity of the ear to moderate changes in phase, two waveforms can look quite different and yet sound identical. The digitized waveform is therefore not a suitable representation of speech for recognition purposes.

Various methods have been developed that exploit the redundancy in the speech waveform, to reduce by at least an order of magnitude the amount of storage needed to reproduce speech of a given quality. Instead of a single-dimensional function sampled very frequently, the speech is represented by a vector of up to twenty components updated around a hundred times a second. For most of the time, the values of the vector components change slowly between updates, and they can be pictured as tracing out piecewise continuous curves in time. It is primarily between sequences of these vectors that string matching takes place.

String-matching for continuous speech recognition differs in several ways from many of the other string-matching problems discussed in this volume. First, speech is best described as a multilevel sequence: A sentence can be considered to be a sequence of words, and the words a sequence of speech sounds (loosely, "phonemes"). One might prefer to use other elements, perhaps

syllables rather than words, or basic features of speech sounds such as the presence of voicing or frication rather than phonemes; but whatever elements are chosen, at least two levels are necessary if a reasonably compact description of allowed speech sequences is to be obtained. The recognition process itself may, as in our case, carry out string matching at both levels.

The underlying elements of a speech sequence do not have completely consistent acoustic realizations: No two utterances of the same word in the same context by the same speaker will ever be absolutely identical. Consequently, it is not possible to say that an element is present or that another element has been substituted for it; one tries instead to have a measure of the *probability* of a particular element being present, usually in terms of its closeness to some stored model form of the element. The model forms themselves have to be constructed in some way from previous utterances. There are no ideal forms in the sense that, for instance, the ideal form of some manmade signal might be known.

Finally, the elements making up the speech sequence have very variable durations, and in general they are difficult to isolate. There are no clear acoustic cues to word boundaries in speech corresponding to spaces between words in written language. At the phoneme level the situation is worse; although speech is widely transcribed by a linear sequence of phonemes, one could argue that trying to divide a spoken word into discrete phonemes by analogy with letters in a written word is nonsensical. The acoustic features that characterize a phoneme overlap in time and interact with those of adjacent phonemes. There is, for example, no way of cutting a piece of magnetic tape containing a recording of the word "do" so that a recognizable "d" sound can be heard in isolation from the following vowel; and the major acoustic difference between "once" and "ones" lies in the length of the "n" sound rather than in the form of the fricative sound that follows it. Even speech sounds that can be cleanly separated from their neighbours are usually influenced to some extent by the sounds around them.

3. APPROACHES TO THE PROBLEM

In tackling the problem of continuous speech recognition, a wide variety of approaches have been adopted, many of them involving dynamic programming. This section does not purport to be a survey of the field of continuous speech recognition (see Reddy (1976) for example), but is simply intended to set our own work in some perspective.

One of the major divisions in approaches to speech recognition is between systems that try to segment the speech into phoneme-like units and those that do not. The nonsegmenting systems usually describe the speech by a series of frames that are "isochronous," i.e., equally spaced in time (typically 10 milliseconds apart). There are generally several of these frames per phoneme and the changes between frames are in general fairly gradual. Thus, the

nonsegmenting systems are less economical in their descriptions of words, and they are less well suited to pronunciation variations that involve the substitution, deletion, or insertion of whole phonemes. Systems that do attempt segmentation into speech sounds group varying numbers of isochronous frames together to form units of unequal duration intended to represent phonemes or phoneme subunits (such as stop gaps). The segmenting systems are faced with the difficulties in isolating phonemes that we mentioned in the previous section.

In the early 1970s the U.S. Department of Defense Advanced Research Projects Agency (ARPA) initiated a five-year program for the development of speech understanding systems. All of the major ARPA systems, namely HWIM, Hearsay II, Harpy, and the SDC system (see the review by Klatt, 1977) attempt segmentation into phonemes, though HWIM carries out a whole-word template-matching process on the isochronous frames in order to verify words hypothesized from the phoneme string. Complex logical rules have been proposed to deal with context problems in phoneme recognition (De Mori, 1979), and inevitable segment-identification ambiguities are often handled by holding an ordered list of likely phoneme candidates for subsequent higher-level analysis. Harpy and HWIM are examples of two different approaches to segmentation, in that Harpy first segments and then tries to recognize the phonemes (with the result that one phoneme often corresponds to several segments), while HWIM makes its segment-boundary decisions dependent on its phoneme hypotheses, holding several hypotheses with different boundaries in a "phoneme lattice." With its freedom to choose the optimum boundaries for each phoneme-sequence hypothesis, the HWIM approach presumably leads to better phoneme identification, but the Harpy method is more suited to the string-matching algorithms described in this volume.

Harpy demonstrates a further advantage of systems that classify sounds into phonemes: speaker adaptation can be achieved by collecting enough material from the new speaker to modify the reference phonemes. A system that does not classify into phonemes must either collect much more material and adapt at the higher (word or syllable) level, or else it must attempt some general acoustic transformation.

Some workers have carried out segmentation into purely acoustic units, such as regions of constant spectrum, without classification of the segments into phonemes. The primary advantage of this over nonsegmenting methods is in computational efficiency. Bridle and Sedgwick (1978) have described a dynamic-programming algorithm for the optimum division of a speech sequence into a given number of acoustic segments.

Equally, some systems that do not segment speech into phonemes nevertheless attempt to assign each individual frame to a phoneme-like class. This formulation of the speech-recognition task is the one that casts it in a form most similar to other sequence-matching problems such as macromolecule comparison. Substitution errors are inevitably very common. Kashyap (1979) recently described a system of this kind in which substitution costs were related to the estimated confusability of the phoneme classes.

It is, however, much more common for nonsegmenting systems to describe each frame by a set of continuous parameters. Work at IBM (Bahl *et al.*, 1979) is particularly interesting here because they have evaluated examples of the complete range of possibilities. Their first system contained a module (MAP) which segmented the speech and assigned a single phoneme label to each segment. Their next two systems (CSAP–1 and CSAP–2) divided the speech into centisecond (that is, 10 millisecond) frames and assigned a phoneme-like label to each frame, CSAP–2 differing from CSAP–1 in having more labels. Their most recently described system, WBAP, is also nonsegmenting, but it describes each frame by continuous parameters. There has been a uniform improvement in performance throughout this sequence of systems. Their results suggest that it is preferable to preserve the continuous spectral information for higher-level string matching rather than make early hard decisions on phoneme identities and locations, with a consequent loss of information.

