PROCEEDINGS OF THE IEEE, VOL. 64, NO. 4, APRIL 1976

Continuous Speech Recognition by Statistical Methods

FREDERICK JELINEK, FELLOW, IEEE

Abstract—Statistical methods useful in automatic recognition of continuous speech are described. They concern modeling of a speaker and of an acoustic processor, extraction of the models' statistical parameters, and hypothesis search procedures and likelihood computations of linguistic decoding. Experimental results are presented that indicate the power of the methods.

I. Introduction

THIS PAPER DESCRIBES statistical methods of automatic recognition (transcription) of continuous speech that have been used successfully by the Speech Processing Group at the IBM Thomas J. Watson Research Center. The sources of these procedures will be referenced where practicable, but the working style of the Group has been deliberately cooperative (as the Acknowledgment Section indicates), so a certain amount of inadequate or unjust crediting is inevitable. The author tried his best to keep it at a minimum.

The exposition, appearing as it does in an IEEE publication, is aimed mostly at engineers who are less familiar with speech and language than with information transmission, statistics, or signal processing. At the same time, the author would like to enable speech specialists to read the more mathematical parts of the paper. Inevitably, a compromise between these two audiences has been attempted that resulted in a somewhat lengthened presentation. The author would like to invite his readers to skip rather boldly over material familiar to them. The first six sections contain the essence of the formulation that is centered on the design of an actual speech recognition system. Readers of Section VI that contains experimental results may feel somewhat dissatisfied with the fact that no comparisons are attempted with performance achieved by alternate design philosophies. Unfortunately, such judgments are made difficult by the great variety in utterance corpora to be recognized, in experimental conditions, and in recognizer function goals.[1] However, the accompanying survey paper by Reddy [23] does assess the merits of the various speech recognition projects, and can serve as an excellent introduction to the field for the nonspecialist.

In the speech recognition community, our project is somewhat controversial, since it attempts to model utterance production statistically, rather than through a grammar that would describe syntactically and semantically the allowable (mini-) universe of discourse. It is much too early to tell which emphasis is sounder. There is little doubt that before automatic recognition of speech is accomplished, the statistical utterance models used will incorporate more grammatical features, and statistics will have been grafted onto grammatical models. Most methods presented here concern modeling of the speaker's and acoustic processor's performance and should, therefore, be universally useful.

Automatic recognition of continuous (English) speech is an attempt to use computers to transcribe naturally spoken utterances (i.e., without artificial pauses between phonemes, syllables, words, or sentences) in accordance with the rules of English orthography.

Fig. 1 diagrams this process when it is assumed that a speaker is a transducer that transforms into speech the text of thoughts he intends to communicate.[2] The acoustic signal put out by the speaker is first transformed into some digital string by an *acoustic processor*. That string is then analyzed by the *linguistic decoder* whose output is the best estimate (in a probabilistic sense) of the text "inputted" into the speaker. For its analysis the linguistic decoder needs a model of text generation by the source (the *language model*), of phonetic production by the speaker, and of the acoustic processor's performance. It is our task to describe these models and to show how to estimate their statistical parameters.

The acoustic processor is simply that part of the speech recognizer that is designed without regard to the language model or to the phonetic (as opposed to acoustic) behavior of the speaker. In different systems it varies in complexity from a simple analog-to-digital converter, through a spectrogram generator, to a device that attempts to put out the string of phones generated by the speaker.

The diagram of Fig. 1 is reminiscent of the usual description of an information transmission system that involves a source, a channel, and a decoder. Indeed, our methods are in great part inspired by those developed in information theory.[3] Thus we will regard the speaker–acoustic processor combination as an *acoustic channel*, albeit with somewhat unusual statistical properties. The *text generator* will be viewed as a combination of the usual *information source* with a *channel encoder*. The search procedures of the linguistic decoder will be based on the *stack algorithm* of sequential decoding [1], [2].

In this paper, we will make no attempt to account for the "ordinary" kind of noise that would result from other sound sources, room reflections, electrical interference, etc. Our speech will be assumed to take place in an acoustically treated room. The recognition system will be designed for a known speaker whose acoustic and phonetic parameters have previously been determined. Finally, we will not consider questions

Manuscript received July 5, 1975; revised October 15, 1975.

The author is with the Speech Processing Group, Computer Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

[1] Recording can be carried out with good microphones in acoustically treated rooms or in noisy surroundings over the telephone. Recognizer cost may or may not be important. Real time recognition may or may not be aimed at. The system may be designed for a single or for multiple speakers. A command language with a simple syntax, or a natural language corpus may be used.

[2] Many would dispute this description of a speaker's behavior. For instance, it does not take into account the way in which his speech would reflect the varying difficulty with which his thoughts were formulated. Hopefully, our formulation approximates well enough the behavior of an announcer reading text that she has himself previously written (it is difficult to be a nonsexist).

[3] For a simple exposition of this subject, see the book by Abramson [19]. A more thorough discussion can be found in [20] and [21].

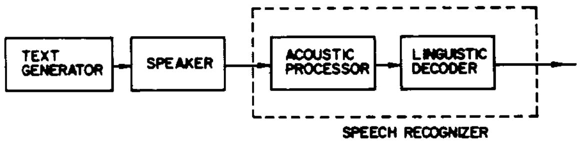

Fig. 1. A schematic diagram of the automatic speech recognition process.

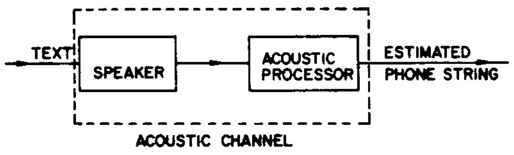

Fig. 2. The phone-based acoustic channel model.

of real time implementation. Thus we shall ignore difficult problems that will have to be addressed before practical continuous speech recognition becomes a reality.

This paper is divided into ten sections. The first six contain an integrated discussion of the "main-line" IBM speech recognizer, including its experimental evaluation. They attempt to concentrate on the main design ideas and to minimize mathematical difficulty. Section II describes a phone-based acoustic processor and develops its simple statistical model. Section III starts by discussing some problems of phonetics and introduces a base form and rule oriented model of speaker phonetic performance. Section IV combines the acoustic processor and phonetic performance models into an integrated word-oriented acoustic channel model whose parameters can be statistically estimated. The automatic estimation process itself is described in Section VII to which the reader may skip, if he so desires. Section V completes the recognizer description by addressing the problem of linguistic decoding. In Section VI are presented results of experiments evaluating the performance of the recognizer.

The remaining four sections of the paper are more specialized and mathematically oriented. Section VII presents and evaluates a method of acoustic channel model parameter estimation based on the forward-backward algorithm described in detail in Appendix III. The latter is in some sense an elaboration of the well-known Viterbi algorithm [11], [12] presented in Appendix II. Section VIII develops a method of channel transmission probability evaluation needed for efficient linguistic decoding. In Section IX, we introduce two word-oriented acoustic channel models that do not attempt phone recognition. They are based on a very simple acoustic processor that requires no elaborate data flows, decision algorithms, or parameter adjustments. The models are automatically trainable in a manner described in the concluding Section X, and their recognition performance is encouraging.

II. A Phone-Based Acoustic Processor

A standard acoustic processor attempts to transcribe speech into a string of symbols taken from a moderately large (about 60 for English) phonetic alphabet. The underlying assumption is that the symbols can be selected so that in some appropriate space their acoustic shapes fall into separable regions and that sets of strings of these symbols can be used to describe pronunciations of continuously spoken words. This approach naturally leads to a channel description consisting of two cascaded parts: the speaker performance model followed by the acoustic processor model (see Fig. 2).

Such an acoustic processor is intended to act essentially as a data compression device whose output symbol sequence carries sufficient information to allow identification of the utterance that gave rise to it. We would like consecutive symbols to be directly related to consecutive intervals in the acoustic waveform, and as far as possible, we would like an output symbol to depend only on its own interval. Finally, we would like the symbol alphabet to be as small as practicable. This leads to

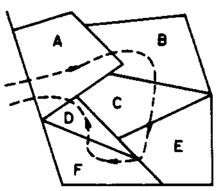

Fig. 3. Example of an utterance trajectory through a simplified acoustic space partitioned into phone regions.

the natural requirement that if the waveforms within two time intervals are sufficiently similar, the acoustic processor should designate them by the same symbol.

Consider some suitable acoustic recognition space. To any utterance spoken, there will correspond a trajectory through the space. One would wish to partition the space into regions such that the specification of the succession of regions occupied by the trajectory would be sufficiently characteristic of the utterance to distinguish it (with high probability) from other utterances. Fig. 3 attempts to illustrate the preceding idea.

Phonologists have attempted to find the smallest set of sound units, *phonemes*, sufficient to distinguish among different utterances. For instance, /*p*/ and /*b*/ are distinct phonemes of English, because they distinguish such words as *pin* and *bin* from each other.[4] However, it should not be thought that acoustic intervals labeled by the same phoneme would necessarily sound alike. Different sounds may be *allophones* of the same phoneme if at least one of two conditions is met that prevents them from keeping utterances apart. Two allophones either never occur in the same sound environment (such as the aspirated word initial *p* of *pot* and the unaspirated final *p* of *top*) or if they do, the substitution of one for the other does not produce a different word, but merely a different pronunciation of the same word (as an example, the *glottal stop* "ʔ" is a free variant of "t" for the Cockney pronunciation of words like *city*).

It is clear from the above, that while a phonemic alphabet would satisfy most of the requirements of the second paragraph of this section, it will have to be somewhat enlarged to lend itself to convenient pattern recognition by an acoustic processor. First, the region of the recognition space used that corresponds to the acoustic variants of a given phoneme will have to be partitioned into a minimal number of compact, convex subregions, each subregion to be labelled by a different symbol, referred to as *phone*. Second, if two convex subregions corresponding to phones of different phonemes should substantially overlap, the overlap will be excluded from both phones and given its own phone designation. Third, it will be advantageous to provide additional labels for such regions of the recognition space which the acoustic processor is particularly capable of identifying. Finally, since the design of the structure of the speaker production model and of the acoustic

[4] For an introductory discussion of phonetics see, for instance, Lyons [5, pp. 99-132].

processor should be guided by phonetic experience, it will be desirable to keep the partitioning system such that the obtained subregions are made up of more or less traditional perception units.

In Appendix I, we list the phone alphabet used by the IBM speech recognizer. A schematic diagram of the *IBM Modular Acoustic Processor*[5] (MAP) is given in Fig. 4. The first box, called Second Parametrically Controlled Analyzer (SPCA) transforms the acoustic input into an essentially spectrographic output to which are added energy, pitch, and spectral change indicators. The spectrogram is based on a short-term FFT analysis of the prefiltered[6] acoustic waveform. A 20-ms wide Hamming window is used and an 80-element spectral vector, called Spectral Time Sample (STS), is produced once every centisecond. The computation of the latter is based on a 20-kHz sampling rate of the acoustic waveform.

The output of the SPCA is fed into the Hierarchically Operated String Transcriber (HOST) that consists of three stages: *segmentation*, *dynamic segment classification*, and *event consolidation*. For each phone, the *segmenter* stores an STS-type prototype. The relative similarity of the STS input string to the prototypes, as well as the energy and spectral change indicators put out by the SPCA are used by the segmenter to mark the center of each phone. The same data are also used to generate one list of five most likely phone classifications for each center-mark. The STS subsequences between inserted center-marks are normalized by interpolation to fixed image width, employing a measure of spectral change as the normalizing function. Measurements of similarity of these *transemes* to stored prototypes, one for each phone pair, are used by the *dynamic segment classifier* to generate for each center-mark two additional lists of five most likely phones. The first of these lists corresponds to measurement results on the transeme preceding the center-mark, and the second to those following the center-mark. Finally, the above three lists for each center-mark are examined by the *event consolidator* which puts out its best estimate of the identity of the corresponding phone.

The MAP processor may make two kinds of errors: it may misplace center-marks and it may mislabel phones corresponding to placed center-marks. The channel model of Fig. 2 assumes that both inputs and outputs of the acoustic processor are phone strings. If, as a first approximation, we suppose MAP to be memoryless, then a single phone input may result in zero, one, or more phone outputs. We then say that a deletion, substitution, or insertion, respectively, has taken place. The probabilistic finite state machine of Fig. 5 is a simple statistical model for the output generating process initiated by a single input phone. It is started in the initial state S_1 and terminated when the final state S_3 is reached. The dashed *null* transition corresponds to no output, each solid transition to a single phone output. To each solid transition $S_i \rightarrow S_j$ there corresponds an output probability distribution $q_\alpha(\beta|S_i \rightarrow S_j)$ over the phones β. The subscript α denotes the phone input to the acoustic processor and identifies the machine. Denoting the probabilities of deletion, substitution, and insertion by $P_\alpha(D)$, $P_\alpha(S)$, and $P_\alpha(I)$, respectively, the probability that the output pair β_1, β_2 resulted from the input

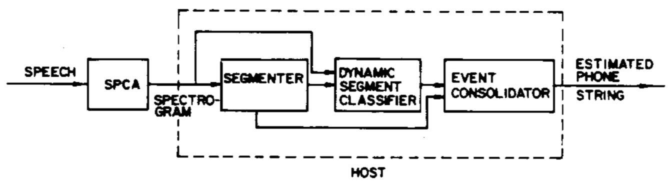

Fig. 4. Schematic diagram of the MAP.

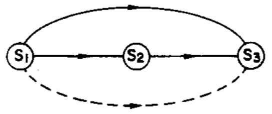

Fig. 5. A finite-state machine model of MAP output symbol generation due to a particular symbol input.

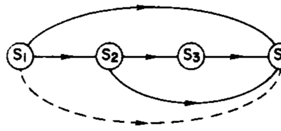

Fig. 6. A more complex model of the MAP.

α is then given by

$$P_\alpha(I)\, q_\alpha(\beta_1|S_1 \longrightarrow S_2)\, q_\alpha(\beta_2|S_2 \longrightarrow S_3). \tag{1a}$$

Similarly, the probability of the single output γ is

$$P_\alpha(S)\, q_\alpha(\gamma|S_1 \longrightarrow S_3) \tag{1b}$$

and that of no output is

$$P_\alpha(D). \tag{1c}$$

In practice, for some input phones MAP occasionally produces three or more outputs and so the model of Fig. 6 is somewhat more realistic. One obvious shortcoming of both models is their memoryless nature. Not only is real phone production influenced by context, but the incorporation of the dynamic segment classifier into MAP itself produces at least pairwise dependence.[7]

We will postpone the somewhat involved description of an automatic method deriving estimates for the statistical parameters q and P involved in (1), until Section VII.

III. A Model of Speaker Phonetic Performance

We next come to the model of speaker phonetic performance. For any word string input, it should generate a corresponding phone string output with a probability reflecting the speaker's production propensities. It is worth stressing again that the purpose of all our models is to facilitate auto-

[5] The details of the design and philosophy of the MAP can be found in a series of papers by Dixon and Silverman [6]-[8]. We will limit ourselves here to only a rudimentary description.

[6] The purpose of the filtering is to preemphasize high frequencies, to compensate for peculiarities of the speaker's vocal tract, to suppress background noise, and to prevent aliasing.

[7] A possible, though considerably more complicated model that incorporates pairwise dependence is given in Fig. 7. There are three starting states S_1, S_2, S_3 and three final states S_1', S_2', S_3', and the particular machine chosen to generate outputs depends on input pairs α_1, α_2. Output probability distributions are associated with all solid transitions. The starting state S_3 is the deletion state: no outputs result from transitions out of it. From the state S_1 it is possible to reach any of the final states, and also the starting state S_2. If the final state S_i' is reached, the next machine corresponding to the input pair α_2, α_3 starts its operation out of the starting state S_i. The correspondence of states to "reality" is as follows. The outputs from machine α_1, α_2, are to be thought of as those resulting from the phone input α_1 and "influenced" by α_2. S_1 is the starting state if the center-mark for α_1 was correctly placed. The transition will be to S_2' if the center-mark for α_2 was placed early, it will be to S_1' if that center-mark is accurate, to S_2' if it is placed late, and to S_3' if the center-mark was omitted. Similarly, the starting state will be S_2 if the center-mark for α_1 was placed late, and it will be S_3 if it was omitted. Fig. 8 attempts to diagram the situation.

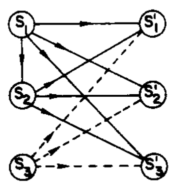

Fig. 7. A MAP model incorporating possible output dependence on pairs of consecutive input phones.

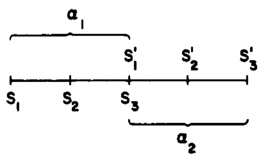

Fig. 8. A diagram of the conceptual time segment correspondence of the states of Fig. 7 to the input phones α_1, α_2.

matic speech recognition and not to account for any linguistic and other phenomena for their own sake.

It was already implied in Section II that the variations in phonetic word realizations by even a single speaker are of at least two kinds. Those[8] like "ajəl," "ajīl" (for *agile*) or "ēthər," "īthər" (for *either*) are specific to the words themselves and no profitable generalization over all English can be made.[9] On the other hand, those variations like "hān labəliŋ" (*hand labeling*), "ban lēdər" (*band leader*), "fān memərēz" (*fond memories*), and "sȯfness" (*softness*) [examples of consonant cluster simplification], or "drȯriŋ" (*drawing*) and "kyübə ran Africə" (*Cuba and Africa*) [examples of intrusive "r" insertion]; or "fis sanwich" (*fish sandwich*) and "gash shordij" (*gas shortage*) [examples of sibilant assimilation] represent applications of optional rules valid for the entire language.

In accordance with the above observations, we will organize our *speaker performance* model as a cascade combination of a *base form selector* followed by a *phonetic rule applier* (see Fig. 9).[10] The former selects for each word one of a number of its basic pronunciations, called *base forms.* The set of these was chosen with a given speaker in mind so that every phonetic variant of an utterance (*a surface form*) could be generated by applying phonetic rules to the string of phonemes formed by a concatenation of appropriately selected base forms that correspond to the words of the utterance. With each base form there is associated the probability of its being used, given that the corresponding word was uttered.[11] By a word we mean here its spelled form together with a part-of-speech and possibly a semantic marker. Thus ⟨produce, N⟩ but not ⟨produce,

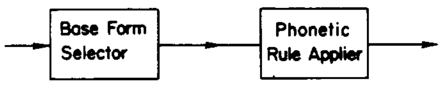

Fig. 9. A speaker phonetic performance model.

V⟩ will have as one of its base forms "prädyüs|," but both will have "prōdyüs|." Note the appearance of the word end markers "|" which are needed to ensure proper operation of the rule applier. A base form is then a string of symbols that are either phonemes or end markers such as "|" or the affix marker "#" in "prōdyüs#iz|" (which is a base form belonging to ⟨produces, V⟩).[12]

Let $b_1(w), b_2(w), \cdots, b_k(w)$ be the set of base forms corresponding to the word w, and let $P_w(i)$ be the probability that the i^{th} $(1 \leqslant i \leqslant k)$ base form will be "used by the speaker" when pronouncing the word w. If the sentence $w_1 w_2 \cdots w_n$ is put into the base form selector, then the phoneme sequence (*base string*)

$$b_{i_1}(w)\, b_{i_2}(w_2) \cdots b_{i_n}(w_n)$$

will be put out with probability

$$\prod_{j=1}^{n} P_{w_j}(i_j).$$

The *surface phone string* is obtained by applying (in general optional) phonetic and phonemic rules to the base phoneme string. As indicated in the first paragraph of this section, the rules attempt to characterize such phonetic phenomena as fast speech, dialect differences, word juncture modification, etc.[13] They are nonordered, noncyclic, left-to-right and context sensitive, and have the general form

$$\alpha \longrightarrow b/c\_\delta. \tag{2}$$

In (2) the Greek letters α and δ are base phoneme strings, while the Roman letters b and c are surface phone strings. Rule (2) says that the string α can be transformed into string b provided the string immediately preceeding α is c and that immediately following α is δ. The rule applier is fed some base sequence ξ_0. Its initial subsequences are then examined for possible application of rules (there will always exist at least one rule that applies). One of these is chosen to be executed and the string is transformed into a new string $x_1\xi_1$ where x_1 is a surface string and ξ_1 is a base string. Next, a rule is executed on some initial subsequence of ξ_1 and as a result $x_1\xi_1$ is transformed into $x_2\xi_2$ where $\|x_2\| \geqslant \|x_1\|$ and $\|\xi_1\| \geqslant \|\xi_2\|$ (here $\|x\|$ denotes the length of the string x). The process continues until no base string remains.

As an example, suppose that some of the rules are

I.	$\alpha \longrightarrow b/\_$	V.	$\beta \longrightarrow N/\_\gamma$
II.	$\alpha\beta \longrightarrow a/\_\gamma$	VI.	$\beta\gamma \longrightarrow b/a\_$
III.	$\alpha\beta \longrightarrow ab/d\_$	VII.	$\beta\gamma \longrightarrow c/b\_\alpha\gamma$
IV.	$\alpha\beta\sigma \longrightarrow e/d\_$	VIII.	$\gamma \longrightarrow d/\_\alpha$

and that the string $\underline{\alpha\beta\gamma\alpha\beta\gamma}$ is put out by the base form selector. First either rules I or II apply (the left context requirements

[8] We use the Webster's Third New International Dictionary system of phonetic transcription rather than IPA, in order to make our examples as accessible as possible.

[9] No reasonable context specification can be found in which "ē" could always be replaced by "ī", or vice versa. This is attested to by the fact that we have "prehens ə" (for *prehensile*) but not "prehensīl," and "pərsentīl" for (*percentile*) but not "pərsentəl." Similarly, we have "brēthər" for (*breather*) but not "brīthər," and "līthər" for (*lither*) but not "lēthər".

[10] For a thorough discussion of the workings of the model as well as for its phonological underpinnings see the paper by Cohen and Mercer [9].

[11] Again, the limitation of this essentially memoryless formulation is obvious. In reality, the mood that induced a speaker to say "təmādō" might induce him to say "īthə" and possibly even "pətäd.o", if these words occur close by. To disregard this fact will be much less serious than the lack of memory in rule application that will also be postulated. It is, however, very hard to see how the appropriate statistics could be extracted in practice if memory was introduced into the base form selector model.

[12] Actually, the base form lexicon used at IBM is somewhat more complex. For instance, stress information is included. However, the above outline gives the essential flavor.

[13] Note however, that intonation and other prosodic features cannot be handled in this way, since they surely depend on the utterance syntax, and perhaps semantics. At this moment the IBM recognizer puts out no prosodic markers, and the modeling formulation is not easily extendable to handle these.

of rules III and IV are not met). If rule I is chosen for execution, the string becomes $\underline{b\beta\gamma\alpha\beta\gamma}$. Now rules V and VII apply. If V is executed, the string becomes $\underline{b\gamma\alpha\beta\gamma}$ (N stands for a null string so V is a deletion rule). Next we necessarily get $\underline{bd\alpha\beta\gamma}$, and consequently the choice will be between rules I through IV. If IV is chosen, the surface string $\underline{bde}$ results and the process terminates.

Rule execution is carried out probabilistically. To every conceivable situation $x\xi$ there corresponds a subset of rules $\mathcal{S}(x\xi)$ that may apply. These are rules whose left-hand side is some η such that $\xi = \eta\zeta$, whose left context is some final substring of x, and whose right context is some initial substring of ζ (any of the substrings mentioned may be empty). Since the total number of rules is finite, so is the number of distinct rule subsets. To each rule subset $\mathcal{S}(x\zeta)$ there corresponds a probability distribution $P(\ |\mathcal{S}(x\zeta))$ over the rules. In situation $x\xi$ the rule applier executes a rule $r \in \mathcal{S}(x\xi)$ with probability $P(r|\mathcal{S}(x\xi))$.

Since rules operate on strings and are restricted by context whose left part is a surface (i.e., transformed) string, this formulation is in principle capable of introducing far-reaching memory into the statistics. However, in practice it is necessary to have left-hand sides as well as context strings limited to two or three phonemes. As a result, in this formulation the actual execution of a fast speech rule at one point of the utterance does not increase the probability of execution of fast speech rules at following points. The same observation holds for dialectal and other rules. One possible improvement would be to introduce parameters that would inhibit or encourage the execution of a rule class over an entire utterance.

The limited memory in this formulation is, however, comparable to the influence length of the MAP dependencies. Since that device is modeled without memory, it was found possible to improve the overall *channel model* by adding to the phonetic rules a small set of artificial *front-end rules* written so as to reflect the performance of MAP, rather than that of the speaker. As an example, one front-end rule inhibits the production of successive fricatives because such inhibition is actually observed at MAP's output. The sometimes bad effect of front-end rules will be illustrated by the last example of Section VI.

We have now formally described the operation of the phonetic rule applier. The problem of estimating the probabilities of application of the various optional rules will be treated in Section VII. In Appendix V to their paper [9], Cohen and Mercer list the set of rules used by the IBM Speech Recognition Project. They comment on their phonetic significance in their Appendix I.

IV. An Acoustic Channel Model

In order to carry out linguistic decoding, it is convenient to combine the acoustic processor and speaker performance models into an integrated *acoustic channel model.* Formally, the latter will turn out to be a finite-state stochastic machine. That fact will enable us to estimate efficiently the values of the necessary statistical parameters.

The purpose of the speaker performance model is to provide a fast enumeration of all the possible surface forms of a hypothesized utterance, together with the generation probabilities. From the *linguistic decoder's* point of view, the base form and phonetic rule origins of these surface forms are irrelevant. It is, therefore, advantageous to prestore in a lexicon the collections of generatable word surface forms, and to form utterance hypotheses by their concatenations. The appropriate storage structure for such a collection is a directed graph whose arcs are labeled by phones. A probability distribution over outgoing arcs is attached to each node. Fig. 10 is an example of such a graph for the word *apprentice*. The graph is drawn so that every left-to-right path through it corresponds to a possible pronunciation of the word, which is given by the concatenation of phone labels associated with the arcs of the path. Of course, every surface form obtainable by application of phonetic rules to a base form of the word must correspond to some path in the directed graph.[14] The probabilities associated with the arcs leaving a node are directly calculable from the probabilities of the base form and of the phonetic rules that must be used to generate the surface forms whose element is the phone by which the arc is labeled. The probability of the pronunciation associated with a particular path is the product of probabilities on its branches (arcs).

It will be noticed that the arcs at both ends of the graph of Fig. 10 have conditions (such as ⟨*palatal consonants*⟩) associated with them. These terminal arcs are called *hooks* and are needed to take care of the influence of the neighboring word context on the pronunciation of the given word. Consider the right terminal conditions in Fig. 10. The first arc with the phone label "$\check{s}$" (pronounced "sh") will be connected to any initial arc of the succeeding word whose base form starts with a voiced obstruent type phoneme and whose arc condition is in turn satisfied by the phone "$\check{s}$." The second arc will be connected to any arc whose condition is satisfied by the phone "s." The third arc will be connected to any arc that has no attached conditions and belongs to a baseform starting with "s." Finally, the fourth arc will be connected to any arc whose condition is satisfied by "z" that belongs to a baseform starting with a voiced obstruent. Fig. 11(b) shows the result of connecting the phonetic graph of Fig. 10 to the phonetic graph for *sorcerer* given in Fig. 11(a) when initial and final silence is assumed. Of the four right-terminal arcs of *apprentice*, only the middle two were connected, both to the second left-terminal arc of *sorcerer*. The first left-terminal arc of the latter was annihilated because neither of "$\check{s}$," "s," or "z" is a voiced obstruent. The first and last right-terminal arcs of *apprentice* were annihilated because the base form of *sorcerer* does not start with a palatal consonant or a voiced obstruent.[15]

Having associated words with their phonetic (directed) graphs, it is now possible to combine the speaker and acoustic processor models into a finite-state *acoustic channel* model. We proceed as follows. Consider the finite state MAP model of Fig. 5. Denote it by $F(x)$ if it pertains to the input phone x. Clearly, the model $F(x_1, x_2)$ appropriate to the input phone pair x_1, x_2 is obtained from $F(x_1)$ and $F(x_2)$ by connecting the final state of $F(x_1)$ to the initial state of $F(x_2)$ by a null arc that produces no output (such as we used for the deletion transition $S_1 \rightarrow S_3$). This is diagrammed in Fig. 12, and the process completely generalizes, so that the machine for $F(x_1, x_2, \cdots, x_n)$ is obtained by interconnecting the machines $F(x_1), F(x_2), \cdots, F(x_n)$. Therefore, if the phonetic graph of a word consisted of a single surface form

[14] To construct such a graph automatically, using the base form and phonetic rule collections, is a rather complicated task that is outlined in Section IV of the paper by Cohen and Mercer. The problem is to generate a minimal structure in terms of, say, the number of arcs used.

[15] It is interesting to note that there are 31 104 distinct paths through the graph of Fig. 11(b). Thus a compact representation of surface form collections, such as a directed graph, is truly imperative.

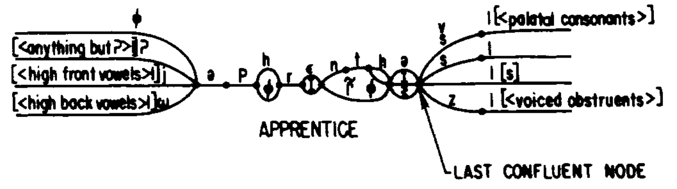

Fig. 10. A directed phonetic graph that lists the possible pronunciations of the word *apprentice*.

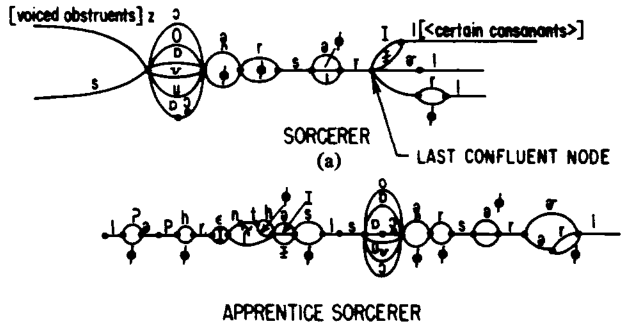

Fig. 11. (a) A phonetic graph for *sorcerer*. (b) The phonetic graph resulting from the hookup of graphs for *apprentice* and *sorcerer* when the phrase *apprentice sorcerer* is surrounded by silence.

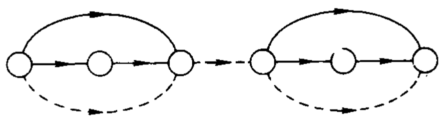

Fig. 12. The MAP model machine corresponding to a pair of consecutive inputs.

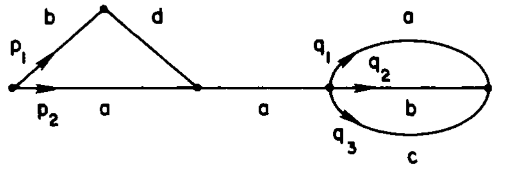

Fig. 13. A possible phonetic graph with node-leaving transitions labeled by their probabilities.

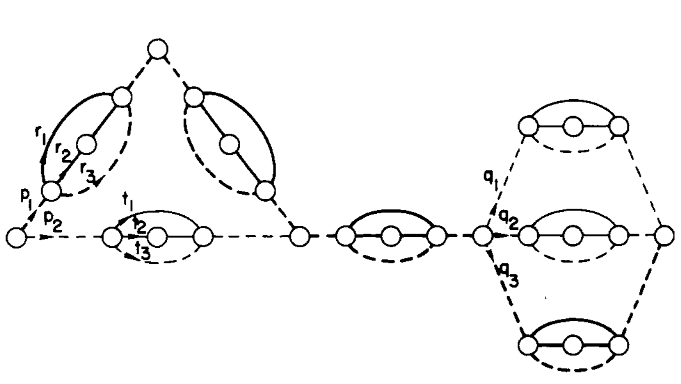

Fig. 14. The acoustic channel machine corresponding to the phonetic graph of Fig. 13.

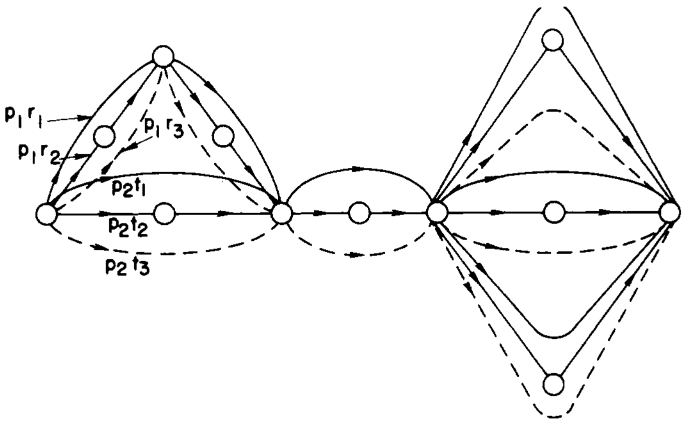

Fig. 15. The reduced acoustic channel machine equivalent to that of Fig. 14.

$x_1, x_2, \cdots, x_n$, the corresponding MAP output could be viewed as generated by the machine $F(x_1, x_2, \cdots, x_n)$. In reality, most words are represented by nondegenerate phonetic graphs (such as Fig. 10). Machines corresponding to such words are obtained by regarding nodes of the phonetic graphs as states and by replacing arcs labeled x by the machine $F(x)$. Thus the machine of Fig. 14 corresponds to the phonetic graph of Fig. 13. It is easy to check that to any path of the latter by phones $x_1, x_2, \cdots, x_n$ there corresponds a path through the former which in reality is the machine $F(x_1, x_2, \cdots, x_n)$. To illustrate this, we have used heavy lines to mark the (sub) machine in Fig. 14 corresponding to the path *bdac* of Fig. 13.

Fig. 14 can be further simplified by noting that there is no distinction between states internal to the machines $F(x)$ and those that used to be nodes in the directed graph. Furthermore, it is possible to eliminate all states that have only null incoming transitions by simply connecting their outgoing transitions to all their immediate predecessor states, and adjusting appropriately the probabilities of these transitions. The machine of Fig. 15 can be obtained in this way from that of Fig. 14.

We have now derived an integrated channel model for the acoustic processor and speaker performance models of the preceding two sections. To each word in the lexicon there corresponds a finite state machine of the same type as Fig. 15 that probabilistically generates the phones put out be the acoustic processor.

There are two other approaches to integrated acoustic channel models that we have found useful at IBM. However, in order to give a better overview of the total speech recognition process, we postpone their discussion to Section IX of this paper. For the same reason we also delay the description of model statistics extraction methods to Section VII. Nothing should prevent the interested reader from skipping directly to that material if he should be so inclined.