Most nonsegmenting methods store acoustic models, "templates," representing whole words. A template consists of a sequence of frames containing information on the short-term power spectrum of the word as a function of time. It is aligned in time to a portion of the unknown speech and the spectral information is compared. Since different utterances of the same word can vary widely in their total duration and in the relative durations of parts of the word, the alignment process generally allows parts of the template and unknown speech to be stretched or compressed with respect to each other. Dynamic-programming template-matching algorithms are described in Sect. 5.

The great advantage of the word-matching approach is that since most phonetic context effects occur within words, whole-word templates automatically contain most of the required context information. The great problem is the difficulty of segmenting into words: Since there are no general acoustic cues to word boundaries, it is difficult to know which portion of speech should be matched against a template. Usually, the templates are matched consecutively across the speech, so that when a template has fitted part of the speech well, it is assumed that the next word starts at the point in the speech where the match to the previous word ended (Bridle and Brown, 1979). Thus the decision as to where to put the word boundary depends only on the match to the end of the previous word and not on the match to the beginning of the next word, although in some formulations (Levinson and Rosenberg, 1979), "dithering" of the word boundary is allowed in matching the next word.

4. THE SYLLABLE-BASED SYSTEM

In the following sections we describe our own syllable-based approach to the problem of continuous speech recognition.

4.1 Motivation for Using Syllables

The energy profile of a speech signal shows a modulation due to the syllable structure. This may make syllables the only elements of speech that can be consistently isolated independently of recognition, and our system in fact determines the syllable boundaries before any recognition is attempted.

The syllables to be recognized are matched against stored syllable templates. Syllable templates share the advantages of words in being substantially free of phonetic context effects, but in addition to being more compact for storage they are free of many of the disadvantages of word templates. It is possible to know, before recognition begins, how many templates need to be matched to the speech and between which points the matching must take place. Most importantly, alternative sentence hypotheses have the same syllable boundaries; thus they can be easily compared using dynamic-programming string-matching methods.

We do not pretend that our kind of syllable-based approach is the ultimate answer to the problem of continuous speech recognition. Future large-scale systems will almost certainly incorporate phonemic knowledge. However, the advantages outlined above make a syllable-based system attractive for the many practical applications where only a limited field of discourse is required.

The syllable method depends critically, of course, on there being an acceptable division into syllables, and this may set the ultimate limit to the performance of our method. Happily, sentences with a syllabification error (at present, roughly 10% of all sentences) almost always result in the recognition phase rejecting the sentence rather than misunderstanding it. For most applications a rejection and request for the sentence to be repeated will be less serious than a misrecognition.

The details of how a sentence is split into syllables (Mermelstein, 1975) are not pertinent to this volume, and we will describe only the most general features of the method here. It works primarily on the energy in the speech signal. Sufficiently deep local minima in the energy are candidates for syllable boundaries. The potential syllables produced by these boundaries then have to satisfy criteria concerning their length and loudness and degree of voicing. Note that the "syllables" produced in this way to not have to correspond to the conventional idea of what the syllables should be: there is no objection to "twenty" being taken to be a single syllable provided that this happens sufficiently often for the system to be aware of the possibility.

Moreover, absolute consistency is not required. For example, the system can store both single-syllable and two-syllable reference versions of "twenty" and so recognize either form. Difficulties can, however, arise when a "syllable" spans a syntactic boundary, such as when "twenty-eight" is split into "twen" and "ty-eight," and when syllable boundaries are placed in completely unexpected places.

4.2 System Overview

The sequence of operations involved in recognizing a sentence is shown in Fig. 1. The sentence is first low-pass filtered at 3.7 kHz, sampled at 8 kHz, and digitized.

The preliminary processing seeks to represent the speech in a compact manner by eliminating redundancy and features that provide information about the speaker or his mood rather than about the words he is using. We believe that the representation chosen should also reflect some of the properties of the human ear. As we mentioned in Sec. 2, the ear is relatively insensitive to phase, so it is appropriate to compute the short-term power spectrum and discard the phase information. The perceived loudness of a sound is roughly proportional to the logarithm of the intensity, so we work with the log power spectrum. Because the capacity of the ear to resolve two close frequencies decreases at frequencies above 1 kHz, we make use of a standard scale of frequencies known as the "mel" scale, in which equal differences in mel values correspond to equal perceived pitch differences. The varying resolving power of the ear is reflected in our system by dividing the spectrum into twenty channels of equal width on the mel scale.

Because speech spectra are fairly smooth, the energies in adjacent channels are highly correlated. Such correlation can be substantially removed by expanding the channel energies in a Fourier series. Since the spectrum is considered to be symmetric about 0 Hz, the only nonzero terms in the expansion are the cosine terms. The first few of these coefficients, which are known as "mel-scale cepstrum coefficients," form a very compact description of the spectrum. We compute the coefficients every 6.4 milliseconds. At the same time we estimate the perceptual loudness, which we use in the syllabifier. The syllabifier separates speech from silence and divides the speech regions into syllables. The rest of the system is concerned with trying to recognize the sequence of syllables.

In speech-recognition jargon, our system is "top-down," "left-to-right." That is to say, the syllables are considered in the order in which they occur in the sentence. Each syllable to be recognized is compared by the syllable comparator only with those templates that would be consistent with the syntax and the system's knowledge of the sentence so far. The output from the comparator is a measure of similarity between the template and the unknown and is referred to as a (squared) distance, although it would not always satisfy the requirements of a true distance measure.

To compare a syllable with a template, string-matching is used. During this process, as we shall see in Sec. 5, each frame making up the unknown syllable may be compared to a frame in a reference template, or it may be skipped over, or it may be involved in a comparison with two or more consecutive reference frames. The comparison rests entirely on a similarity measure, and does not involve any classification.

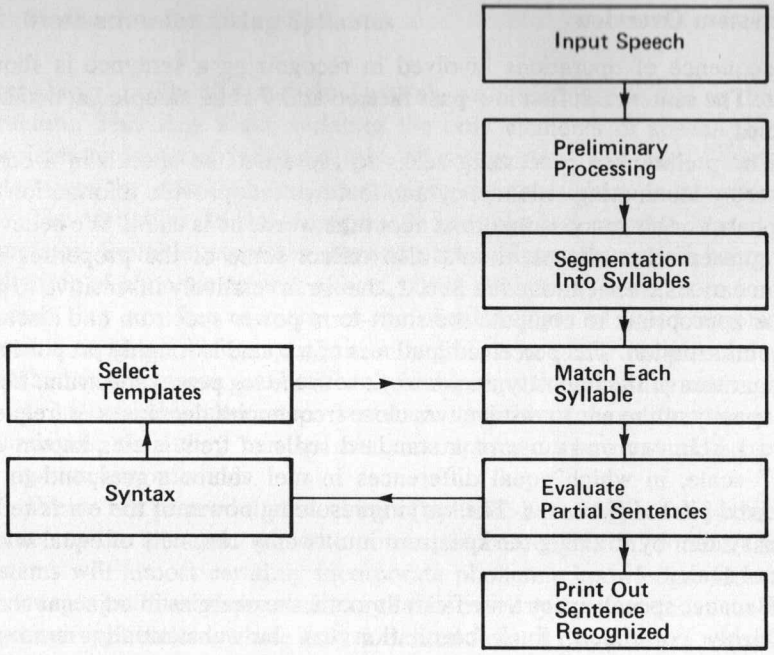


Figure 1. Processes involved in recognizing a sentence.

Frequently, the best fit to a syllable does not correspond to the correct interpretation, and it is only one or more syllables later that this becomes apparent. Some method of exploiting hindsight is therefore needed in interpreting a sentence. We do this by what is usually known as a "beam search" (Lowerre, 1976). At each stage, the most promising sentence hypothesis is held, together with a number of other possible sentence hypotheses. A fixed number of alternatives could be held, but it seems better to let the number held depend on the situation. The beam-search method keeps all hypotheses that are no more than a given threshold value worse than the current best one. The threshold value, which is a parameter in the system, therefore controls the number of hypotheses being held, i.e., the "width" of the beam. One seeks to set the beam width to be as narrow as possible while being fairly certain that a hypothesis that will turn out to fit the whole sentence best does not stray outside of the beam.

The squared distances returned by the syllable comparator are taken to be a measure of the negative log probability that the template and unknown syllable have the same interpretation. By summing the squared distances between the unknown syllables and the string of templates involved in a particular partial-sentence hypothesis, we obtain a measure of how promising that hypothesis is. If this sum exceeds that for the current best hypothesis by

more than the threshold value, then the hypothesis under consideration is dropped.

When the end of the sentence is reached, the sentence with the lowest total distance is selected. If that distance is below a threshold, the sentence is printed out; otherwise the sentence is rejected and has to be spoken again.

To construct a sentence out of a sequence of syllables, string-matching is also used, though the process is quite different from that used at the lower level in matching unknown syllables against the syllable templates. By constrast with that process, the construction of a particular sentence hypothesis requires each unknown syllable to be assigned, with a degree of confidence derived from the syllable-comparison process, to a certain syllable class. The assignment is on a strictly one-to-one basis with no possibility of deletion or insertion. The distances output by the syllable comparator can be considered to be the substitution cost incurred when an unknown syllable is replaced by a particular reference form.

4.3 Syntax

The sentences we are currently trying to recognize specify dates and times. Two examples are: "September twenty-seventh at eight thirty-two P.M." and "First May at noon." Figure 2 shows the general structure of acceptable sentences. Slightly more than a million such sentences are possible. The rules that are used to specify these sentences must satisfy two main requirements:

i) The description has to be compact (to permit use of a small computer);
ii) The syntax should be written in terms of possible transitions between states such that the possible transitions out of any state do not depend on how that state was reached (to permit use of dynamic programming in evaluating sentence hypotheses).

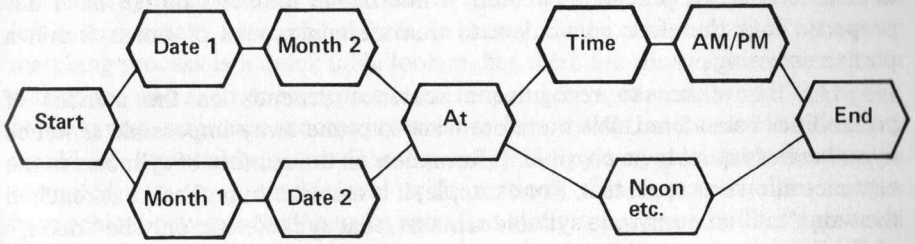


Figure 2. General structure of the date and time sentences used in this study.

Two further requirements are these:

iii) The syntax description should be as easy to read as possible;
iv) It should be easy to make changes to the general rules of the syntax.

These four requirements are not immediately compatible. To take the months as an example, it is clear that the list of months and the sequence of syllables making up each month will be the same whether the month occurs before or after the day number. It would therefore be very wasteful to specify all the month information both for the case of the month coming in first position and for its appearance in second position. On the other hand, the sequence of syllables allowed to *follow* the month depends on what has preceded it.

Conflicts of this kind have been resolved by having a four-level hierarchical syntax network. The sequence of syllables in a word forms the lowest level, and the units shown in Fig. 2 form the highest level. The levels in the hierarchy have been labeled syllable, word, word group, and phrase. In a sentence hypothesis each syllable is assigned a "state." A state is specified by four indices, which correspond to the four levels of the hierarchy and thus define the particular syllable, the word in which the syllable is embedded, and so on.

Given a syntactic state, the syntax network is used to determine all states that can grammatically follow that state. This involves horizontal and vertical searches. For example, if the state contained the first syllable of the word "seven," the network would point to the second syllable. If, however, the state contained the second syllable of the same word, the network would show that the word was complete and it would be necessary to move up one level in the hierarchy and test whether the network pointed to further words in the same phrase. The network might indicate that the phrase was definitely complete, in which case the network search would have to be continued at the phrase level, or that the phrase might or might not be complete, in which case the search would have to be continued at both the word and phrase levels.

The level designated as "word group" needs some explanation. It is used principally for the numbers in dates and times. Certain groups of numbers behave in similar ways; for example, when we specify minutes, the numbers one to nine have to be preceded by "oh," while higher numbers do not have this property. It is therefore convenient to treat certain groups of words within a phrase as a single unit.