V. Linguistic Decoding

We now get to the problem of recognizing the spoken utterance. We will assume that we deal with sentences $w = w_1 w_2 \cdots w_n$ where w_i is the ith spoken word. The speaker reads some sentence w^* which is then processed by the acoustic processor, and the latter puts out an estimated phone sequence U. It is the task of the *linguistic decoder* (LD) to determine that sentence $\hat{w}$ which has most probably caused the observed sequence U, i.e., one which maximizes the *a posteri-*

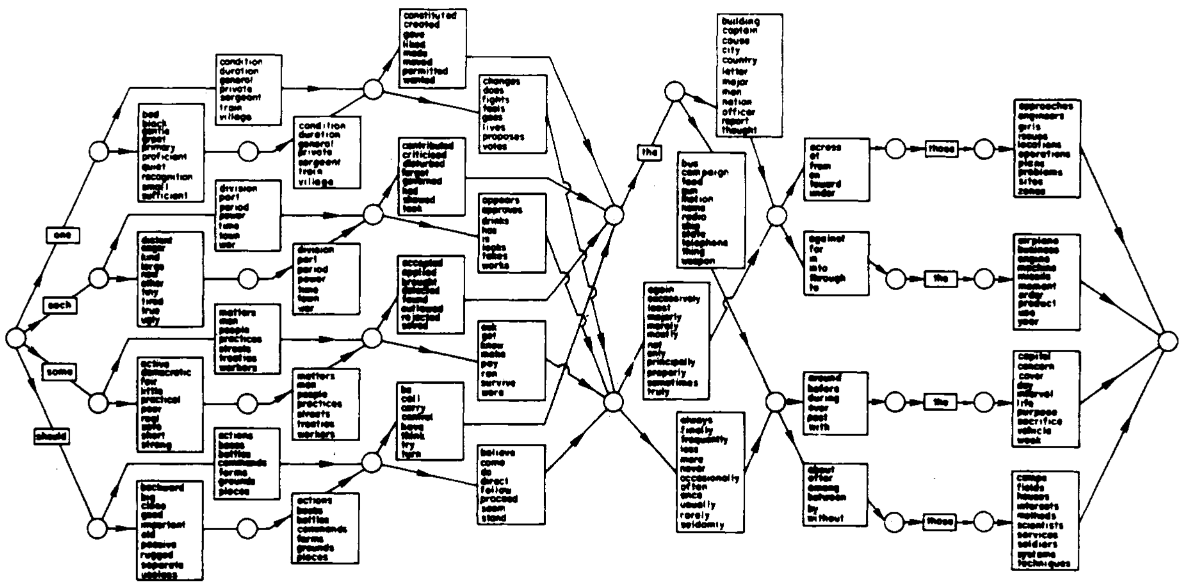
one
each
some
should
bad black gentle great primary proficient quiet recognition small sufficient
distant eager kind large real other tiny tired true ugly
active democratic few little practical poor real simple short strong
backward big close good important old passive rugged separate useless
condition duration general private sergeant train village
division part period power time town war
matters man people practices streets treaties workers
actions bases battles commands forms grounds pieces
constituted created gave liked made moved permitted wanted
contributed criticized disturbed forgot governed had showed took
accepted applied brought detected found outlawed rejected solved
be call carry control have think try turn
changes does fights feels goes lives proposes votes
appears approves drinks has is looks takes works
ask get know make pay run survive were
believe come do direct follow proceed seem stand
the
building captain cause city country letter major man nation officer report thought
bus campaign food gun motion name radio ship state telephone thing weapon
again successively least majorly merely mostly not only principally properly sometimes truly
always finally frequently less more never occasionally often once usually rarely seldomly
across of from on toward under
against for in into through to
around before during over past with
about after among between by without
those
the
the
those
approaches engineers girls issues locations operations plans problems sites zones
airplane business engine machine missile moment order product use year
capital concern cover day interval life purpose sacrifice vehicle week
camps fields houses interests methods scientists services soldiers systems techniques

Fig. 16. Model of the New Raleigh language.

ori probability $P\{w|U\}$.[16] Now by Bayes theorem,

$$P\{w|U\} = P\{U|w\}\,P\{w\}/P\{U\}. \tag{3}$$

In (3), the probability $P\{U|w\}$ reflects the channel model, and thus the interaction of the acoustic processor with the speaker, while $P\{w\}$ corresponds to the *a priori* probability of generation of the sentence w. The distribution $P\{U\}$ is really unimportant for decoding, since $P\{U\}$ remains fixed as the LD varies w trying to maximize (3).

It is the function of the *language model* (LM) to provide us with estimates of $P\{w\}$ for all word strings w.[17] We will assume that the LM is a Markov source in the sense of Appendix II, i.e., a Markov chain whose state transitions are associated with word output probability distributions. We will leave to later research the problem of LM construction for natural language corpora,[18] and will simply assume here that the LM is given.

An example of a very simple (and artificial) language model is given in Fig. 16 which diagrams the so-called *New Raleigh language*. The graph branches have state transition probabilities attached to them. The boxes on the transitions list the words that can be put out when that transition is taken.

The output probability is uniformly distributed over the list. The model is capable of generating such English-like sentences as:

Some man brought the weapon into the airplane.

and

Each town is often without those services.

and many more semantically deranged sentences such as

Each distant division appears always about those camps.

and

Should backward actions be the cause among those camps?

The total vocabulary used consists of 250 words.

Since (by design) every possible word string of the New Raleigh language corresponds to one state path only, the decoding task is a simple one: given the MAP output U, find the most likely path through the diagram of Fig. 16. It might seem that one could proceed as follows:

1. "Replace" each word in Fig. 16 by the acoustic machine F appropriate to it (the character of the latter depending on the channel model used), thus obtaining a new Markov source with transitions that produce either a single output or no output at all.
2. Using the Viterbi algorithm,[19] find the most likely state path through the source.
3. Determine the word string corresponding to the latter path.

While the above approach can indeed be carried out for the simple language model of Fig. 16 (see the next section), it will run into great difficulties for more complex (and, therefore, more English-like) models, since the number of Viterbi trellis states will be much too big. But even if the method were a practical one, it would not be optimal because the path found in step 2 would determine the word string as well as its pronunciation (since the search would find a specific phonetic arc sequence) which is a quantity we are not interested in.

[16] It might seem strange that we do not say that the task of the linguistic decoder is to find the sentence w^* actually spoken. But such a statement would lack operational precision. Obviously, in "true life" a listener weighs what he hears together with his expectation of what might have been said and then arrives at his estimate of the sentence uttered (or, perhaps, *intended to have been uttered*, since we often make allowances for speaker error).

[17] This statement may seem innocuous to information theorists, but is very controversial with linguists, many of whom believe that the task of a language model is to decide whether a word string is grammatical, and if so, perhaps to "interpret" its meaning. We will admit the possibility of a zero probability assignment to word strings that do not constitute a sentence, but will expect that probability values of other strings will vary.

[18] Some ideas dealing with that task are discussed in [16]. Our group at IBM is testing these on the so-called *Laser Patent Text* which is a 2.5-million word long collection of laser patent applications supplied to us by the US Patent Office. The vocabulary is about 12 000 words large.

[19] The Viterbi algorithm [11], [12] is an optimal search procedure for finding the most likely state sequence of a Markov source. It is described in Appendix II.

An effective procedure applicable to the present problem is the stack algorithm of sequential decoding [1], [2] that originated in the field of information transmission.[20] It carries out a left-to-right nonexhaustive search through the tree of word string hypotheses.[21] Left-to-right search makes sense, since the beginning of the sentence is presumed known, and so the variable length of words as well as phone deletions and insertions by the acoustic processor may be handled. The task of the search algorithm is to examine in some orderly manner initial partial paths of the tree, and direct the search along extensions of those paths that seem most promising. The search process utilizes a stack and depends on an evaluation function L that assigns values to the partial paths:

1. Initially, the sole stack entry is the empty path corresponding to the root node of the tree.
2. Take the partial path whose L-value is highest off the top of the stack. If the partial path is complete (leads to a terminal tree node), stop. Else L-evaluate all the one-branch extensions of the path, and insert the corresponding new extended partial paths into the stack so that the entries of the latter are arranged in order of decreasing L-value.
3. Repeat from 2.

It remains to specify the L function. Note that in assigning a value to a partial path leading from the root node to some intermediate node n^* we are really assigning a value to the set $\mathcal{S}(n^*)$ of complete paths that lead through node n^*.[22] It is thus natural to require L to be the likelihood that the "true" path[23] is one belonging to $\mathcal{S}(n^*)$. If we do so, then the second step of the stack decoding algorithm will partition the most likely set $\mathcal{S}(n^*)$ into subsets, thus helping to pinpoint more accurately the identity of the "true" path. The desired formula for L is then[24]

$$L(n^*) = \sum_{w \in \mathcal{S}(n^*)} P\{U|w\} P\{w\}. \tag{4}$$

Now the problem with (4) is that it renders the stack algorithm effectively useless since it requires the knowledge of $P\{U|w\}$. $P\{w\}$ for all w.[25] We wish therefore to approximate the above $L(n^*)$ by an easily computable formula. Let n^* be at tree depth k (measured in number of branches from the root node) and let

$$w = w_l^k w_{k+1}^m$$

where m is the total length of w. Let U be of length l, and denote by $P\{U_1^h, h|w_1^k\}$ the probability that the first k words of w resulted in exactly h acoustic processor outputs, U_1^h. Then (4) can be rewritten as

$$L(n^*) = P\{w_1^k\} \sum_{h=0}^{l} P\{U_1^h, h|w_1^k\} P\{U_{h+1}^l | U_1^h, w_1^k\} \tag{5}$$

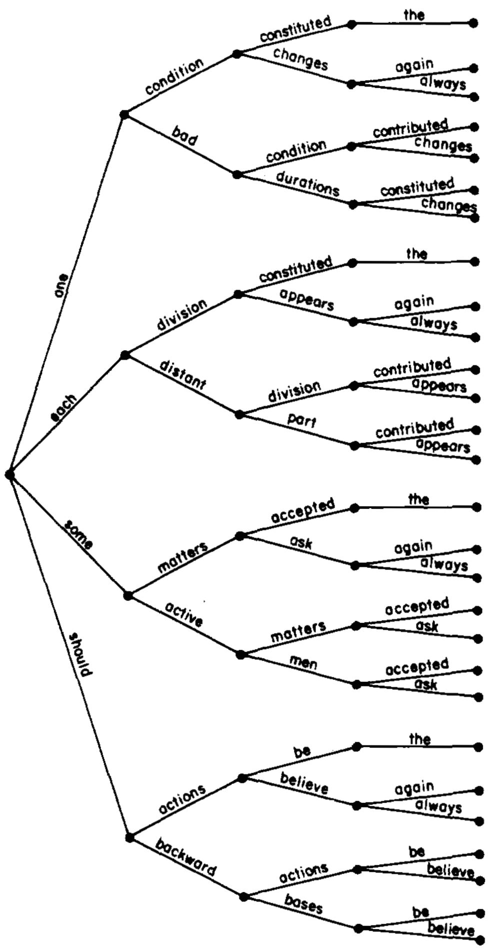

Fig. 17. The initial stages of a hypothesis search tree corresponding to a truncated version of the language of Fig. 16.

where

$$P\{U_{h+1}^l | U_1^h, w_1^k\} = \sum_{\mathcal{S}(n^*)} P\{U_{h+1}^l, w_{k+1}^m | U_1^h, w_1^k\} \tag{6}$$

is the probability that the remainder U_{h+1}^l of the output sequence is produced by some word string stemming from the node n^*. It is the probabilities (6) that must be approximated. It is possible to make a simple Markov model for the acoustic processor outputs and estimate its parameters by direct observation. Such a model will provide us with probabilities $P\{U_{h+1}^l | U_1^h\}$ that we can use in formula (5) instead of $P\{U_{h+1}^l | U_1^h, w_1^k\}$. Other, more exact approximations are also possible.

Since in step 2 of the algorithm the top partial path in the stack is replaced by a set of one-branch successor paths, it would be convenient to be able to calculate the L values for the latter from the L value of the former. To do so, one must be able to calculate

I. $P\{w_1^{k+1} = w_1^k w_{k+1}\}$ from $P\{w_1^k\}$

II. the set $P\{U_1^h, h|w_1^{k+1}\}$, $h = 0, 1, \cdots, l$
from the set $P\{U_1^h, h|w_1^k\}$, $h = 0, 1, \cdots, l$

III. the set[26] $\hat{P}\{U_{h+1}^l | U_1^h, w_1^{k+1}\}$, $h = 0, 1, \cdots, l-1$
from the set $\hat{P}\{U_{h+1}^l | U_1^h, w_1^k\}$, $h = 0, 1, \cdots, l-1$.

[20] Some people working in artificial intelligence call this the *best first scheme* [22].

[21] Fig. 17 diagrams the initial branching of a hypothesis tree corresponding to a modification of the language of Fig. 16 that results when at most two transitions out of each state exist.

[22] In Fig. 18 the terminal nodes of paths belonging to $\mathcal{S}(n^*)$ are indicated by a check.

[23] i.e. the path w^* having the highest $P\{U|w\}P\{w\}$ value.

[24] The ideas of the remainder of this section have appeared in [16].

[25] If we knew that much, we could find the maximizing path w^* directly.

[26] The caret symbol $\wedge$ indicates that an approximation is being used.

Assuming that the approximation $\hat{P}\{U_{h+1}^{l} | U_1^h, w_1^k\}$ is such as to allow the calculation III (in the Markov $P\{U_{h+1}^{l} | U_1^h\}$ case the problem does not even arise), only calculations I and II need be considered. The latter is going to be dealt with in Section VIII. As to the former, if the language model is such that w_1^k determines a unique LM state $S(w_1^k)$ (as in Fig. 16), then

$$P\{w_1^{k+1}\} = P\{w_{k+1} | S(w_1^k)\} P\{w_1^k\}. \tag{7}$$

Otherwise, if $\mathcal{O}(w_1^k)$ is the set of states that the LM can be in after the sequence w_1^k is put out, then

$$P\{w_1^{k+1}\} = \sum_{\mathcal{S} \in \mathcal{O}(w_1^k)} P\{w_{k+1} | \mathcal{S}\} P_k\{\mathcal{S}, w_1^k\} \tag{8}$$

where $P_k\{\mathcal{S}, w_1^k\}$ is the probability that w_1^k was put out and the LM ended up in state $\mathcal{S}$. Since

$$P_{k+1}\{\mathcal{S}', w_1^{k+1}\} = \sum_{\mathcal{S} \in \mathcal{O}(w_1^k)} q(w_{k+1} | \mathcal{S} \longrightarrow \mathcal{S}') \cdot P(\mathcal{S}' | \mathcal{S}) P_k\{\mathcal{S}, w_1^k\} \tag{9}$$

then for language models in which the output word string does not determine the LM state, the stack entry corresponding to w_1^k must store the set of probabilities $\{P_k\{\mathcal{S}, w_1^k\}, \mathcal{S} \in \mathcal{O}(w_1^k)\}$ whose elements are then used in (8). The set appropriate for stack entries corresponding to extensions of w_1^k is obtained by (9).

The above version of stack decoding may be impractical if the number of path extensions is too large. In that case L should be modified to apply to sets of partial paths as well as to a single partial path, and step 2 should be changed to[27]

2'. Take the top entry off the stack. If it corresponds to a single partial path, then do 2'a.

2'a. If the partial path is complete, stop. Else L-evaluate the *a priori* most probable one-branch extension of the path, and the set of remaining one branch extensions. Insert the resulting two entries into the stack so as to maintain the decreasing L-value stack arrangement. Else, if the top entry corresponds to a set of paths, then do 2'b.

2'b. L-evaluate the *a priori* most probable partial path from the set, and also the set consisting of the rest of partial paths (if any). Insert the resulting one or two entries into the stack to maintain its decreasing L-value arrangement.

[27] Since even in this version every stack entry corresponds to some set of complete paths $\mathcal{S}$, the appropriate L-function is given by

$$L(\mathcal{S}) = \sum_{\mathbf{w} \in \mathcal{S}} P\{U | \mathbf{w}\} P\{\mathbf{w}\}. \tag{*1}$$

The problem is how to approximate (*1). One way is to proceed as follows. The set $\mathcal{S}$ in (*1) is either the set of all paths leading through some node n^*, or a subset of such a set $\mathcal{S}$. In the former case, $\mathcal{S}$ is a subset of the set $\mathcal{S}'$ of all paths leading through the immediate predecessor node n^+ of the node n^*. We can thus write (*1) as

$$L(\mathcal{S}) = P\{U, \mathcal{S}\} = P\{U, \mathcal{S}, \mathcal{S}'\} = P\{\mathcal{S} | U, \mathcal{S}'\} P\{U, \mathcal{S}'\}$$

where the second equality is valid since $\mathcal{S}$ is a subset of $\mathcal{S}'$. Since $\mathcal{S}'$ is a set of all paths stemming from some node n^*, $P\{U, \mathcal{S}'\}$ is given by formula (5). If we then approximate $P\{\mathcal{S} | U, \mathcal{S}'\}$ by $P\{\mathcal{S} | \mathcal{S}'\}$, a possible L-function is

$$L(\mathcal{S}) = P\{w_1^k\} \sum_{h=0}^{l} P\{U_1^h, h | w_1^{k-1}\} \hat{P}\{U_{h+1} | U_1^h, w_1^{k-1}\} \tag{*2}$$

for sets $\mathcal{S}$ of all paths stemming from the node corresponding to w_1^k. For other sets $\mathcal{S}$ it is

$$L(\mathcal{S}) = P\{w_1^{k-1}\} \left[\sum_{\mathcal{C}} P\{w_k | w_1^{k-1}\} \right] \sum_{h=0}^{l} P\{U_1^h, h | w_1^{k-1}\} \cdot \hat{P}\{U_{h+1}^{l} | U_1^h, w_1^{k-1}\}. \tag{*3}$$

The set $\mathcal{C}$ in (*3) is such that the set of all paths that are continuations of $w_1^{k-1} w_k$, $w_k \in \mathcal{C}$ is identical with $\mathcal{S}$.

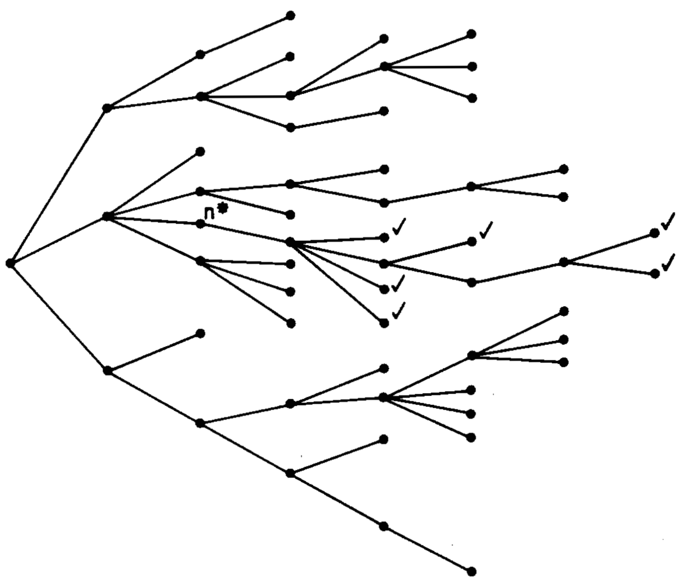

Fig. 18. A hypothesis tree corresponding to a possible language whose sentence length varies. Sentences belonging to the set $\mathcal{S}(n^*)$ have been marked by a check.

VI. Some Experimental Results

We will now review the outcomes of some speech recognition experiments performed on the system described in the preceding sections. Additional results[28] pertaining to the efficiency of our statistics extraction procedures are presented in the next section. The reader interested in performance comparisons with other design philosophies is directed to the accompanying paper by Reddy [23]. Our experiments were run under relatively favorable recording conditions: a "standard" American male speaker known to the automatic recognizer read his script into a high-fidelity microphone that was suspended in a sound-treated room.