The time taken to recognize a sentence depends on the number of possibilities considered. We therefore want to prune away impossible sentence hypotheses as quickly as possible. Information on the number of syllables in the sentence allows us to do this. For example, if a sentence hypothesis has reached the word "at" and only one syllable remains, that syllable can only be "noon," while if there are several syllables left it is pointless to consider "noon" as a possibility for the next syllable. In general, whenever the syntax proposes a

state, there is a maximum and a minimum number of syllables that will be required to complete the sentence if that state is accepted. If the number of syllables remaining to be recognized lies outside the acceptable range, then the state must be rejected. The task of determining the range for any syllable in any position does not impose large storage or computational requirements. Each "unit" (e.g., phrase, syllable, etc.) has associated with it the maximum and minimum number of syllables that can follow it before it is necessary to rise to a higher level or, in the case of phrases, before the sentence is completed. Then, for any state, the maximum and minimum number of syllables to the sentence end is given by summing these four integers over the four units making up the state.

We have so far assumed that the acoustic form of a syllable is independent of its context. While we believe this to be substantially true, there is one sense within our system in which it is decidedly not true: The syllabifier frequently misplaces a boundary so that an initial or final consonant is assigned to the adjacent syllable. For example, the words "ninth October" may have a syllable boundary before, after, or during the "th" fricative, resulting in "nine thoct," "ninth oct," and "ninth thoct," respectively. It also occasionally happens that a weak fricative is not recognized as speech, giving the fourth possibility "nine oct." The syntax copes with this by allowing the "deep-structure" form "ninth" to be realized as "ninth" or "nine," and the deep-structure form "oct" to be realized as "oct" or "thoct" (or several other forms with different transferred consonants). All forms are proposed by the syntax, but before a match against "thoct" is tried, the system checks whether the deep-structure form of the previous syllable ended in "th."

Readers familiar with Harpy may have noticed several similarities between that system and the one being described here. In particular, both systems use a hierarchical syntax description and a left-to-right beam search. One might say that our system does at a syllable level what Harpy does at a phonemic level. There is, however, one major practical difference: In our system the number of fundamental recognition units—syllables—in a sentence is quite small, but the comparison process for identifying each syllable is computationally expensive and dominates the recognition time. By comparison, time taken to explore the four-level syntax network to find all the syllables that can grammatically follow the latest syllable of each hypothesis is quite negligible, so the network can be left in its compact, hierarchical form. In Harpy, on the other hand, the phonemic matching process is a quick table lookup, but there are many phonemic units in a sentence and each one requires a reference to the syntax network. The speed of the syntax-reference process is therefore very critical, and a multilevel search would be impossibly slow. Consequently, Harpy's syntax table has to be compiled out to a single level with sections that appeared once in the hierarchical network having to be specified many times over in the single-level network. This results in a very large structure that will not easily fit into the memory of a small computer.

5. THE SYLLABLE COMPARATOR

The syllable comparator determines the distance between an unknown syllable and a stored template. As described in Sec. 4.2, the comparator is invoked by the syntax as it advances through the unknown utterance, one syllable at a time, exploring different recognition hypotheses. To the comparator, the unknown syllable is represented as a sequence of acoustic-parameter vectors, each vector corresponding to one time frame. Syllable templates are represented in the same form as syllables.

This section describes how the distance between an unknown syllable and a syllable template can be defined in such a way as to be useful for speech recognition. Distance, in this sense, is a measure of acoustic dissimilarity between two syllables or between a syllable and a template.

5.1 The Need for Time-Warping

The fundamental source of difficulty in automatic speech recognition and, in fact, what makes speech recognition a more difficult task than speech synthesis, is variability. Not only does speech vary from speaker to speaker, but even within a single speaker's output appreciable variation occurs in pronouncing the "same" utterance. One aspect of speech variation is that of timing: Syllables may be pronounced faster or slower and the relative durations of their different portions may change. The syllable comparator is designed to take into account such timing variation in speech.

Given a situation in which we are required to find the distance between an unknown syllable and a template of different duration, one possible approach would be to linearly interpolate between time frames of the shorter syllable so that the result would have the same number of frames as the longer syllable. This is equivalent to a linear dilation of the time axis of the shorter syllable. Syllable-to-template distance could then be calculated by summing the vector distances between frames of the two syllables over their duration.

The problem with this linear-dilation approach is that it is not an accurate model of what goes on when speaking rate changes. In fact, certain phonetic segments are more elastic than others and hence change more with a change in speaking rate. For example, there is evidence that vowels are more elastic than consonants (Kozhevnikov and Chistovich, 1965). Although little quantitative data are available on the subject, it is reasonable to assume that many other factors besides segment type may affect the detailed timing variations that occur in speech.

In the absence of detailed quantitative data on how the time scales of syllables are dilated and compressed, we allow nonlinear distortion of the time scales of the unknown syllable and template, choosing the frame-to-frame correspondence that gives rise to the smallest distance between the two syllables. This technique, known as *time-warping,* has been employed in

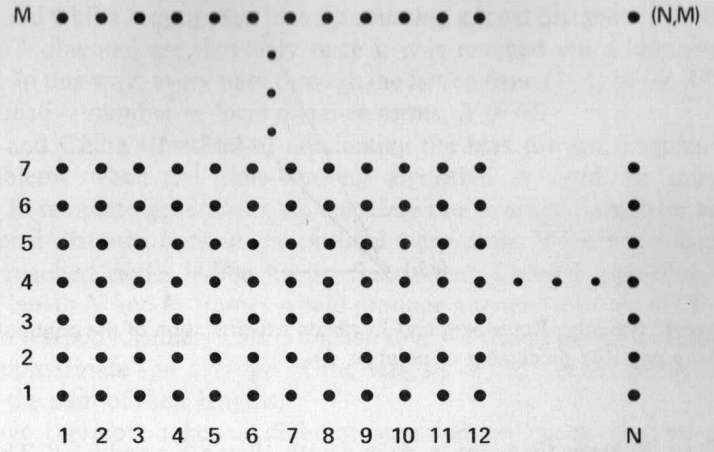


Figure 3. Lattice defined by the time frames of two syllables.

various forms (Velichko and Zagoruyko, 1970; Bridle and Brown, 1974; White and Neely, 1976; Sakoe and Chiba, 1978). In all these applications, a computationally efficient dynamic-programming algorithm simultaneously finds the optimum frame-to-frame correspondence between the template and the unknown speech segment and calculates the distance between them over that optimum frame correspondence.