The utterances of the first experiment were continuously spoken seven digit telephone numbers. The vocabulary consisted of eleven words: oh, zero, one, two, $\cdots$, nine. Table I[29] gives the results for (a) a stack linguistic decoder and (b) a Viterbi linguistic decoder. The acoustic processor was trained and designed on the basis of the utterance set used. In particular, the output alphabet was restricted to phonemes that actually occur in digit words.

Table II indicates the quality of the MAP. It lists the values of some of the statistical parameters pertaining to the model of Fig. 5, as extracted by the methods of Section VII.[30]

[28] Complete results of IBM recognition experiments run during the year 1974-1975 are given in [18].

[29] In this section, the *size of the experiment* refers to the number of utterances the system attempted to recognize. Unless otherwise indicated all results apply to *test data*, i.e., utterances previously unseen by the recognizer.

[30] It should be stressed that Tables II, III, V, VI, VIII, and IX necessarily reflect any inadequacies of the acoustic processor model and of the automatic extraction procedure that operates on the complete channel model. Performance statistics obtained by comparison of hand labeled data with MAP output can be found in [6]-[8], and [18].

TABLE I
DIGITS RECOGNITION EXPERIMENT PERFORMANCE (11-1-74)

	Size of Experiment	% Correct Utterances	% Correct Digits
Stack	100	89	98.3
Viterbi	100	82	97.4

TABLE II
AUTOMATICALLY EXTRACTED MAP MODEL PARAMETERS FOR TELEPHONE DIGIT EXPERIMENT

Class	Phone	Deletion	Insertion	Substitution	First Sub	Second Sub
Liquid-Glides	*RX*	0.01	0.05	0.94	0.97	
	WX	0.51	0.029	0.45	0.76	
Nasals	*NX*	0.02	0.02	0.959	0.955	
Voiceless Stops	*?X*	0.253	0.025	0.721	0.766	
	TX	0.017	0.0073	0.97	0.90	
	KX	0.008	0.156	0.835	0.86	
	XX	0.05	0.01	0.94	0.68	
Low-Energy Fricatives	*FX*	0.001	0.02	0.97	0.96	
	TH	0.0028	0.2	0.79	0.61(*TH*)	0.31(*FX*)
	VX	0.1	0.08	0.81	0.84	
High-Energy Fricatives	*SX*	0.001	0.024	0.974	0.96	
	ZX	0.0014	0.024	0.974	0.87	
Miscellaneous	*VF*	0.037	0.01	0.95	0.885	
	R$ (FLAP)	0.017	0.22	0.75	0.43(*NX*)	0.39(*?X*)
	QX (Aspiration)	0.0005	0.007	0.99	0.98(*IIQ*)	

The value entered in the first subcolumn gives the probability that the phone[31] in question was correctly recognized, given that a substitution took place. If a phone appears in parentheses, then the entry refers to the probability of that phone being put out. If another phone is put out with substantial probability, then a probability value is entered in the second subcolumn.

Another indication of MAP performance can be gleaned from Table III. The telephone number 937-5965 was spoken and resulted in the indicated MAP output phone string. Using the acoustic channel model, the Viterbi algorithm[32] was run to find the most likely path through the phonetic graph of the spoken number (known to the decoder), given the observed phone string. Table III aligns the path and output phone strings. Two output phones in a row indicate an insertion, and no phone indicates a deletion (word boundaries "|" are often deleted since they are not phonetically realized). The MAP model of Fig. 5 allows at most a single insertion. The alignment for the digit 5 indicates that this model limitation may be excessive: the speaker's *AX* phone most likely resulted in the triplet *AA AA AA*. Thus one use of the Viterbi algorithm is to check the adequacy of the model.

The decoding errors made in the telephone digits experiment were invariably limited to short digits. Thus *oh* was occasionally deleted, or *one* changed to *nine*. Contextual errors also took place. For instance, *six six seven* was once changed to

TABLE III
PHONETIC GRAPH-PROCESSOR OUTPUT VITERBI ALIGNMENT FOR TELEPHONE NUMBER 937-5965

Best Path Through Phonetic Graph		Aligned Processor Output String
|	. .	
NX	. .	*NX*
AX	. .	*AA*
IXG	. .	*IX*
NX	. .	*NX*
|	. .	
TH	. .	*IX TH*
RX	. .	*RX*
EE	. .	*EE*
|	. .	
SX	. .	*SX*
EH	. .	*EH*
VF	. .	*VF*
IXG	. .	*EH*
NX	. .	*NX*
|	. .	
FX	. .	*FX*
AX	. .	*AA AA*
IXG	. .	*AA IX*
VX	. .	*VX*
|	. .	
NX	. .	*NX*
AX	. .	*UX*
IXG	. .	
NX	. .	*NX*
|	. .	
SX	. .	*SX*
IX	. .	*IX*
KX	. .	*KX*
|	. .	
FX	. .	*FX*
AX	. .	*AA AA*
IXG	. .	
VX	. .	*IX*
|	. .	

six eight seven because of the indicated possible degemination: "siks(s)ik(s)sevən."

The second series of experiments involves utterances generated by the New Raleigh Language Model (Fig. 16) that was adjusted so that all words that could be generated by a transition out of a given state were equally likely. Table IV gives the over-all results for MAP followed by a stack decoder. The fourth column is a measure of performance of the stack algorithm in that it gives the percentage of cases in which the correct sentence had a greater likelihood than the decoded sentence.

The legend "training" indicates that the 100 sentences decoded were chosen from the 1070 sentences that were used to extract statistics for the acoustic channel model.

For this experiment, the design of the MAP processor parameters was based on an $8\frac{1}{2}$ minute segment of phonetically unconstrained general American speech unrelated to the New Raleigh language. The phonetic component of the channel model had a 133 phone output alphabet, and the acoustic processor component a 62 phone alphabet. Table V gives the statistical parameter values for the phone subset identical to the one presented in Table II. The seeming inferiority of the performance can be accounted for by the fact that MAP attempts to decide between 62 phones rather than between the 24 used in the digit experiment. Table VI shows the extracted model statistics for a vowel subset.

[31] The phonetic significance of the phone alphabet used can be found in Appendix I.
[32] See Appendix II.

TABLE IV
NEW RALEIGH EXPERIMENT PERFORMANCE (1-28-75)

	Size of Experiment	% Correct Sentences	% Correct Words	% Decoding Problems
Training	100	85.0	97.9	0.2
Test	363	81.0	97.3	0.073

TABLE V
A CONSONANT SUBSET OF AUTOMATICALLY EXTRACTED MAP MODEL PARAMETERS FOR THE NEW RALEIGH LANGUAGE EXPERIMENT

Class	Phone	Deletion	Insertion	Substitution	First Sub	Second Sub
Liquid-Glides	*RX*	0.21	0.03	0.71	0.42(*ER*)	0.28(*RX*)
	WX	0.12	0.18	0.7	0.6	
Nasals	*NX*	0.04	0.12	0.84	0.81	
Voiceless Stops	*?X*	0.45	0.02	0.53	0.24(*DX*)	0.19(*BX*)
	TX	0.16	0.04	0.8	0.37(*TX*)	0.16(*KX*)
	KX	0.073	0.056	0.87	0.29(*DX*)	0.22(*KX*)
	XX	0.37	0.1	0.53	0.79(*SB*)	0.1(*FX*)
Low-Energy Fricatives	*FX*	0.003	0.17	0.827	0.875(*FX*)	
	TH	0.01	0.29	0.70	0.49(*FX*)	0.34(*TH*)
	VX	0.12	0.19	0.69	0.37(*VX*)	0.25(*DX*)
High-Energy Fricatives	*SX*	0.0004	0.12	0.88	0.97	
	ZX	0.0003	0.19	0.81	0.68(*ZX*)	0.27(*SX*)
Miscellaneous	*VF*					
	R$	0.12	0.004	0.87	0.58(*DX*)	0.11(*NX*)
	QX					

TABLE VI
A VOWEL SUBSET OF AUTOMATICALLY EXTRACTED MAP MODEL PARAMETERS FOR THE NEW RALEIGH LANGUAGE EXPERIMENT

Class	Phone	Deletion	Insertion	Substitution	First Sub	Second Sub
Front Vowels	*EE*	0.0038	0.09	0.9	0.86	
	EH	0.0027	0.0616	0.935	0.94(*EH*)	
	EI	0.0029	0.147	0.849	0.858(*EI*)	0.1(*IX*)
	IX	0.002	0.025	0.97	0.967	
	AE	0.1	0.4	0.5	0.76(*EH*)	
	IXC	0.39	0.026	0.58	0.8(*EX*)	0.114(*IX*)
	EEG	0.24	0.05	0.7	0.5(*EI*)	0.24(*NX*)
	IXG	0.077	0.075	0.847	0.93(*IX*)	
Non-front Vowels	*AW*	0.1	0.4	0.5	0.76(*AA*)	
	AX	0.002	0.218	0.779	0.99(*AA*)	
	HX	0.026	0.23	0.745	0.54(*IU*)	0.32(*UU*)
	AA	0.08	0.28	0.63	0.88	
	*OU*1	0.0154	0.1	0.877	0.93(*OU*)	
	*OU*2	0.022	0.053	0.924	0.78(*OU*)	
	*UH*1	0.055	0.177	0.767	0.88(*UH*)	
	UHØ (*SHWA*)	0.4	0.029	0.56	0.64(*EH*)	0.26(*UH*)
	UU	0.0077	0.244	0.75	0.62(*UU*)	0.19(*IX*)
	UUG	0.2	0.067	0.73	0.69(*UU*)	
	UXG	0.548	0.39	0.059	0.4(*UU*)	0.3(*OU*)

Let us analyze further the recognition results summarized in Table IV. The errors were again limited to rather short words. Their list is presented in Table VII, where negative numbers indicate changes in the reverse direction. Thus *goes* was changed into *does* a total of four times, and *does* was changed into *goes* once. Comparison with the language model of Fig. 16 will show that only "within box" errors took place, so that no error in model state estimation ever occurred. It should be noted that most phonetically similar words were placed on transitions emanating from the same state.

TABLE VII
MOST FREQUENT WORD CONFUSIONS (NEW RALEIGH 1-28-75)

For ⟶ Through	3, -3	Do ⟶ Believe	2
Goes ⟶ Does	4, -1	Carry ⟶ Try	2
Good ⟶ Big	3, -1	Thing ⟶ Food	2
Think ⟶ Be	3	Kind ⟶ Tired	1, -1
To ⟶ Into	3	Year ⟶ Order	1, -1
Gave ⟶ Made	2, -1	At ⟶ Across	1, -1
Into ⟶ In	2	At ⟶ For	1, -1

It might seem puzzling at first why *do* was changed twice into *believe*. It will prove instructive to investigate this problem. The left section of Table VIII shows the Viterbi alignment of the MAP output string with the best path through the phonetic graph of the sentence *Should separate commands do least on those sites?* The right section of the table shows the alignment of the same output string when *believe* is substituted for *do* in the phonetic graph. We can see that the cause of the decoding error is the triple insertion of the first vowel in *least*. This allows the Viterbi algorithm to align the excessive output vowels *EE EE* with the second vowel of *believe*, thus compensating for the phone length difference between *do* and *believe*. This example shows the desirability of adding a prosodic component to MAP that would put out word boundary indicators inhibiting phone alignment shifts across boundaries.

The second instance of *believe* for *do* substitution is examined in Table IX. There the output string for *Should close places do sometimes toward those plans?* is aligned with the most likely path through the corresponding phonetic graph. We see that apparently a path exists allowing the surface form *BX BQ LX EE*1 *SX UH*1 *MX* · · · for *believe some(times)*. This is due to a dummy *front-end* phonetic rule[33] allowing the deletion of one of two successive fricatives. The rule was introduced to compensate for the lack of memory in the acoustic processor model and to reflect the fact that MAP design parameters are set so as to inhibit the outputting of fricative strings. Table IX thus exhibits one instance of backfiring of an otherwise beneficial practice of front end rule creation.

VII. ESTIMATION OF CHANNEL MODEL PARAMETERS

To use the channel model of Section IV, it is necessary to determine the values of its statistical parameters. Model parameter adjustment to fit a particular set of circumstances (such as a speaker or an acoustic processor) is often referred to as *training*. It is clear that any realistic training procedure must be almost completely automatic so as to require the very minimum of human intervention. Consider, for instance, the simple performance model of Fig. 5 pertaining to the acoustic processor of Section II. How should one determine the probability $q_x(y \mid S_1 \rightarrow S_3)$ of the event that MAP put out the phone y when phone x was spoken and resulted in a substitution? How can one even be sure that x and no other phone was pronounced within some specified time interval? Certainly not by taking the speaker's word for it! Perhaps one could use trained phoneticians to segment and label the speech sample. Ignoring the probable discrepancy in labeling by different phoneticians

[33] See the next to last paragraph of Section III.

TABLE VIII

Comparison of Alignment of the Spoken Sentence "*Should separate commands do least on those sites?*" with the wrongly decoded sentence "*Should separate commands believe least on those sites?*"

Best Path Through Phonetic Graph		Aligned Output String	Best Path Through Phonetic Graph		Aligned Output String
|	..				
XX	..	*SB*			
|	..				
SH	..	*SH*			
*UX*1	..	*IX*			
DX	..	*DX*			
|	..				
SX	..	*TQ SX*			
*EH*1	..	*EH WX*			
PX	..	*TX*			
PQ	..	*TQ*			
*ER*0	..	*ER*			
?X	..	*GX*			
|	..				
KX	..	*XX*			
KQ	..	*KQ*			
*UH*0	..	*EH*			
MX	..	*MX*			
*AE*1	..	*EH IX EH*			
NX	..	*NX*			
ZX	..	*ZX*			
|	..				
DX	..	*GX*	*BX*	..	*GX*
DQ	..	*PQ*	*BQ*	..	*PQ*
*UX*1	..	*IX*	*EE*0	..	*IX*
UUG	..		*LX*	..	*RX*
|	..		*EE*1	..	*EE EE*
LX	..	*RX*	*VX*	..	
*IX*1	..	*EE EE EH*	|	..	
EEG	..	*EE*	*LX*	..	*EH*
SX	..	*SX*	*EE*1	..	*EE*
TX	..	*KX*	*SX*	..	*SX*
TQ	..	*TQ*	*TX*	..	*KX*
|	..		*TQ*	..	*TQ*
*AA*0	..	*IX UH UX*			
|	..				
N$	..	*MX*			
*OU*1	..	*LX*			
|	..				
SX	..	*SX*			
*AA*1	..	*AA*			
IXG	..	*IX*			
TX	..	*DX*			
SX	..	*SX*			
|	..				
XX	..	*XX ?Q*			
|	..				

(a recent study [24] shows that their subjective judgments agree only about 51 percent of the time), one could never hire enough of them to generate as much data as would be necessary to estimate reliably the probability of a deletion, substitution, or insertion.

Our semiautomatic approach is based on the forward-backward (FB) extraction algorithm outlined in Appendix III. In order to apply the former we will introduce the concept of a lattice which is a generalization of the Viterbi trellis (see Appendix II). The lattice idea will be useful also in the next section dealing with efficient computation of channel transmission probabilities during linguistic decoding.

TABLE IX

Comparison of Alignment of the Spoken Sentence "*Should close places do sometimes toward those plans?*" with the wrongly decoded sentence "*Should close places believe sometimes toward those plans?*"

Best Path Through Phonetic Graph		Aligned Output String	Best Path Through Phonetic Graph		Aligned Output String
|	..				
XX	..	*TQ*			
|	..				
SH	..	*SH*			
*UX*1	..	*IX EH*			
DX	..	*DX*			
DQ	..	*ZH*			
|	..				
KX	..	*DX*			
KQ	..	*KQ*			
*UH*1	..	*UH*			
UXG	..	*OU*			
SX	..	*SX*			
|	..				
PX	..	*XX*			
PQ	..	*DH*			
LX	..	*EH*			
*EI*1	..	*EI*			
SX	..	*SX*			
*IX*0	..	*NX*			
ZX	..	*ZX*			
|	..				
DX	..	*GX*	*BX*	..	*GX*
DQ	..	*BQ*	*BQ*	..	*BQ*
*UU*1	..	*EE*	*LX*	..	
|	..		*EE*1	..	*EE*
SX	..	*SX*			
*UH*1	..	*UH*			
MX	..	*NX NX*			
TX	..	*DX*			
TQ	..	*TQ*			
*AA*2	..	*UH*			
EEG	..	*EE*			
MX	..	*RX*			
SX	..	*SX*			
|	..				
TX	..	*TX*			
TQ	..	*KQ*			
*UH*0	..				
WX	..	*WX*			
*AW*1	..	*OU*			
RX	..	*IX*			
DX	..	*GX*			
|	..				
BQ	..	*BQ*			
*UH*1	..	*UH*			
UXG	..	*LX*			
ZX	..	*ZX*			
|	..				
PX	..	*PX*			
PQ	..	*KQ*			
LX	..	*LX*			
*AE*1	..	*UH IX UX*			
NX	..	*DX*			
ZX	..	*SX*			
|	..				
XX	..				
|	..				

The training data will consist of a set $\mathfrak{U} = \{U_1, U_2, \cdots, U_k\}$ of utterances. U_i denotes the phone string put out by the acoustic processor of Fig. 4 when a word string $w_1^i w_2^i \cdots w_{l_i}^i$ was spoken into it. The machine corresponding to U_i is obtained by first hooking together (see Section IV) the phonetic graphs of the words $w_1^i, w_2^i, \cdots, w_{l_i}^i$, thus obtaining a graph G_i, and then replacing the phone arcs of G_i by the phone machines of Figs. 5 or 6 (the replacement process is the one used to transform Fig. 13 into Fig. 14). The resulting machine F_i will have two kinds of states: *word states* that correspond to the rules of graph G_i, and *phone states* that correspond to the phone machines. During any iteration of the extraction process, the states of F_i will be distinguished in the FB algorithm phase. However, states having the same word or phone origin will be lumped together for purposes of statistics extraction.

The machines F_i have many *null* transitions (those producing no output) that cannot be eliminated by some simple expedient. As a consequence it will be necessary to somewhat modify the trellis creation procedure as well as the FB algorithm.

A *lattice* (just like a trellis) unfolds in time a finite state machine transition process in that it places in evidence the totality of state paths that could have given rise to an observed output sequence $x_1^n = x_1, x_2, \cdots, x_n$. The lattice is a diagram consisting of $n + 1$ columns (i.e., one more than the number of observed outputs). The $(t + 1)$th column contains all the machine states that could have been reached as a result of generating the tth output x_t. The transitions between the states of the tth and $(t + 1)$th column are those that could have produced x_t. All null transitions out of any state of any column will lead to the appropriate states of the *same* column, as determined by the underlying machine (no self-transitions exist in our machines). As an example, Fig. 20 shows the lattice diagram connections corresponding to the machine of Fig. 19. The actual lattice for a two phone output sequence would be given by the first three columns of Fig. 20.

The FB algorithm is a method that computes recursively the model state transition probabilities for the observed output string x_1^n. Since transitions between states of the same lattice column exist, we must modify the recursions (A.10) and (A.11) of the FB algorithm (see Appendix III). If s_t denotes a state of the tth lattice column $(t = 2, 3, \cdots, n + 1)$ then the new forward recursion is

$$P\{s_t = S, x_1^t\} = \sum_{S'} P\{s_{t-1} = S', x_1^{t-1}\} P\{s_t = S, x_t | s_{t-1} = S'\} + \sum_{S''} P\{s_t = S'', x_1^t\} P\{s_t = S | s_t = S''\} \quad (10)$$

where the second term is a sum over all null transitions into S. The fact that the right-hand side of (10) depends on probabilities $P\{s_t = S'', x_1^t\}$ means that the computations of the left-hand sides cannot be carried out in an arbitrary order. It can be shown that if it is impossible to return to any state by following null transitions only (as is the case in all channel models) then there is a state ordering $S^{i_1}, S^{i_2}, \cdots$ such that the computation of $P\{s_t = S^{i_j}, x_1^t\}$ involves only the probabilities $P\{s_t = S^{i_k}, x_1^t\}$, for $k = 1, 2, \cdots, j - 1$.[34]

[34] This allows the probabilities (10) to be computed in the natural order $P\{s_t = S^{i_1}, x_1^t\}, P\{s_t = S^{i_2}, x_1^t\}, \cdots$, each computation involving only previously calculated quantities.

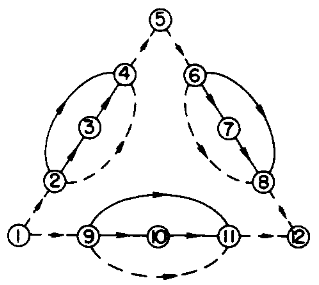
Fig. 19. A possible acoustic channel model.

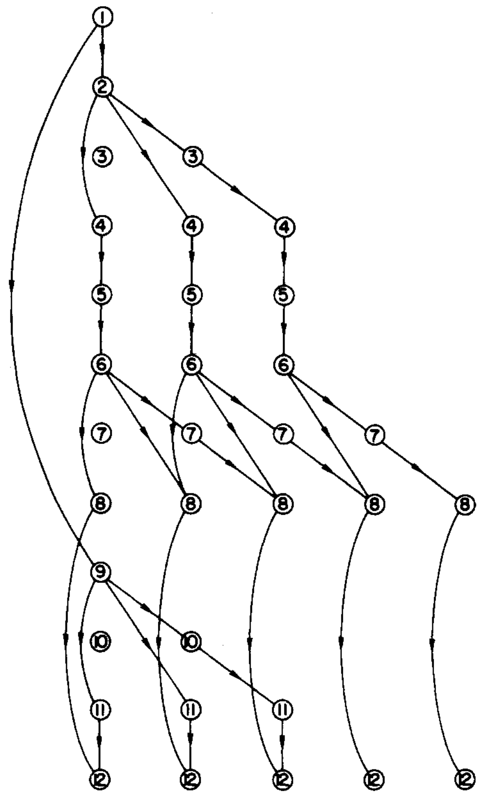
Fig. 20. Lattice diagram for the machine of Fig. 19.

The new backward lattice recursion is

$$P\{x_t^n | s_{t-1} = S\} = \sum_{S'} P\{x_{t+1}^n | s_t = S'\} P\{s_t = S', x_t | s_{t-1} = S\} + \sum_{S''} P\{x_t^n | s_{t-1} = S''\} P\{s_{t-1} = S'' | s_{t-1} = S\} \quad (11)$$

and again an appropriate state ordering exists allowing the second summation of (11) to be carried out. The probabilities of output producing transitions are given by

$$P\{s_{t-1} = S, s_t = S', x_1^n\} = P\{s_{t-1} = S, x_1^{t-1}\} \cdot P\{s_t = S', x_t | s_{t-1} = S\} P\{x_{t+1}^n | s_t = S'\} \quad (12)$$

as before (see (A.9)), and the null transition probabilities by

$$P\{s_t = S, s_t = S', x_1^n\} = P\{s_t = S, x_1^t\} = P\{s_t = S' | s_t = S\} P\{x_{t+1}^n | s_t = S'\}. \quad (13)$$

To simplify matters somewhat, let us assume that each word has only a single base form.[35] Then transitions out of word states correspond to the application of different phonetic rules. Each node of G_i (and thus each word state) will be labeled to indicate the set of rules $\mathfrak{R}$ that generated its outgoing branches. As a result, two nodes with the same label will have an identical outgoing branch topology (but not vice versa). Similarly, each phone state of F_i will be labelled to designate the phone to which it corresponds and its order within the phone machine (thus if the machines of Fig. 5 are used, there will be three different labels for each phone x corresponding to the three machine states). Denote the labeling function by H, word state labels by A, A', etc., and phone state labels by B, B', etc. Then a state S labeled by A satisfies the relationship

$$H(S) = A. \tag{14}$$

Let the probability of taking the rth arc out of a word node labeled A be $P\{r|A\}$. Let the transition probability from phone state B to state B' be $P(B'|B)$ [the value is 0 unless B and B' belong to the same phone], and let the probability of output phone x resulting from an output producing transition $B \to B'$ be $q(x|B \to B')$.

The modified iterated FB extraction process appropriate to the channel model of Section IV is then as follows.

I. An initial guess is made as to the starting probabilities $\hat{P}_0\{r|A\}$, $\hat{P}_0(B'|B)$, and $\hat{q}_0(x|B \to B')$. A training utterance data set $\mathfrak{U} = \{U_1, U_2, \cdots, U_k\}$ is generated.

II. At the beginning of the lth iteration step the FB algorithm is applied to the utterance of $\mathfrak{U}$, based on the machines $\{F_1, F_2, \cdots, F_k\}$. If a word state S of F_i is labeled A, then the rth outgoing transition is assigned the probability $\hat{P}_0\{r|A\}$. If phone states S and S' of F_i are labeled B and B', respectively, then the transition between them is governed by $\hat{P}_0(B'|B)$, and their output generation by $\hat{q}_0(x|B \to B')$.

III. Let $P(s_{t-1} = S, s_t = S', U_i)$ be the probability assigned by the FB algorithm to an output producing transition $S \to S'$ at lattice level t when U_i is processed. Then define the quantities[36]

$$N(B, B', u) \triangleq \sum_{i=1}^{K} \sum_{t=1}^{n_i} \delta(u_{it}, u) \sum_{S,S'} \delta(H(S), B)\, \delta(H(S'), B') \cdot P\{s_{t-1} = S, s_t = S', U_i\} \tag{15}$$

and

$$N(B, B') \triangleq \sum_{u} N(B, B', u); N(B) = \sum_{B'} N(B, B'). \tag{16}$$

The output and transition probability model estimates for the $(l + 1)$th iteration step become

$$\hat{q}_l(u|B \longrightarrow B') = \frac{N(B, B', u)}{N(B, B')}$$

$$\hat{P}_l(B'|B) = \frac{N(B, B')}{N(B)}. \tag{17}$$

IV. Let $P\{s_t = S, s_t = S', U_i\}$ be the probability assigned to a null transition $S \to S'$ of lattice column t when U is processed. Then define

$$N_\phi(C, C') \triangleq \sum_{i=1}^{k} \sum_{t=1}^{n_i} \sum_{S,S'} \delta(H(S), C)\, \delta(H(S'), C') \cdot P\{s_t = S, s_t = S', U_i\} \tag{18}$$

where C and C' indicate either word or phone labels, and the subscript ϕ indicates lack of output. $N_\phi(A, A')$ or $N_\phi(A, B')$ are transition "counts" out of word states. The null transition probabilities for the $(l + 1)$th iteration step then become

$$\hat{P}_l^\phi(B'|B) = N_\phi(B, B') \Big/ \sum_{B''} N_\phi(B, B'') \tag{19}$$

and the rth-arc probability out of a word node labeled A becomes

$$\hat{P}_l\{r|A\} = N_\phi(A, C') \Big/ \sum_{C''} N(A, C'') \tag{20}$$

where C' is the terminal node of the arc.

In order to prevent excessive reliance on possibly insufficient data, the parameter values extracted by the above procedure are "smoothed" by formula (A.8) before they are used in the channel model for speech recognition. Because of our labeling strategy, the number of parameters to be determined is not excessive. Below, we will give some data on the amount of training needed for convergence. It is clear that the process is fully automatic once the initial distributions $\hat{P}_0(B'|B)$, $\hat{P}_0\{r|A\}$, and $\hat{q}_0(x|B \to B')$ have been determined.

At IBM, we have approached the last task by a combination of expert advice and hand labeling. The distribution $\hat{P}_0\{r|A\}$ was selected to be uniform over all rules. The remaining probabilities were obtained through comparison of hand labeled spectrographic data with corresponding acoustic processor output. Since the amount of hand labeled data was insufficient to provide enough information about individual phones, the latter were classified into categories (such as stops, front vowels, high-energy fricatives, etc.) and relative occurrence frequencies were computed for the latter. The phone probabilities were then made uniform for phones belonging to the same category.[37] There was a vast difference between the initial guesses and the final extracted values. Also, a big improvement was noted in recognition when based on the latter values (see below).

The above formulation seemingly separates speaker and acoustic processor performance characteristics. It would be tempting to conjecture that once the model is trained, then a change of speaker would involve redetermination of only the $P\{r|A\}$ probabilities, leaving the $q(x|B \to B')$ and $P\{B'|B\}$ distributions intact. This, however, is almost surely not the case, since the interaction of these parameters is a very complex one. In fact, the memoryless modeling of the acoustic processor will transfer some of its dependent characteristics onto the values of the extracted phonetic parameters. Thus when changing the speaker or the acoustic processor, the most one can hope for is that the old parameter values will serve well as a starting point for a new extraction process.

[35] The generalization to multiple baseforms is straightforward, and is left to the reader.

[36] In (15) $\delta(\ ,\ )$ denotes the Kronecker delta function. Thus for the tth lattice level the last sum is over all transitions from any node labeled B to any node labeled B'.

[37] Thus, e.g., insertion probabilities would depend on the phone category and not on the phone identity, or, given substitution, the probabilities of correct identification of phones "n" and "m" were made equal as were the probabilities of substituting "s" for "n" and "z" for "m."

We have carried out experiments whose results are an indication of the efficiency of the approach presented in this section. Table X shows the variation in decoding performance of the stack recognizer of the New Raleigh language (see Section VI) as a function of the number of training iterations used in determining the channel parameters. All the results given are for a training set of 800 sentences[38] and a test set of 100 sentences. The 0th iteration entry gives the performance based on the best human estimation of the parameters. Using the latter as the initial values $\hat{q}_0$ and $\hat{P}_0$, the first iteration of the extraction procedures is seen to result in a huge improvement in sentence recognition from 20 to 79 percent. Of the 21 errors made after the first iteration, 3 were due to decoding problems. That is, the decoder failed to find the sentence which maximized the probability of the observed phonetic sequence according to the model which was being used. Thus these 3 errors were not necessarily due to the model parameters. After another iteration, the performance improved slightly (to 83 percent correct). A third iteration brought no further improvement, and in fact, an extra decoding problem occurred. The "average stack" column is a measure of decoder search. It gives the average number of likelihood calculations needed to decode a sentence.

It is, of course, desirable to extract statistics from the smallest amount of data that will do the job. Table XI gives the quality of the performance with varying training set size when the system attempted to recognize 100 test and 100 training sentences. Notice that with 200 training sentences performance is lower on the "test" set and is higher on the "training" set than it is with larger training sets. This divergence of the test and training results indicates that the model is being trained to special features of the training set which do not occur in the same way in the test set. There is always a danger with small training sets that special features will occur which are a coincidence rather than a characteristic of the phenomena being studied. The problem would be even worse for a smaller training set. In this experiment, 600 sentences seems to be an adequate training size.

From our statistical point of view, the speaker–acoustic-processor combination is a channel whose function is to provide sufficient mutual information about the spoken text to allow transcription by the linguistic decoder. It is then in some sense beside the point how well the phone-based acoustic processor recognizes phones. We have taken the acoustic processor as designed for male speaker A,[39] and used it to recognize utterances by male speakers B and C when the channel statistical parameters were trained for each of them. Table XII gives the somewhat disappointing results that are indicative of the unfortunate (and well-known) variability of phone production by speakers, and of the sensitivity of acoustic processor performance relative to its parameter values.

TABLE X
PERFORMANCE WITH VARIABLE NUMBER OF ITERATIONS OF AUTOMATIC TRAINING

Iteration	% Correct Sentences	Decoding Problems	Average Stack
0	20	8/10 st. overfl.	
1	79	3	226
2	83	2	200
3	82	3	191

TABLE XI
EXPERIMENT WITH VARIABLE TRAINING SAMPLE SIZE

TEST

Size	% Correct Sentences	Decoding Problems	Average Stack
0	20	8/10 st. overfls.	
200	77	3	253
400	80	2	191
400	85	1	186
800	82	3	191
1070	83	3	192

TRAINING

Size	% Correct Sentences	Decoding Problems	Average Stack
200	88	3	143
400	87	2	147
600	84	3	151
800	84	2	150
1070	86	2	156

TABLE XII
MULTIPLE SPEAKER EXPERIMENT

	Size	% Correct Sentences	Decoding Problems
Speaker B			
Training	100	68	7
Test	100	49	11
Speaker C			
Training	100	66	8
Test	100	60	6

VIII. Computation of Channel Transmission Probabilities

In this section, we would like to complete the discussion of linguistic decoding and show how to compute conveniently the probabilities $P\{U_1^h, h \mid w_1^k\}$ needed for the evaluation of the likelihood function (5). Obviously, $P\{U_1^h, h \mid w_1^k\}$ depends on the acoustic channel model used, and our approach will be valid as long as the output string U can be thought of as being generated by a stochastic machine. The general problem was treated in [17].

Our method is based on the lattice that corresponds to the stochastic machine in the manner described at the beginning of Section VII. Let us first assume that the machine for w_1^k has a single initial and a single terminal state. Then we can draw a lattice for it in such a way that the states in the ith row correspond to the ith state S^i, $i = 1, 2, \cdots, N$, where we assume that the state numbering is such that S^1 is the initial and S^N the final state, and the computation of $P\{s_t = S^i, x_1^t\}$ by formula (10) does not involve the probabilities $P\{s_t = S^j, x_1^t\}$, $j = i, i+1, \cdots, N$.[40] This numbering arrangement is followed

[38] In our previous terminology, the training data are $\mathfrak{U} = \{U_1, U_2, \cdots, U_{800}\}$ where U_i are sentences.

[39] That means that the processor parameters, such as phone prototypes and recognition thresholds, were adjusted to maximize correct phone recognition for speaker A.

[40] As pointed out in Section VII, such numbering is possible whenever the machine contains no loop of null transitions.

in the lattice of Fig. 20 that corresponds to the machine of Fig. 19. That lattice has a finite number of columns since the machine has no transition loops. A lattice corresponding to a machine with loops would have an infinite number of columns. An example is shown in Fig. 22 which corresponds to the machine of Fig. 21.

For each state S^i of lattice column t we can compute the "forward" probability $P\{s_t = S^i, U_1^t\}$ appropriate to the forward-backward algorithm. As pointed out in Section VII, this can be done recursively by formula (10). Because $P\{U_1^h, h \mid w_1^k\}$ is the probability that w_1^k resulted in exactly h outputs U_1^h, and because output generation must end in the terminal state S^N, we get the relationship

$$P\{U_1^h, h \mid w_1^k\} = P\{s_h = S^N, U_1^h\}. \tag{21}$$

Our remaining task is to show how to compute the probabilities $P\{U_1^h, h \mid w_1^{k+1}\}, h = 0, 1, \cdots, l$ when the probabilities $P\{U_1^h, h \mid w_1^k\}$ are known and $w_1^{k+1} = w_1^k w_{k+1}$ for some word w_{k+1}. The machine for w_1^{k+1} is obtained by connecting the single terminal node of the machine of w_1^k to the single initial node of the machine of w_{k+1} by a null transition. Let us number the states of the resulting machine from 1 to $N + M$, where M is the number of states pertaining to w_{k+1}. Then from (10),

$$\begin{aligned} P\{s_t = S^{N+i}, U_1^t\} = &\sum_{j=0}^{M} P\{s_{t-1} = S^{N+j}, U_1^{t-1}\} \\ &\cdot P\{s_t = S^{N+i}, U_t \mid s_{t-1} = S^{N+j}\} \\ &+ \sum_{j=0}^{i-1} P\{s_t = S^{N+j}, U_1^t\} \\ &\cdot P\{s_t = S^{N+i} \mid s_t = S^{N+j}\}. \end{aligned} \tag{22}$$

Both sums in (22) contain only nonnegative values of j because all paths from states of w_1^k to states of w_{k+1} lead through the terminal state S^N of w_1^k. Furthermore, the upper limit of the second sum is $i - 1$, since appropriate state numbering is possible because of our assumption of nonexistence of any loop of null transitions.

It follows from (22) that probabilities $P\{s_t = S^{N+i}, U_1^t\}$ can be obtained recursively column-by-column and by row within a column, if "boundary" values of the probabilities (21) are known. We can thus conclude the discussion of the stack decoding algorithm by stating that each stack entry must carry in it the values of

I. $\{P_k\{S, w_1^k\}, S \in \mathcal{O}(w_1^k)\}$ (see formula (9))

II. $\{P\{U_1^h, h \mid w_1^k\}, h = 0, 1, \cdots, l\}$ (see (21) and (22)).

III. $\{\hat{P}\{U_{h+1} \mid U_1^h, w_1^k\}, h = 0, 1, \cdots, l\}$ (this may not be necessary if this approximation is independent of w_1^k).