In our speech-recognition system, the same time-warping algorithm used to calculate syllable-to-template distance is also used during system training to combine training syllables of the same phonetic class into composite templates. Templates are formed by averaging corresponding frames together, the correspondence being determined again by minimizing the summed vector distance over corresponding frames. The use of composite templates, generated in this way, results in better recognition performance than the use of a single typical syllable to represent the entire class. This empirical result is discussed in Sec. 5.5.

5.2 Details of the Time-Warping Algorithm

The time-warping problem may be represented graphically as a two-dimensional lattice in which the dimensions are the time axes of the two syllables being warped together, as shown in Fig. 3. Each point in the lattice has associated it with a *local distance* equal to the vector distance, appropriately defined, between the pair of parameter vectors defined by the point. The goal is to find a path through the lattice, beginning at (1, 1) and ending at (N, M), which is continuous and monotonic and which minimizes the sum of the local distances of the points it traverses. The monotonicity constraint insures that if

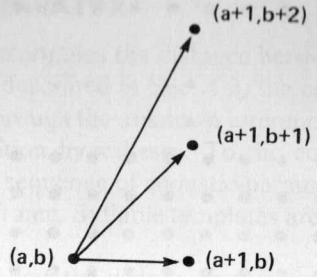


Figure 4. Rabiner, Rosenberg and Levinson's formulation of the continuity constraint, showing possible successors of point (a, b).

point (a, b) precedes point (c, d) in a path, then $a \leqq c$ and $b \leqq d$. This constraint follows from the physical requirement that warping of time scales must preserve the relative ordering of time points.

The continuity constraint has been formulated differently by different experimenters. A formulation by Rabiner, Rosenberg, and Levinson (1978), which has been widely reported on in the recent literature, allows the following three possible successor points to a point (a, b):

$$(a + 1, b), \quad (a + 1, b + 1) \quad \text{and} \quad (a + 1, b + 2).$$

This continuity constraint may be schematized as in Fig. 4. This formulation results in an asymmetric, syllable-to-syllable distance function since the distance so defined from syllable A to syllable B is not, in general, equal to the distance from syllable B to syllable A.

Velichko and Zagoruyko (1970), Bridle and Brown (1974), White and Neely (1976), and Sakoe and Chiba (1978) all propose a symmetric formulation of the continuity constraint, allowing point (a, b) to have as successors points $(a + 1, b)$, $(a + 1, b + 1)$, and $(a, b + 1)$, as illustrated in Fig. 5. Sakoe and Chiba found experimentally that the symmetric formulation performed better in recognition than did the asymmetric one, although it should be noted that their data were isolated words and not syllables of continuous speech.

It is seen that, while in the Rabiner *et al.* formulation the number of local distance terms being summed to calculate total syllable-to-syllable distance is always equal to the number N of time frames on the abscissa, the same is not true in the symmetric formulation. In the symmetric method, the number of terms being summed depends upon the path chosen through the space. It can vary between $\max(N, M)$ and $N + M - 1$. Since all the local distances are positive, the algorithm is biased in favor of paths involving fewer local distance terms, i.e., those paths involving fewer nondiagonal arcs.

Sakoe and Chiba remove this bias by counting a local distance *twice* if it is reached via a diagonal arc, but only once if it is reached via a horzontal or vertical arc. In this way, every path through the lattice from $(1, 1)$ to (N, M) has the same effective number of local distance terms, $N + M$.

Sakoe and Chiba's method of eliminating the bias toward diagonal arcs causes problems when the time-warping algorithm is used for template generation. In template generation, we calculate one average parameter vector for every local distance term in the optimal warp path. If we were to count diagonally reached nodes twice, as in Sakoe and Chiba's algorithm, two syllables of length N and M frames would produce a template of length $N + M$ frames when warped together. This is undesirable; we would prefer the template length to approximate the *average* of the lengths of its constituent syllables rather than the sum of their lengths.

We have therefore taken a different approach to insure that all paths contain effectively the same number of local distances. We use the continuity constraint, first formulated by Mermelstein (1978), that a point (a, b) may be followed by points $(a + 1, b + 1)$, $(a, b + 2)$, or $(a + 2, b)$, as illustrated in Fig. 6. It is seen that any path from $(1, 1)$ to $N, M)$ will contain the same number of local distances, $(N + M)/2$. In the case where $N + M$ is odd, we seek the minimum path from $(1, 1)$ to either $(N, M - 1)$ or $(N - 1, M)$, resulting in $(N + M - 1)/2$ local distance terms. This formulation is desirable since the resulting length (number of frames) of warping two syllables together will always be the integer average of the two syllable lengths.

It may appear from Fig. 6 that we are losing information by ignoring points $(a, b + 1)$ and $(a + 1, b)$. In fact, since spectra are generally derived from overlapping windows (in our work the overlap is 75%), successive frames are sufficiently redundant that omitting single frames rarely leads to the loss of useful information.

5.3 Time-Warping as Sequence Comparison

It will be helpful at this point to draw some comparisons between the problem of sequence comparison and the problem of time-warping. Whereas sequence

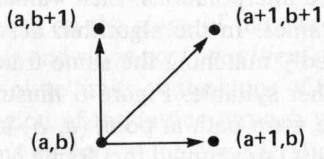


Figure 5. Symmetric formulation of the continuity constraint, showing possible successors of point (a, b).

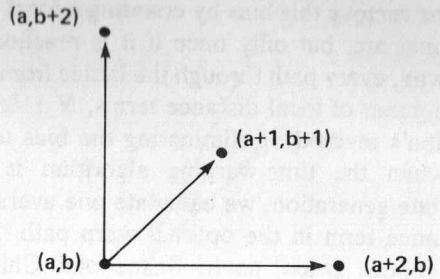


Figure 6. Continuity constraint used in this study, showing possible successor points of point (a, b).

comparison is a basically discrete problem, time-warping of speech is a discrete approximation to a continuous problem. This leads to different definitions of the cost functions.

In sequence comparison, the objects to be compared are strings of symbols from a fixed alphabet. Therefore, it makes sense to speak of deletion of a symbol, insertion of a symbol, and substitution of one symbol for another, assigning fixed costs for each. The symbols themselves are discrete units.