It is useful to define a *generalized lattice* corresponding to a stochastic machine as one having an infinite number of columns, each of which contains all machine states. The transitions between states of the same column correspond to null machine transitions, and those between states of neighboring columns correspond to the remaining transitions. The generalized lattice of the machine of Fig. 21 is then given in Fig. 23. Note that if the state in the first row and column of the generalized lattice is designated as the unique starting state (its probability is made equal to 1 in the recursion (10)) then the regular lattice results (compare Figs. 23 and 22).

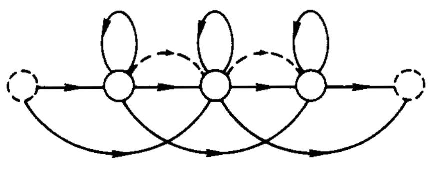
Fig. 21. A stochastic machine capable of producing an infinitely long output string.

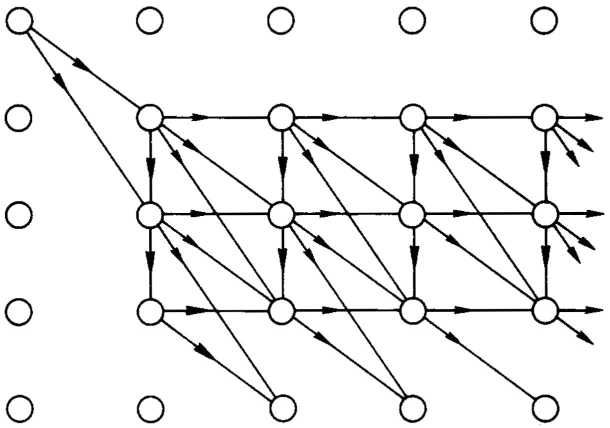
Fig. 22. The lattice corresponding to the machine of Fig. 21.

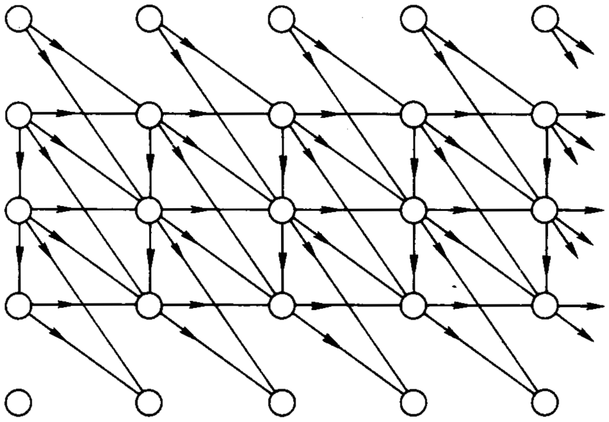
Fig. 23. The generalized lattice corresponding to the machine of Fig. 21.

It is easy to see that the regular lattice for w_1^{k+1} is obtained by attaching the generalized lattice of w_{k+1} to the regular lattice for w_1^k. This is done by creating probability-one transitions from the states in the last row of the latter lattice to those in the first row and corresponding column of the former lattice. Fig. 24 shows the result when w_1^k is represented by the machine of Fig. 19 and w_{k+1} by that of Fig. 21.

A lattice interpretation of the recursion (22) is now possible. Equation (22) says that the probability of the lattice state for w_1^{k+1} in row $N + i$ and column t is determined by probabilities of all states from which a transition into the $(N + i, t)$ state exists.

It is clear that the above lattice extension method is invalid when the machines in question do not have single initial and final nodes. The problem is not a serious one as long as word machines are independent of preceding and suceeding word context (we will leave its solution to the reader). But if context is important, then it becomes impossible even to compute $P\{U_1^h, h \mid w_1^k\}$ without knowing the identity of w_{k+1}. In

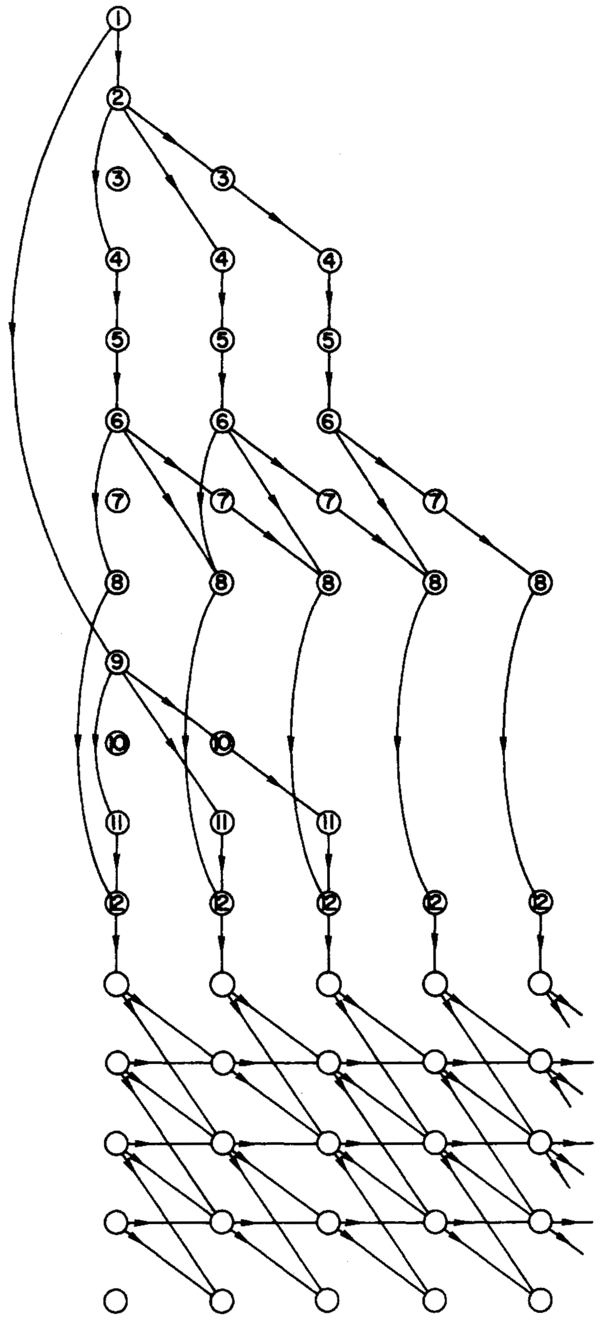

Fig. 24. The regular lattice corresponding to the concatenation of the machines of Figs. 19 and 21. The lattice results from extending the lattice of Fig. 20 by the generalized lattice of Fig. 23.

practice we have been able to circumvent the problem by use of the concept of *last confluent state* that is defined as the right-most state of the machine through which all transition paths must lead regardless of the succeeding word context. As an example, the last confluent states of the machines for the words APPRENTICE and SORCERER correspond to the last confluent phonetic graph nodes marked in Figs. 10 and 11(a). The lattice which is used in computing $P\{U_1^h, h \mid w_1^k\}$ then corresponds to a truncated w_1^k machine from which all states to the right of the last confluent one have been removed. The lattice for w_1^{k+1} is obtained from the lattice for w_1^k by attaching to the latter the generalized lattice for $(w_1^k)w_{k+1}$. The generalized $(w_1^k)w_{k+1}$ lattice results when we take the machine for w_1^{k+1}, and remove from it all the states to the left of the last confluent node of w_1^k and to the right of the last confluent node of w_1^{k+1}. The practical success of this procedure is due to the fact that hooks of real phonetic graphs are relatively short so that the last confluent machine state is very close to the right-hand machine boundary.

IX. Word-Based Acoustic Channel Models

Having described one basic version of a speech recognition system, including appropriate procedures for training its parameters, we will discuss two alternative acoustic channel models based on a simple acoustic processor. The latter has the enormous advantage of being a straightforward device requiring no elaborate data flows, decision algorithms, or parameter adjustments (as is, unfortunately, the case with the MAP processor [6]-[8] of Section II). Only semiautomatic statistics extraction procedures (see next section) similar to those of Section VII need be used to adjust the channel models to a new speaker or recording environment.[41] The acoustic processors involved can be run in conjunction with a linguistic decoder that differs only in a minor way from the one of Section V.

The acoustic channel model (see Fig. 15) derived in Section IV was obtained by integrating into one whole the speaker performance model of Section III and the MAP model of Section II. As a result, a word building block structure was created that agreed with our choice of words as outputs of the text generator.[42] Since our aim is the decoding of word strings and not of their pronunciation, the attempt by MAP to recognize phones, while possibly computationally efficient, is to some extent information destructive. Would it then be feasible to perform word recognition by constructing finite-state machines analogous to those of Fig. 15 directly for each word in the text vocabulary?[43] An attempt to do just that was made by Bakis [3] who achieved remarkable success with a relatively rudimentary finite state word model structure that is unmodified by any *a priori* phonetic information about the word in question.

Assume that the speaker, even though he may be talking continuously, produces words whose acoustic shape is independent of surrounding word context. Then the acoustic channel (formed by the speaker-processor combination) is specified if for each possible word we state the distribution of the corresponding acoustic output symbol strings.

The Bakis acoustic processor can be thought of as a discrete spectrograph which at time $t_i = t_{i-1} + \Delta$ produces an output vector $V_i = (r_1^i, r_2^i, \cdots, r_k^i)$ where

$$r_j^i = \int_{-\infty}^{t_i} a(t)\, \varphi_j(t_i - t)\, dt. \tag{23}$$

In (23), $a(t)$ denotes the speaker's log power spectrum[44]

[41] One might, however wish to change the shape of filters φ_j introduced in (23) below.

[42] In that formulation various inflections of the same stem (e.g., *transcribe, transcribes, transcribing, transcription, transcriptions*, etc.) constitute different symbols. Clearly, it would be possible to use syllables as units, or word stems and endings. In the latter case, one might wish to break up the text generator into a source followed by a fixed encoder, the latter transforming, say, the symbols ⟨brief⟩ and ⟨ity⟩ into the channel input pair ⟨brev⟩, ⟨ity⟩ or into the single input ⟨brevity⟩. Such an approach would, of course, have its own difficulties.

[43] This could alleviate the consequences of the fact that phone pronunciation depends on surrounding phonetic context, which presents great difficulties for phone recognizer design.

[44] The log power spectrum can be thought of as the logarithm of the magnitude of a short term Fourier transform of the speaker's acoustic waveform as registered, say, at the output of a microphone. In practice additional normalizing operations are carried out which need not concern us here.

and the "filters" $\varphi_1, \varphi_2, \cdots, \varphi_k$ are assumed appropriately chosen.[45]

To each word w there corresponds a finite state machine whose form is given in Fig. 25. There are $l + 2$ states (the value of l depends on the word w), the first and last of which are special in that they produce no outputs.[46] There are three transitions leaving each but the last regular state. A return transition, a transition to the next state, and a skip transition to the state after. Whenever a transition into a regular state s_h takes place, an output vector V is produced, whose value is assumed to be a gaussian random vector with a mean m_h and a covariance matrix Σ_h. The Bakis channel model is, therefore, specified by determining for each word w in the vocabulary: (i) the number of regular states, (ii) the transition probabilities between the states, and (iii) the characteristic output means m_h and variances Σ_h for $h = 1, 2, \cdots, l$.[47] We will show in Section X how to extract the statistical parameter values for the Bakis model.

It is clear that the stack algorithm of linguistic decoding is even simpler to implement for this channel model than for the one of Section IV. The data used in the likelihood function evaluation is the output sequence $V_1, V_2, \cdots, V_n$. The finite state machines for consecutive hypothesized words do not have hooks and are therefore directly interconnected as shown in Fig. 26. The method of channel transmission probability evaluation presented in Section VIII applies directly, the complication of nonunique initial and/or terminal machine states not even arising.

Experiments with the performance of a Speech Recognizer based on this model have been very encouraging. Table XIII compares its error rate with that achieved in conjunction with the MAP processor for the Telephone Digits experiment described in Section VI.

One major drawback of the Bakis method is that the large number of model states (necessitated by the filter output sampling increment time $\Delta = 0.01$ s) requires too much decoder processing time. The other is the necessity of separate statistics extraction for different words. Bakis tried to build up his word machines through a concatenation of separately trained syllable-like submachine units, the syllables used being obtained from the word base form. Preliminary results indicate an 85 percent correct sentence recognition rate for the New Raleigh language (compare with the 81 percent result reported in Table IV for the MAP processor).

The presence of return and skip states in the Bakis model provides for a random variation in the output string length, thus accommodating change in the speed of talk. However, in some other respects the model is quite rigid. First, it does not enable contextual adjustments at word boundaries (e.g., the "*d*" in *hand applause* is usually pronounced, but much less so in *hand clapping*). Second, even though the Gaussian nature of the state output vectors V allows varying pronunciations, the centering around m and the unidirectional flow of state transitions favor a standard articulation of words and do not

[45] The analog notation and terminology is not intended to preclude a digital approach. Only the latter, in fact, is being used at IBM. Bakis himself uses five functions φ_i, of which the first four are the eigenfunctions of the principal components of Dutch vowels [4], and the fifth is selected so that r_5^i is a quantity derived from log "instantaneous" energy at time t_i.

[46] Here states and *not* transitions produce outputs. This difference from previous formulation is trivial.

[47] This *ad-hoc* channel model is intended to facilitate the work of the linguistic decoder. Its structure and output vector generation characteristics are chosen in view of their relative simplicity. No claims whatever are made as to its power in modeling the real variability of actual speech production.

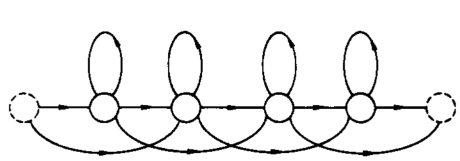
Fig. 25. The structure of the finite-state word machine for the Bakis acoustic channel model.

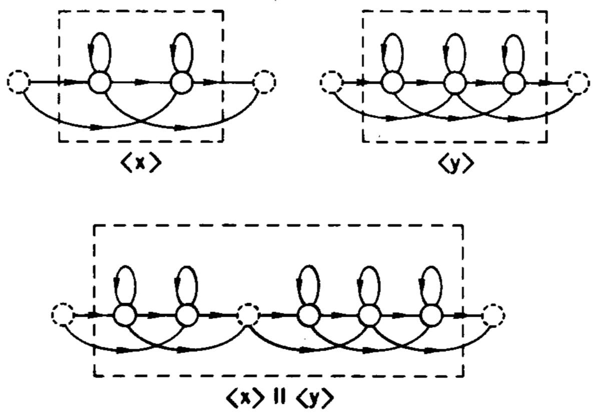

Fig. 26. Example of phone and/or transition unit concatenations.

TABLE XIII
DIGITS RECOGNITION EXPERIMENT PERFORMANCE (11-1-74)

	Size of Experiment	% Correct Utterances	% Correct Digits
MAP	100	89	98.3
Word-Based	054	94	99.2

provide explicitly for, say, fundamental differences as between "təmad.o" and "təmäd.o" (*tomato*) or fast speech variations as between "gəvərmənt" and "gəvmənt" (*government*). One possible solution under consideration is to provide more states and a more flexible transition pattern between them. Another approach being experimented with by Tappert [10] is to predesign the machine transition structure in accordance with the phonetic graphs of Section III. We will present below a somewhat modified version of this method that does not adequately face-up to the processing time problem.

For each phone we will have a *phone unit* which will account for the central portion, or steady-state part, of the spectrum produced by the phone. For each pair of consecutive phones we will have a *transition unit* which will account for the transition linking the steady states of the two phones. We need only have transition units for phone pairs that actually occur in speech.

Both kinds of units are finite state machines of the form of Fig. 25. As before, there is a probability distribution over the transitions out of each state, and when a state S_h is reached, an output vector V [see(10)] is produced with a Gaussian probability distribution of mean m_h and covariance matrix Σ_h.

Thus what characterizes every transition or phone unit is the number of its states, the transition probabilities between them, and their respective means m_h and variances Σ_h. The inventory of transition and phone units is used together with the phonetic graph lexicon to build up finite-state word models.

Let $\langle ab \rangle$ refer to the transition unit corresponding to the phone pair a, b, and let $\langle a \rangle$ refer to the phone unit corresponding to the phone a. Let the concatenation $\langle x \rangle \| \langle y \rangle$ refer to the

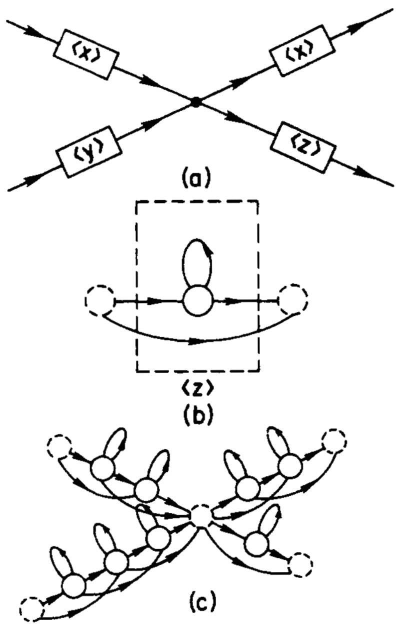

Fig. 27. Replacement of phonetic arcs by corresponding machine units.

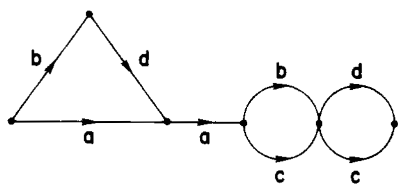

Fig. 28. A simple phonetic graph.

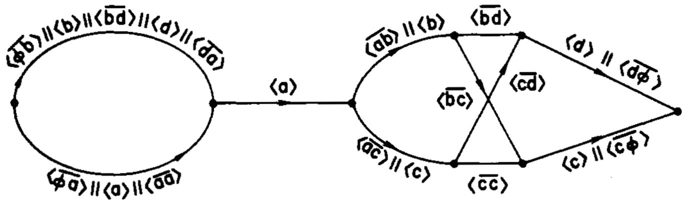
⟨$\overline{\phi b}$⟩ || ⟨b⟩ || ⟨$\overline{bd}$⟩ || ⟨d⟩ || ⟨$\overline{da}$⟩
⟨$\overline{\phi a}$⟩ || ⟨a⟩ || ⟨$\overline{aa}$⟩
⟨a⟩
⟨$\overline{ab}$⟩ || ⟨b⟩
⟨$\overline{bd}$⟩
⟨$\overline{bc}$⟩
⟨$\overline{cd}$⟩
⟨$\overline{ac}$⟩ || ⟨c⟩
⟨$\overline{cc}$⟩
⟨d⟩ || ⟨$\overline{d\phi}$⟩
⟨c⟩ || ⟨$\overline{c\phi}$⟩

Fig. 29. The graph of Fig. 28 modified by inclusion of transition units.

machine that results if we identify the last state of the machine $\langle x \rangle$ with the first state of $\langle y \rangle$ (we recall that both of these states are special in that they produce no outputs). The concatenation process is diagrammed in Fig. 26. To establish a pictorial notation, consider the labeled box graph of Fig. 27(a). If unit $\langle z \rangle$ has the structure of Fig. 27(b) and units $\langle x \rangle$ and $\langle y \rangle$ are as in Fig. 26, then the operation of "replacing the labeled boxes by corresponding units" when performed on Fig. 27(a) results in the machine of Fig. 27(c).