In speech, as we noted in Sec. 2, there are no discrete units on the acoustic level. Rather, we observe a continuous signal in the time domain from which we derive short-term spectra at arbitrary intervals (frames). Once parametrization has been accomplished, it is possible to think of each parameter vector, which denotes a point in continuous parameter space, as belonging to an infinite alphabet of parameter vectors. Substitution cost is then the appropriately defined vector-difference function we have called local distance.

In sequence comparison, where strings of discrete symbols are being compared, the operations of deletion and insertion are well defined. In speech, where the "symbols" are really samples from a continuous process, the notions of time compression and dilation are more appropriate. Deletion of speech frames is a discrete approximation to the continuous operation of time compression, just as insertion of frames is a discrete approximation to time dilation.

In order to approximate time dilation by insertion of extra frames, the inserted frames should be interpolations: their values should depend upon the values of surrounding frames. In the algorithm described in Sec. 4.2, this is accomplished by repeatedly matching the same frame in one syllable against several frames in the other syllable. Figure 6 illustrates how this is accomplished. The arrival of the warp path at point (a, b) implies that frame a of the abscissa is matched against (substituted for) frame b of the ordinate. If the path then proceeds vertically to point $(a, b + 2)$, frame a of the abscissa is also matched against frames $b + 1$ and $b + 2$ of the ordinate. Only one of the two resulting substitution costs (local distances) is summed into the total

distance, however: the one corresponding to frame $b + 2$ of the ordinate. The local distance corresponding to frame $b + 1$ of the ordinate is discarded so that the number of local distances added to determine the total distance will be independent of the warp path chosen.

This formulation captures the desired notion of time compression and expansion better than a fixed cost for deletion would. A fixed cost for deletion of a frame might be well suited to handling bursts of noise in the speech signal, but does not seem appropriate to modelling the nonlinear time translation we are trying to characterize.

What we really want to calculate in speech recognition is the likelihood that an unknown syllable belongs to a given syllable class. In speech there is no absolute standard to which an unknown syllable can be compared. We can estimate a standard for a syllable class by constructing a reference template for that class. An effective way of constructing a template is to average together through time-warping several examples of syllables belonging to the same class. Time-warping is then used to compare unknown syllables to the template in order to measure the likelihood that the unknown belongs to the class represented by the template. The fact that we calculate the distance between a template and an unknown by summing local distances implies independence between frames, which may not be perceptually warranted.

5.4 Constraints on Time Elasticity: A Preliminary Experiment

The time-warping algorithm as described so far, while not permitting changes in the ordering of parameter frames, allows unconstrained compression and dilation of the time scales of the two syllables. While speech is elastic to a certain extent, timing can also carry meaningful information used to distinguish phonetically distinct syllables. For example, the burst of noise following a [t] release, if sufficiently prolonged, might be a good match for an [s] at the beginning of a template. The capacity of timing to carry phonetically relevant information leads to the consideration of methods of constraining the warp path in such a way as to limit the amount of time compression and dilation that occurs.

Constraints on time elasticity may be defined in various ways. We experimented with two types of constraints, which in the lattice representation are interpreted as constraints on the slope of the warp path. These two types were *hard slope constraints* and *slope-cost penalties* (or *soft slope constraints*). Hard constraints place absolute limits on the slope of the path; this also has the effect of narrowing the region of the lattice through which the path can pass. This reduces computation, an important consideration when many phonetic comparisons must be made in a short time. Slope-cost penalties (soft constraints) assign a penalty or added cost to every vertical or horizontal (rather than diagonal) arc in the path (cf. Fig. 6).

We did a preliminary experiment to test the combined effect of hard and soft slope constraints on the performance of our speech-recognition system. The hard slope constraint we implemented prevents the warp path from containing two consecutive vertical or two consecutive horizontal arcs. The soft slope constraint we implemented is based on minimizing a sum over all *arcs* in the path (instead of a sum over all nodes in the path), where the value associated with each arc is the local distance of its destination node times the following factor:

$$\begin{cases} 1 & \text{if the arc is diagonal,} \\ 1.5 & \text{if the arc is not diagonal.} \end{cases}$$

Since these slope constraints cause the path to tend toward the 45° line, they are correct only when the grid is square (i.e., when the syllables contain an equal number of time frames). In the experiment described in this section, the lengths of the syllables were equalized by extending the ends of the shorter syllables with *dummy frames.* Two different types of dummy frame were tried and their effects on recognition rate were compared:

1. *Mean end frames.* The initial frames of all syllables in the training set were averaged together to create an average initial frame to use for preceding dummy frames. The average final frame was used for following dummy frames. During template generation these dummy frames were averaged into the template at the ends whenever the syllables were of unequal length.
2. *Constant differences.* These can be thought of as abstract frames whose distance to any other frame is a constant. The constant chosen was approximately halfway between the average interframe distances for a correct and for an incorrect identification. During template generation these frames were not averaged into the template. Instead, the frames of the longer syllable were simply copied into the template.

To determine how many dummy frames should be placed before, versus how many after, the shorter syllable, the total distance between the syllables was minimized as a function of this split, where distance was based on the straight-line path which is the diagonal of the square (i.e., no warping).

The two different types of dummy frame were tested with and without slope constraints; the results are shown in Table 1. For each experimental condition, a set of composite templates was created from a set of 59 training sentences, using the sequential method of template generation (see Sec. 5.5). A total of 135 templates were created from 495 training syllables. Twelve of the 59 training sentences were chosen as the test set. These particular sentences were chosen because most of them were segmented correctly by the syllabifier. They contain 95 test syllables.

Table 1. Error rates for syllable recognition without syntax under two constraint conditions and two dummy-frame conditions ($N = 95$).

Dummy frames	Slope constraints			
	Both constraints		No constraints	
	No.	%	No.	%
Mean end frames	13	(14%)	11	(12%)
Constant difference	31	(33%)	28	(29%)

In order to focus on the deficiencies in the syllable comparator, the recognition system was tested without using any syntactic information. In particular, each test syllable was treated as an unknown and its distance to each of the 135 templates was calculated using the time-warping algorithm. The identification was treated as correct only if the single correct template had the smallest distance. Of course, this error estimate is pessimistic both because no syntax is used, and because some of the errors reflect confusion between templates for different pronunciations of the same syllable, which would not cause any error in practice.