If Φ denotes the silence phone, then the machine for the word w that corresponds to a single phone string $x_1 x_2 \cdots x_n$ is simply given by the unit concatenation

$$\langle \Phi x_1 \rangle \| \langle x_1 \rangle \| \langle x_1 x_2 \rangle \| \langle x_2 \rangle \cdots \langle x_{n-1} \rangle \| \langle x_{n-1} x_n \rangle \| \langle x_n \rangle \| \langle x_n \Phi \rangle. \tag{24}$$

The machine corresponding to a word represented by a phonetic graph must be such that any path through the latter will correspond to a state sequence built up by unit concatenation (24). It is best to proceed by an example. Consider the phonetic graph of Fig. 28. If no transition units existed, one would simply replace the arcs by the appropriate phone units. In the case at hand, however, one must first transform the graph so its paths include all the different transitions that can take place (as well as phones) and so that the number of arcs is as small as possible. The different phone strings lying on the various paths of the transformed graph should be exactly those obtainable from the original graph. Fig. 29 constitutes the desired transformation of Fig. 28. The machine corresponding to any given word is obtained by replacing the labeled boxes on the arcs of the transformed graph by the corresponding transition and phone units (in the manner of Fig. 27).

We will show in the next section how to determine the statistical parameters that define the phone transition units. Initial recognition experiments carried out with this model have been encouraging, but it will be necessary to speed up decoder processing considerably before this method becomes practical.

X. Estimation of Word-Based Channel Model Parameters

In this section we show how the Viterbi and Forward-Backward extraction algorithms (Appendices II and III) can be used to estimate parameter values for the word-based channel models of the preceding section. The discussion pertaining to the Bakis model, although following the notation and approach of Section VII, is quite independent of it.

Let us recall that in the Bakis channel model, to each word in the dictionary there corresponds a stochastic machine (Markov source) having the overall structure of Fig. 25. The machine is determined by (i) the number l of regular output producing states, (ii) the transition probabilities between the states, and (iii) the regular state output mean vectors $\boldsymbol{m}_h$ and covariance matrices Σ_h for $h = 1, 2, \cdots, l$.

To determine l for a word w, Bakis inspects examples of a few (say L) spectral output vector sequences $V = V_{t+1}, V_{t+2}, \cdots, V_{t+n}$ (where $V_i = (r_1^i, r_1^i, \cdots, r_k^i)$ and r_j^i is obtained as indicated in (23)) that correspond to different spoken instances of w. (This initial step requires experienced and careful judgment.) Eliminating some obviously deviant examples, he finds the sequence V^* with the largest values of n and sets l equal to the latter.

To begin the statistics extraction process, it is necessary to make an initial guess at the parameter values. In fact, Bakis chooses $\boldsymbol{m}_h = V_{t+h}^*$ for $h = 1, 2, \cdots, l$ and assigns the value $\frac{1}{2}$ to the probability of each forward transition, and the value $\frac{1}{4}$ to the probability of each skip and return transition. The covariance matrices $\Sigma_h = \Sigma$ are restricted to be diagonal with initial values set in an experimentally determined, uniform, and word-independent way.

Let $\mathcal{V} = \{V_1, V_2, \cdots, V_L\}$ be the set of vector sequences considered. Using these as data, and the machine initialized as specified above, a preliminary, modified Viterbi extraction[48] is carried out until successively extracted parameter sets become essentially invariant:

I. If $N(S^h, S^{h+j})$ is the number of times the transitions $S^h \rightarrow S^{h+j}$ take place in the decoded state sequences $\mathcal{S} = \{S_1, S_2, \cdots, S_L\}$ then the initial transition probabilities for the

[48] Viterbi decoding (see Appendix II) of the sequence V_i is performed on a trellis whose transition levels correspond to the successive output vectors $V_{i1}, V_{i2}, \cdots, V_{in_i}$. Since the last machine state is nonoutput producing, to keep the formulation consistent, one may augment every V_i sequence by a special dummy empty vector.

next iteration will be

$$P(S^{h+j}|S^h) = N(S^h, S^{h+j}) \Big/ \sum_{i=0}^{2} N(S^h, S^{h+i}). \tag{25}$$

II. Let $Y_1, Y_2, \cdots, Y_M$ be the set of elements of sequences of $\mho$ that correspond to some transition into the state S^h of the machine (as determined by the Viterbi algorithm).[49] Then the next initial value of m_h will be given by the formula

$$m_h = \frac{1}{M}\sum_{i=1}^{M} Y_i \tag{26}$$

and that of σ_{hg}^2, the gth diagonal element of Σ_h, by

$$\sigma_{hg}^2 = \left[\sum_{i=1}^{M}(r_g^i)^2 - \frac{1}{M}\left(\sum_{i=1}^{M} r_g^i\right)^2\right] \Big/ (M-1) \tag{27}$$

where $Y_i = (r_1^i, r_2^i, \cdots, r_k^i)$ and r_j^i are filter outputs specified in (23).

The preliminary parameter values obtained on the last iteration of the preceding process are used as the initial guesses in the ultimate large-scale training procedure, except that the variances are further smoothed by the formula

$$\hat{\sigma}_h^2 = \frac{k}{k+M}\eta_g^2 + \frac{M}{k+M}\sigma_{hg}^2 \tag{28}$$

where η_g^2 is the gth diagonal element of the initial covariance matrix Σ.

The above preliminary process is first completed for all words in the vocabulary. The data for the ultimate extraction is a relatively large set of complete utterances $\mathfrak{U} = \{U_1, U_2, \cdots, U_k\}$ and the training is carried out for all words simultaneously. Let utterance U_i correspond to the word string $w_1 w_2 \cdots w_{l_i}$. Then the machine used to Viterbi decode U_i will consist of the series connection of the machines for the individual words $w_1, w_2, \cdots, w_{l_i}$.[50] The final extraction procedure is identical to the preliminary one described above, provided that states of different word machines are given different labels and are thus distinguished. It is necessary to maintain separate counters for each possible state transition pair S, S' of any word, so that values $N(S, S')$ can be stored. Furthermore, for each state it will be necessary to accumulate the vector sums ΣY_i and ΣY_i^2 (where by definition $Y_i^2 = ((r_1^i)^2, (r_2^i)^2, \cdots, (r_k^i)^2)$). The value of M (used in II above) for the state S' is obtained by the formula

$$M = \sum_S N(S, S'). \tag{29}$$

It should be noted that the entire training procedure involves a minimum of human intervention: that of selecting the preliminary word data sets $\mho$. The final extraction is automatic since the utterances U_i presumably arose from the reading of a script consisting of word strings $w_1 w_2 \cdots w_{l_i}$. The statistical parameters assigned to the word machines are those obtained from the last iteration of the extraction process. No smoothing is performed.

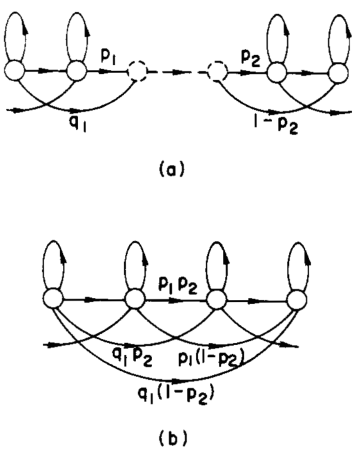

Fig. 30. Elimination of null transitions.

The last extraction problem concerns Tappert's modular word-based model. We wish to determine the identity and statistics of the phone and transition units, as well as the transition probabilities for the phonetic graphs (see Section III) underlying the acoustic channel. The approach suggested will parallel but not copy that taken by Tappert in his experiments.

Our plan is to first train the phone units and to add transition units later. We recall that the phone units are machines having the structure of Fig. 25. As in the Bakis case, for each phone x we must determine (i) the number of states l, (ii) the transition probabilities, and (iii) the state means m_h and covariance matrices Σ_h. We will start the training by letting the quantities (i) and (ii) depend on the phone x but not on the machine state. To determine the initial values, we hand label enough speech to gain several samples of each phone used. Suppose $V_1, V_2, \cdots, V_L$ are spectral time sample (STS vector) strings that are all hand labeled by the phone "a," where $V_i = V_{i1}, V_{i2}, \cdots, V_{in_i}$ and $V_{ij} = (r_1^{ij}, r_2^{ij}, \cdots, r_k^{ij})$.[51]

Then the initial values of l and of the $\langle a \rangle$ unit transition probabilities are chosen so as to make the expected number and variance of outputs generated by the phone unit[52] approxi-

[49] Thus $Y_i = V_{jh}$ for some $j \in \{1, 2, \cdots, L\}$ and $h \in \{1, 2, \cdots, n_j\}$, where the jth sequence of the set $\mho$ is denoted by $V_j = V_{j1}, V_{j2}, \cdots, V_{jn_j}$.

[50] In order to be able to create the usual trellis whose columns correspond to successive output vectors of U_i, it is best to adjust the resulting machine so as to eliminate intermediate states which produce no output. What is necessary is illustrated in Fig. 30 whose two machines are exactly equivalent.

[51] The r_m^{ij} quantities are sampled filter outputs, as in (23).

[52] If each return transition has probability α, each forward transition probability β, each skip transition probability γ, and if the number of regular states is l, then the following recursion holds for the average number of outputs $M(l)$ generated,

$$M(l) = \frac{1}{1-\alpha} + \beta M(l-1) + \gamma M(l-2) \tag{1*}$$

where

$$M(1) = \frac{a+b}{1-\alpha}, \quad M(2) = \frac{1}{1-\alpha}(1 + (\alpha+\beta)\beta).$$

The corresponding recurrence for the variance, Var (l), is

$$\text{Var}\,(l) = \frac{\alpha}{1-\alpha} + \beta\,\text{Var}\,(l-1) + \gamma\,\text{Var}\,(l-2) \tag{2*}$$

where

$$\text{Var}\,(1) = (\alpha+\beta)\frac{\alpha}{1-\alpha}, \quad \text{Var}\,(2) = (1 + (\alpha+\beta)\beta)\frac{\alpha}{1-\alpha}.$$

A good arbitrary choice is $\beta = 4\gamma$ and then (1*) and (2*) become a two variable system since necessarily $\alpha = 1 - 5\gamma$.

mately equal to the corresponding observed quantities

$$\bar{n}(a) = \frac{1}{L}\sum_{i=1}^{L} n_i \tag{30}$$

$$\sigma^2(a) = \frac{1}{L-1}\left(\sum_{i=1}^{L} n_i^2 - L\bar{n}(a)^2\right). \tag{31}$$

Choosing the middle three components of each V_i, we create a composite prototype and equate it to the mean value of the Gaussian distribution

$$m = \sum_{i=1}^{L}\sum_{j=1}^{3} V_{ig_i+j} \tag{32}$$

for all $h = 1, 2, \cdots, l$. The uniform covariance matrix Σ is chosen diagonal, and its elements are given by the vector

$$D = \frac{1}{3L-1}\left(\sum_{i=1}^{L}\sum_{j=1}^{3} (V_{ig_i+j})^2 - 3L(m)^2\right). \tag{33}$$

When the above initialization is completed for all units, their training starts. It is almost identical to that for the phone-based channel model as discussed in Section VII, the only difference being that Gaussian means and variances are being extracted rather than output probabilities $q(x|B \rightarrow B')$. We will, therefore, not describe it in detail.

The training data consists of a set $\mathfrak{U} = \{U_1, \cdots, U_k\}$ of utterances, where $U_i = U_{i1}, U_{i2}, \cdots, U_{in_i}$ is an STS string put out by the acoustic processor when the word string $w_1^i w_2^i \cdots w_{n_i}^i$ was spoken. Hooking together of phonetic graphs corresponding to the words w_j^i results in a graph G_i, and replacement of the arcs of the latter by the appropriate phone units results in the machine F_i that models the source of U_i. The states of F_i are again labeled, word states by A, A' etc., and phone states by B, B', etc. A lattice based FB extraction (Appendix III) is then performed. Word as well as phone state probabilities are extracted and iterated upon as in Section VII. The iteration of phone state mean and variance values is as follows.

Let

$$P\{s_t = S', U_i\} = \sum_{S} P\{s_{t-1} = S, s_t = S', U_i\}$$

be the probability that the output producing state S' was reached at lattice level t when utterance U_i took place. Then

$$m(B) = \frac{1}{N}\sum_{i=1}^{k}\sum_{t=1}^{n_i} U_{it}\sum_{S}\delta(H(S'), B)\,P\{s_t = S', U_i\} \tag{34}$$

will be the mean vector assigned in the next iteration to all states labeled B, where H is the labeling function (14) and

$$\bar{N} = \sum_{i=1}^{k}\sum_{t=1}^{n}\sum_{S'}\delta(H(S'), B)\,P\{s_t = S', U_i\}. \tag{35}$$

Similarly, the vector whose elements will become the diagonal entries of the covariance matrix Σ will be

$$D(B) = \frac{1}{N-1}\left[\sum_{i=1}^{k}\sum_{t=1}^{n} U_{it}^2\sum_{S'}\delta(H(S'), B) \cdot P\{s_t = S', U_i\} - \bar{N}m(B)^2\right]. \tag{36}$$

When the phone unit training is completed, transition unit creation is begun. The variances pertaining to phone unit end states are compared with an experimentally chosen threshold. To each phone unit x there will correspond a triplet of integers $(\lambda(x), \mu(x), \nu(x))$ whose sum equals the unit length $l(x)$. $\lambda(x)$ and $\nu(x)$ are equal to the number of left and right unit states, respectively, whose variances exceed the chosen threshold. The transition unit $\langle \overline{xy} \rangle$ will then have $\nu(x) + \lambda(y)$ regular states. The initial statistical parameters for the first $\nu(x)$ (last $\lambda(y)$) states will be those extracted for the last $\nu(x)$ (first $\lambda(y)$) states of the phone unit $\langle x \rangle$ ($\langle y \rangle$). After the initial transition units for all x, y combinations are established, new phone units $\langle x \rangle$ are obtained from old ones by eliminating the first $\lambda(x)$ and last $\nu(x)$ states of the latter. Machines F_i for training utterances U_i that use both phone and transition units are then created in the manner described in the preceding section. Appropriate state labeling is performed[53] and FB extraction carried out using formulas (18)–(20), (34)–(36). It is expected that the final estimates of word state transition probabilities $P\{r|A\}$ will be quite different from those gained for the phone based channel model.

Appendix I
Alphaphonetic Symbol Set

XX	silence		
PX	*p*an	*+H*	be *h*ere
BX	*b*an	*-H*	*h*ow
TX	*t*an	*PQ*	aspiration
DX	*D*an	*BQ*	aspiration
KX	*c*an	*TQ*	aspiration
GX	*g*ab	*DQ*	aspiration
MX	*m*an	*KQ*	aspiration
NX	*N*an	*GQ*	aspiration
NG	ri*ng*	*EE*	f*ee*t
FX	*f*an	*IX*	f*i*t
VX	*v*an	*EI*	p*a*ce
TH	ba*th*	*EH*	f*e*d
DH	*th*an	*AE*	f*a*d
SX	*s*ip	*ER*	b*ir*d
ZX	*z*ip	*UH*	b*u*t
SH	*sh*ip	*AA*	c*o*d
ZH	mea*s*ure	*AW*	b*ough*t
WX	*w*in	*OU*	b*oa*st
JX	*y*es	*UX*	b*oo*k
RX	*r*ip	*UU*	b*oo*t

Appendix II
The Viterbi Algorithm

In this and the next appendix we will describe certain algorithms that are useful in linguistic decoding and in automatic extraction of statistical model parameters. The algorithms are not new (the Viterbi algorithm has been used extensively) but their inclusion allows us to establish a convenient uniform notation and point of view.

Consider a finite state stationary Markov source characterized by a transition matrix $P(S'|S)$ and an output probability function $q(x|S \rightarrow S')$, $x \in A$ (A is the alphabet of outputs that can be put out when any transition $S \rightarrow S'$ takes place). As the source changes state, outputs are generated (according to the

[53] It will be necessary to distinguish word states, transition unit states, and phone unit states (see the discussion of the fourth paragraph of Section VII).

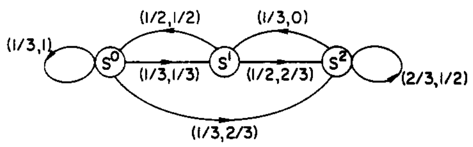

Fig. 31. A binary Markov source.