It can be seen from Table 1 that the type of dummy frame used has a much larger effect on performance than the absence or presence of slope constraints. At the same time we see that the effect of slope constraints, although small and probably not statistically significant, is to degrade performance in both dummy-frame conditions.

We can explain the large difference in performance between the two dummy-frame conditions by calling attention to a phenomenon we have not yet mentioned but which has consistently obtained in the data we have examined: When we compare different tokens of the same syllable, we find greater variability among tokens near the edges of the syllables than near the middle. By generating templates from syllables extended with mean end frames, we are averaging the most variable vectors of longer syllables with constant vectors. When shorter syllables are then compared with the template, the ends are extended, with the same constant vectors giving rise to small local distances at the edges of the syllable. The use of a constant vector as a dummy frame therefore attenuates the effect of the high variability of end frames. Since abstract constant-difference frames are never averaged into the template and no actual distance is calculated, they cannot have this effect.

We conclude from the results in Table 1 that the slope constraints proposed above do not improve recognition performance, at least in conjunction with the dummy-frame extension procedure used here. Future work should explore more elaborate slope constraints applicable to rectangular grids.

Initial tests on these data of a rectangular grid algorithm without slope constraints produced 13 misrecognitions (14%), a performance one percentage point worse than the best performance condition of Table 1. This unconstrained, rectangular time-warping algorithm, because it combines good performance with simplicity and intuitive appeal, was chosen as the standard algorithm for use in the recognition system. The experiment described in the following section uses this algorithm.

5.5 Template Generation

Composite templates are generated from a set of training sentences. The training sentences are preprocessed and automatically segmented by the syllabifier. Each sentence is then transcribed by a human operator using the ARPAbet, an ASCII phonetic alphabet: Sentences are played back one syllable at a time and the phonetic transcription of each syllable is entered at the terminal. After verification, the system combines all the syllables corresponding to a distinct ARPAbet transcription into a single composite template associated with that transcription. This section describes and compares three alternative processes for creating one composite template from a set of phonetically equivalent syllables. The first template-generation procedure that was implemented combines syllables sequentially. In this procedure, the time-warping algorithm is used to determine an optimal frame-to-frame correspondence between the first two syllable tokens of the phonetic class. Corresponding parameter vectors are averaged to form a template of weight = 2 (the *weight* of a template is the number of syllable tokens from which it was derived). The warping algorithm is invoked again to determine the optimal frame correspondence between the template of weight = 2 and the third token of the phonetic class. Parameter vectors of corresponding frames are averaged as before, except that vectors belonging to the template are weighted twice as heavily as those belonging to the token, since they contain information from two tokens. The resulting template has weight = 3. The process is continued until all the syllable tokens in the phonetic class have been combined.

When we combine a new syllable token with a template, we weight the vectors of the template more heavily so that each training token in the phonetic class contributes equally to the final template. Because of the symmetric nature of the time-warping algorithm, however, the token and template contribute equally to the warp path. This means that the last token combined with the template affects the temporal structure of the final template as much as all the other tokens combined. Since the temporal structure of the syllable is to some extent neutralized by the warping process anyway, this problem is less serious than it may appear at first, although it should not be ignored.

Conceptually, we would like to make the template proportionally less flexible as its weight increases, although it is not clear how this could be done. It is theoretically possible to warp together all the syllable tokens in a phonetic class simultaneously by formulating the problem in a lattice space of dimensionality equal to the number of tokens involved. This solution was considered too computationally expensive for our purpose. Instead, we approximate this generalized averaging process by sequential combinations. The approximation seems fairly good: Davis (1979) found that for the slope-constrained, symmetric warping algorithm, differences in the order of sequential combination had no significant effect on recognition performance.

Table 2. Performance of the recognizer without syntax, using three different template-generation procedures.

Template generation procedure	Number of misrecognitions ($N = 495$)	Percent misrecognition
Sequential-combination	92	19%
Binary-combination	84	17%
Single-token templates	159	32%

In order to eliminate temporal distortion due to sequential combination, a second template-generation procedure was implemented, which first combines tokens in pairs. Then the resulting templates of weight two are combined in pairs, resulting in a set of templates of weight four. This process of *binary combination* is continued until a single template results. If the original number of tokens is a power of two, no warp between templates of different weights occurs. Otherwise, at least one warp between templates differing in weight by unity must occur. This is still a much smaller weight difference than in the sequential-combination procedure.

A third approach to template generation, which avoids problems of combination, is to choose a single typical token to represent a phonetic class. An experiment was performed to see whether composite templates performed better or worse than single token templates. One syllable token from each phonetic class was selected as a template. In each class, the syllable selected was the one whose summed distance to the other syllables in the class was minimum.

Sequential-combination, binary-combination, and single-token template-generation procedures were tested on the set of 59 training sentences as in the previous section, except that, instead of choosing 12 test sentences, all 59 sentences were included in the test set, giving 495 test syllables. The results from this experiment are shown in Table 2.

The table shows that composite templates perform dramatically better than single-token templates, even when the single-token templates are chosen optimally. This large improvement is consistent with results from a similar experiment in which automatically generated segmentation boundaries were verified and hand-adjusted where necessary (Mermelstein, 1978). We also see

Table 3. Recognition performance and number of matches carried out as a function of beam-width threshold.

Beam width threshold (Arbitrary squared-distance units)	No. of incorrect sentences (out of 53)	No. of matches	No. of matches per syllable
1	16	5631	10.2
50	9	5841	10.6
100	5	6069	11.0
300	3	6783	12.3

that binary combination, which gives approximately equal weight to the time information of all its constituent tokens, performs somewhat better than does sequential combination. This small difference, which is probably not significant, is consistent with Davis's findings on order dependence.

6. SYSTEM PERFORMANCE

In order to test the performance of the system, 59 date-and-time sentences comprising approximately 550 syllables were recorded to form the training set, and two months later 59 similar sentences were recorded by the same speaker to form the test set. In each set the syllabifier generated syllable-boundary positions in six sentences that were incompatible with the syntax. Of the remaining 53 sentences, 50 were fully correctly recognized when a wide beam was used in the beam search.

On a per-syllable basis, there was a 99% acceptable syllabification, namely six errors out of 550 but, because a single serious syllabification error makes recognition of a sentence impossible, a 1% error rate led to roughly 10% of the sentences being lost to the system because of the syllabifier. Since an acceptable sequence of templates can seldom be found to fit an incorrectly syllabified sentence, most such sentences will be rejected rather than misrecognized. In this experiment it turned out that five out of the six incorrectly syllabified sentences in the test set could be rejected. Moreover, the probability of unacceptable syllabification is primarily a function of sentence length rather than of task difficulty and would not be expected to increase with increasing vocabulary size.

We carried out some experiments to see how performance depended on beam width. The simplest sentence-hypothesis strategy consists of selecting and retaining at each stage the single best match among the syllables proposed by the syntax. Strategies that are able to use hindsight in selecting their sentence

hypotheses should correctly recognize a larger proportion of sentences but at the cost of greater processing times. An upper limit to the performance of the hypothesis-evaluation strategy is reached when the only errors occurring are single-syllable substitutions (e.g., "sixteen" recognized as "fifteen"). Table 3 shows the performance of our system on the 53 correctly syllabified sentences. The three errors remaining at large values of the threshold (i.e., using a wide beam) are all single-syllable substitutions. Consequently, no other evaluation strategy could produce better recognition performance on our data using our syllable-recognition algorithm.

The processing time is approximately proportional to the number of syllable-to-template matches that have to be carried out. It can be seen in the table that the number of matches is not very sensitive to beam width. The beam width at which performance improvement saturates requires only 20% more syllable matches than the fastest strategy using the narrowest possible beam width, yet the number of errors is reduced by a factor of more than five.

Recognition was correct for 546 out of 549 syllables, i.e., 99.5%, leading to correct recognition of 50 out of 53 acceptably syllabified sentences, i.e., to a recognition performance of 94%.

7. CONCLUSIONS

In providing an example of how dynamic programming is being applied to continuous speech recognition, we feel that the ideas should stand on their own; detailed consideration of the overall system performance is not particularly helpful. Overall performance depends critically on many factors not discussed here: the syllabifier, the spectrum range and spectrum parametrization used, and the form of local distance measure, among others. Moreover, our work is at an early stage, and our system performance has neither been optimized nor adequately evaluated. Nevertheless, our results so far at least give us grounds to believe that our approach may be practical.

The vocabulary we used is small by comparison with those used by some other groups and our sentences are relatively short. However, this is balanced by a relatively free syntax and by the fact that many word pairs (e.g., "sixteen" and "fifteen") differ in only one syllable and share the same vowel. Experience so far suggests that close to real-time performance would be possible on fairly modest hardware.

We have aimed in this article to show how one group among many others is applying string-matching techniques to speech recognition. Such techniques have contributed much to speech recognition already and will no doubt continue to do so. We hope, however, that it has become clear that speech has special properties that have to be taken into account. Rather than mutilating the speech to fit some particularly attractive mathematical model, it is important to select suitable techniques carefully and adapt them to the speech signal.

REFERENCES

Bahl, L.R. Bakis, R., Cohen, P.S., Cole, A.G., Jelinek, F., Lewis, B.L., and Mercer, R.L. Recognition Results with Several Experimental Acoustic Processors, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Washington, pp. 249–251, 1979.

Bridle, J.S., and Brown, M.D., An Experimental Automatic Word-Recognition System, *Interim Report, JSRU Report No. 1003,* Joint Speech Research Unit, Ruislip, England, 1974.

Bridle, J.S., and Brown, M.D., Connected-Word Recognition using Whole-Word Templates, *British Institute of Acoustics Autumn Conference*, November 1979.

Bridle, J.S., and Sedgwick, N.C., A Method for Segmenting Acoustic Patterns, with Applications to Automatic Speech Recognition, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Hartford, pp. 656–659, 1978.

Davis, S.B., Order Dependence in Templates for Monosyllabic Word Identification, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Washington, pp. 570–573, 1979.

De Mori, R., Automatic Phoneme Recognition in Continuous Speech: A Syntactic Approach, *Proc. of NATO Advanced Study Institute on Spoken-Language Generation and Understanding,* Bonas (Gers), France, 1979.

Kashyap, R.L., Syntactic Decision Rules for Recognition of Spoken Words and Phrases Using a Stochastic Automaton, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-**1**(2), pp. 154–163, April 1979.

Klatt, D.H., Review of the ARPA Speech-Understanding Project, *J. Acoust. Soc. Am,* **62**(6), pp. 1345–1366, December 1977.

Kozhevnikov, V.A., and Chistovich, L.A., *Speech: Articulation and Perception.* Washington: Joint Publications Research Service, 1965.

Levinson, S.E., and Rosenberg, A.E., A New System for Continuous Speech Recognition—Preliminary Results, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* Tulsa, pp. 239–246, 1979.

Lowerre, B.T., The Harpy Recognition System, Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, 1976.

Martin, T.B., Practical Applications of Voice Input to Machines, *Proc. IEEE,* **64**(4), pp. 487–500, April 1976.

Mermelstein, P., Automatic Segmentation of Speech into Syllabic Units, *J. Acoust. Soc. Am.,* **58**, pp. 880–883, 1975.

Mermelstein, P., Recognition of Monosyllabic Words in Continuous Sentences using Composite Word Templates, *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 708–711, Hartford, 1978.

Rabiner, L.R., Rosenberg, A.E., and Levinson, S.E., Considerations in Dynamic Time-Warping Algorithms for Discrete-Word Recognition, *IEEE Transactions on Acoustics, Speech, and Signal Processing,* ASSP–**26** (6), pp. 575–586, 1978.

Reddy, D.R., Speech Recognition by Machine: A Review, *Proc. IEEE,* **64**(4), pp. 501–531, April 1976.

Sakoe, H., and Chiba, S., Dynamic-Programming Algorithm Optimization for Spoken-Word Recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing,* ASSP-**26** (1), pp. 43–49, Feb. 1978.

Velichko, V.M., and Zagoruyko, N.G., Automatic Recognition of 200 Words, *International Journal of Man–Machine Studies,* **2**, pp. 223–234, 1970.

White, G.M., and Neely, R.B., Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming, *IEEE Transactions on Acoustics, Speech, Signal Processing,* ASSP-**24** (2), pp. 183–188, April 1976.