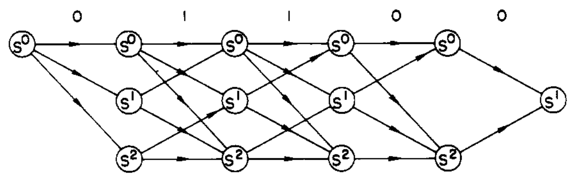

Fig. 32. The Viterbi trellis for the output string 01100 generated by the Markov source of Fig. 31.

q-distribution), one for each transition. We take the point of view that the outputs are accessible to observation, but not the states. To simplify matters somewhat, let the process always start from the state S^I and terminate in the state S^F[54] (which may well be the same one). The problem is to use the observed generated sequence $x = x_1, x_2, \cdots, x_n$ to estimate the state sequence $s = s_0 s_1 \cdots s_n$ (we know that $s_0 = S^I$ and $s_n = S^F$). The maximum *a posteriori* probability estimate of s is best obtained by the so-called Viterbi algorithm [11], [12] which is simply the dynamic programming solution to the problem.[55] As an example, consider the binary Markov Source of Fig. 31. Let the starting and terminal states be S^0 and S^1, respectively. To the arcs of Fig. 31 are attached number pairs. The first pair element denotes the transition probability, and the second denotes the probability that a 0 is generated. Thus, for instance,

$$P(S^1 | S^0) = 1/3 \qquad P(S^2 | S^1) = 1/2$$
$$q(1 | S^0 \longrightarrow S^0) = 0 \qquad q(0 | S^1 \longrightarrow S^2) = 2/3. \quad \text{(A.1)}$$

Let us assume that the source generated the sequence 01100, and that we want to know the most likely state path that "caused" it. The *trellis* diagram of Fig. 32 unfolds the process in time in that the former consists of the totality of paths (from the leftmost to the rightmost state of Fig. 32) that could have resulted in the observed output sequence. Note that each column lists all the states of Fig. 31, and that the transitions between the ith and $(i + 1)$th columns represent the possible ith transitions of the source. The first and last trellis columns contain only the starting and terminal states, respectively, since these are the only ones the source could have occupied at the corresponding times (0 and 5). State S^1 is missing from the next to last column since no transition $S^1 \rightarrow S^1$ exists and so the source could not have been in state S^1 at time 4 if it ended up in S^1 at time 5.

Let x denote the source output sequence, and let s denote a state sequence that could have caused x. The general task is to find that sequence s^* that would maximize the conditional probability $P(s | x)$. But since

$$P(s | x) = \frac{P(x, s)}{P(x)}$$

[54] This leads to no loss of generality. One may always add a dummy extra starting state with null transitions into the remaining states. Similarly, at the end of output generation, the source may be forced into a dummy terminal state.

[55] As we warned in the Section I, assertions of this and similar kind will not be proven here. Instead, all of our explanations will be heuristic ones.

and x is given, we might as well find s^* that maximizes

$$P(x, s) = \prod_{i=1}^{n} P(s_i | s_{i-1})\, q(x_i | s_{i-1} \longrightarrow s_i). \quad \text{(A.2)}$$

Let $k \in \{1, 2, \cdots, n - 1\}$ be fixed, and let $S(i)$ denote the sequence $s_0, s_1, \cdots, s_k = S^i, s_{k+1}, \cdots, s_n$ whose kth state was S^i, where $S^0, S^1, \cdots, S^M$ are the different states of the source. Then

$$\max_s P(x, s) = \max_{0 \leq i \leq M} \max_{S(i)} P(x, S(i)) \quad \text{(A.3)}$$

where the second maximization is over all state sequences whose kth state is S^i. Now

$$P(x, S(i)) = \left\{ \left[\prod_{j=0}^{k-1} P(s_j | s_{j-1})\, q(x_j | s_{j-1} \longrightarrow s_j) \right] \cdot P(s_k = S^i | s_{k-1})\, q(x_k | s_{k-1} \longrightarrow s_k = S^i) \right\}$$
$$\cdot \left\{ P(s_{k+1} | s_k = S^i)\, q(x_{k+1} | s_k = S^i \longrightarrow s_{k+1}) \cdot \prod_{j=k+2}^{n} P(s_{j-1} | s_{j-1})\, q(x_j | s_{j-1} \longrightarrow s_j) \right\} \quad \text{(A.4)}$$

and the first term in braces involves all the states that have been visited before S^i was reached at time k. Hence maximization of $P(x, S(i))$ can be carried out in two steps, the first of which finds the sequence $s_0, s_1, \cdots, s_{k-1}$ maximizing the first term of (A.4). But that state sequence is the best one leading from s_0 to $s_k = S^i$, as we can see by comparing the first term of (A.4) with (A.2). Thus it follows from (A.3) that for *some* $i \in \{0, 1, \cdots, M\}$, the initial subsequence $s_0, s_1, \cdots, s_{k-1}$ of the optimal sequence s^* must be equal to the best sequence leading into state S^i at trellis depth k. Therefore, for any i, all nonoptimal sequences leading into $s_k = S^i$ can be eliminated from consideration. The Viterbi algorithm accomplishes this elimination iteratively:

1. set $J = 2$ (J denotes the trellis column depth);
2. set $i = 0$;
3. kill all nonoptimal "live" paths leading into $s_J = S^i$; if several optimal paths exist, kill all but one of them;
4. if $i < M$, increase i by 1 and repeat step 3, or else do step 5;
5. if $J = n$, stop (the sole live path is the desired one), or else increase J by 1 and repeat step 2.

As an example, we will carry out the Viterbi procedure on the trellis of Fig. 32, assuming that the output was $x = 01100$. In Fig. 33, we have attached probabilities

$$\prod_{j=1}^{J} P(s_j | s_{j-1})\, q(x_j | s_{j-1} \longrightarrow s_j) \quad \text{(A.5)}$$

to all the paths of length $J = 2$. We will kill all but the best paths leading into each of the states S^0, S^1, and S^2 (there is a tie between two paths leading into S^1 which is resolved arbitrarily). In Fig. 34, we attach probabilities (A.5) to all the live paths of length $J = 3$. We kill all but the best paths. In Fig. 35, we attach probabilities (A.5) to all live paths of length $J = 4$. Fig. 36 shows the final result of the procedure: a best path $S^0 S^2 S^1 S^0 S^0 S^1$ with its associated probability.

As stated above, the Viterbi algorithm may be used to estimate values of statistical parameters of Markov sources. The

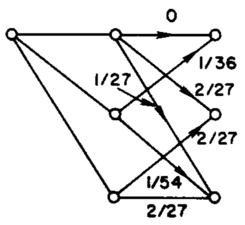

Fig. 33. The first two levels of the trellis of Fig. 32 with indicated path probabilities.

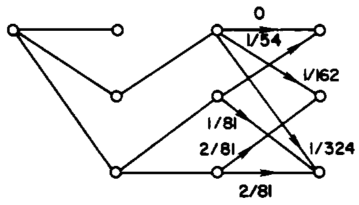

Fig. 34. The first three levels of the trellis of Fig. 32 with indicated probabilities of live paths.

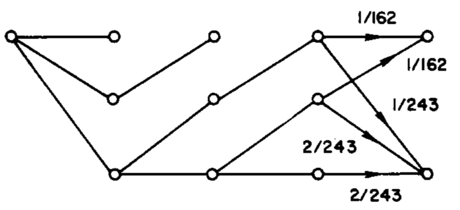

Fig. 35. The first four levels of the trellis of Fig. 32 with indicated probabilities of live paths.

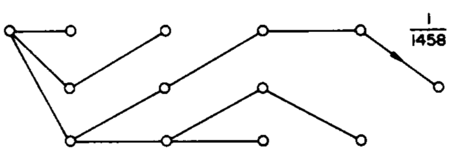

Fig. 36. The live paths of the trellis of Fig. 32 with probability attached to the decoded path.

approach is purely heuristic but it has worked well in the case of the first word-based channel model described in Section IX.

Viterbi extraction presumes that a set of source output sequences $\mathfrak{X} = \{x_1, x_2, \cdots, x_L\}$ has been observed, where $x_j = x_{j1}, x_{j2}, \cdots, x_{jn_j}$. It starts with some guess as to the structure of the Markov Source and its statistics $\hat{P}_0(S' \mid S)$ and $\hat{q}_0(x \mid S \rightarrow S')$. Let the Viterbi algorithm be carried out on the observation set $\mathfrak{X}$ and result in the set $\mathcal{S} = \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_L\}$ of estimated most probable state paths, where $\hat{S}_j = \hat{S}_{j0}, \hat{S}_{j1}, \cdots, \hat{S}_{jn_j}$. We say that the transition $\hat{S}_{ji} \rightarrow \hat{S}_{ji+1}$ corresponds to the output x_{ji}. Let $N(S, S', x)$ denote the number of times the transition $S \rightarrow S'$ takes place in any of the state sequences of the set $\mathcal{S}$ and corresponds to the output x. Define

$$N(S, S') = \sum_x N(S, S', x)$$

$$N(S) = \sum_{S'} N(S, S') \tag{A.6}$$

$$\hat{q}_1(x \mid S \longrightarrow S') = \frac{N(S, S', x)}{N(S, S')}$$

$$\hat{P}_1(S' \mid S) = \frac{N(S, S')}{N(S)}. \tag{A.7}$$

The probabilities (A.7) are used as new estimates of the Markov source probabilities, and the process is iterated until no substantial change in the estimated values takes place.[56]

If the Viterbi estimated state sequences $\{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_L\}$ were the ones actually realized by the source when it generated the data $\{x_1, x_2, \cdots, x_L\}$, then the formulas (A.7) would give the actual maximum likelihood estimates of the probabilities q and P underlying the source. The success of the Viterbi estimation method thus depends on both the closeness of the initial guesses $\hat{q}_0$ and $\hat{P}_0$ to the actual q and P values, and on whether the latter are "sharp" enough to allow the Viterbi algorithm to determine the true state path with sufficient accuracy. When these conditions are not met, then the extraction method must instead be based on the FB algorithm discussed in Appendix III.

Even if the final estimates $\hat{q}_j$ and $\hat{P}_j$ should account well for the data (i.e., be the most likely ones to have produced $\mathfrak{X} = \{x_1, x_2, \cdots, x_L\}$) they may still be inappropriate for our purpose of modeling, say, the behavior of the speaker-acoustic processor combination. If, because the data $\mathfrak{X}$ was either too small or fortuitously skewed, the transition $S \rightarrow S'$ never took place in the set $\mathcal{S}$ pertaining to the last iteration, then $\hat{P}_j(S' \mid S) = 0$ even though the true $P(S' \mid S)$ is nonzero. In that case, the use of $\hat{P}_j$ as an estimate of P might have very bad consequences for the speech recognizer. It is thus advisable to "smooth out" the estimates provided by the Viterbi extractor. The formulas

$$\hat{q}(x \mid S \longrightarrow S') = \frac{k_q}{k_q + N(S, S')} q_0(x \mid S \longrightarrow S') + \frac{N(S, S')}{k_q + N(S, S')} \hat{q}_j(x \mid S \longrightarrow S')$$

$$\hat{P}(S' \mid S) = \frac{k_p}{k_p + N(S)} P_0(S' \mid S) + \frac{N(S)}{k_p + N(S)} \hat{P}_j(S' \mid S) \tag{A.8}$$

have worked out well in practice when the values of k_p and k_q were experimentally optimized. The values $N(S, S')$ and $N(S)$ in (A.8) are those obtained during the last iteration. Note that the deviation of $\hat{q}(x \mid S\, S')$ from $\hat{q}_j(x \mid S\, S')$ will be significant only for rarely observed transitions $S \rightarrow S'$.

Appendix III.
The Forward-Backward Algorithm

The Viterbi algorithm finds the *a posteriori* most probable state sequence $s = s_0, s_1, \cdots, s_n$ given an observed output sequence $x = x_1, x_2, \cdots, x_n$. However, it does not determine for *any* i the most probable transition $s_{i-1} \rightarrow s_i$ given that x was observed. To find the former, it turns out to be necessary to calculate the probabilities of all the transitions at the ith-*trellis level* (those into the states of the $(i + 1)$th-trellis column). This is done efficiently by the FB algorithm.[57]

Denote the sequence $x_k, x_{k+1}, \cdots, x_l$ by x_k^l. Let $P\{s_t = S, x_1^t\}$ be the probability that the source generated the se-

[56] The jth probability estimates $\hat{q}_j(x \mid S\, S')$ and $\hat{P}_j(S' \mid S)$ are based on Viterbi decoding of the same data $\{x_1, x_2, \cdots, x_L\}$ with the Markov source modeled by statistics $\hat{q}_{j-1}(x \mid S \rightarrow S')$ and $\hat{P}_{j-1}(S' \mid S)$. It is reasonable to stop the iteration when the differences between $\hat{q}_j$ and $\hat{q}_{j-1}$ and between $\hat{P}_j$ and $\hat{P}_{j-1}$ become appropriately small.

[57] The version presented here was introduced into information-theoretic literature by [13]. Baum *et al.* have considered a very similar problem [14], [15].

quence x_1^t and ended up in state S (while starting in S^I). Let $P\{x_{t+1}^n \mid s_t = S\}$ be the probability that starting in state S at time t the source generates x_{t+1}^n (and ends up in S^F). Let $P\{s_t = S', x_t \mid s_{t-1} = S\}$ be the probability that if the source is in state S at time $t - 1$, it makes the transition into $s_t = S'$ at time t and as a result generates x_t. Then it follows directly from the Markov property, that the probability that x_1^n is generated and the transition $S \rightarrow S'$ is made at time t is given by

$$P\{s_{t-1} = S, s_t = S', x_1^n\} = P\{s_{t-1} = S, x_1^{t-1}\} \cdot P\{s_t = S', x_t \mid s_{t-1} = S\} \cdot P\{x_{t+1}^n \mid s_t = S'\}. \qquad \text{(A.9)}$$

Furthermore,

$$P\{s_t = S_1, x_1^t\} = \sum_{S'} P\{s_{t-1} = S', x_1^{t-1}\} P\{s_t = S, x_t \mid s_{t-1} = S'\} \qquad \text{(A.10)}$$

$$P\{x_t^n \mid s_{t-1} = S\} = \sum_{S'} P\{x_{t+1}^n \mid s_t = S'\} P\{s_t = S', x_t \mid s_{t-1} = S\}. \qquad \text{(A.11)}$$

The recursions (A.10) and (A.11) enable us to calculate the probability (A.9) in an easy manner. In the *forward* direction, if for each state S' of the tth-trellis column we know the probability $P\{s_{t-1} = S', x_1^{t-1}\}$, then (A.10) allows a one step calculation of such probabilities for the states of the next, $(t + 1)$th-trellis column. In the *backward* direction, if for each state S' of the $(t + 1)$th-trellis column we know the probability $P\{x_{t+1}^n \mid s_t = S'\}$, then (A.11) allows a one step calculation of such probabilities for the previous tth-trellis column.

The FB algorithm then consists of three steps:

1. using (A.10), calculate the probabilities $P\{s_t = S, x_1^t\}$ recursively for all states of trellis columns $t = 1, 2, \cdots, n$;
2. using (A.11), calculate the probabilities $P\{x_t^n \mid s_{t-1} = S\}$ recursively for all states of trellis columns $t = n - 1, n - 2, \cdots, 1$;
3. for all trellis transitions on any level calculate the probabilities $P\{s_{t-1} = S, s_t = S', x_1^n\}$ using formula (A.9).

As an example, we will carry out the FB algorithm on the trellis of Fig. 32 under the assumption that $x = 01100$ was generated.

First the forward part. Fig. 37 shows the probabilities $p\{s_t = S, x_1^t\}$ for $t = 0, 1, 2$, written next to the states of the first three columns, and the probabilities $P\{s_2 = S, x_1^2 \mid s_1 = S'\}$ attached to the transitions. The reader can check for himself that the relation between the second and third columns is that of (A.10). Fig. 38 shows the forward probabilities $P\{s_t = S, x_1^t\}$ attached to all states of the entire trellis.

For the backward direction, Fig. 39 shows the probabilities $P\{x_t^n \mid s_{t-1} = S\}$ attached to the states of trellis columns $t = 5, 4, 3, 2$, as well as the probabilities $P\{s_3 = S', x_3 \mid s_2 = S\}$ attached to the appropriate transitions. Fig. 40 shows the backward probabilities $P\{x_t^n \mid s_{t-1} = S\}$ attached to all states of the entire trellis.

Using Figs. 38 and 40, one can now calculate probabilities of various transitions by formula (A.9). For instance,

$$P\{s_2 = S^0, s_3 = S^1, x_1^n\} = \frac{1}{36} \times \frac{2}{9} \times \frac{1}{36} = \frac{1}{5832}$$

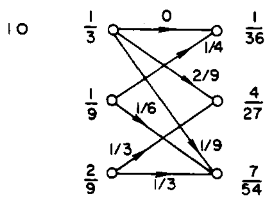

Fig. 37. First two levels of the trellis of Fig. 32 with second level transition probabilities and forward state probabilities.

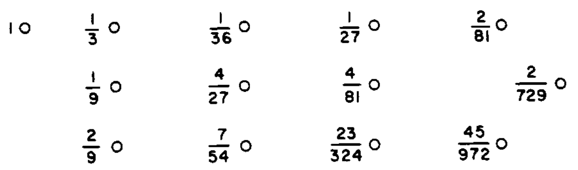

Fig. 38. Forward probabilities for all states of the trellis of Fig. 32.

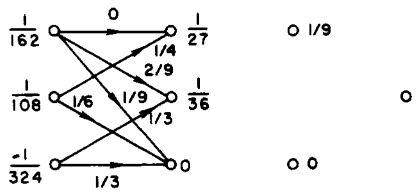

Fig. 39. Last three levels of the trellis of Fig. 32, third level transition probabilities, and backward state probabilities.

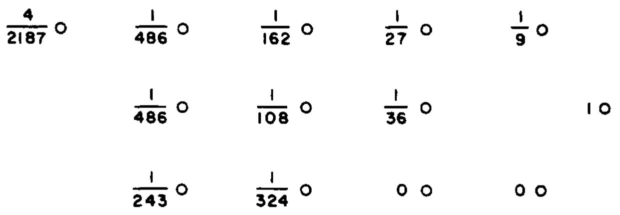

Fig. 40. Backward probabilities for all states of the trellis of Fig. 32.

or

$$P\{s_1 = S^2, s_2 = S^1, x_1^n\} = \frac{2}{9} \times \frac{1}{3} \times \frac{1}{108} = \frac{1}{1458}.$$

As mentioned in Appendix II, our main motivation for introducing the FB algorithm is statistics extraction when the Markov Source parameter values q and P are not sharp enough to guarantee sufficiently successful state estimation by the Viterbi algorithm, or when the initial guess $\hat{q}_0$, $\hat{P}_0$ is too unreliable.

In *FB extraction*, the FB algorithm is first performed on observed data $\mathfrak{X} = \{x_1, x_2, \cdots, x_L\}$, using guessed Markov source probabilities $\hat{q}_0(x \mid S \rightarrow S')$ and $\hat{P}_0(S' \mid S)$. In this way probabilities $P\{s_{it-1} = S, s_{it} = S', x_i\}$ are obtained. Adding their values for all output sequence positions t such that $x_{it} = X$ then results in $N(S, S', X)$. Or, more formally,

$$N(S, S', X) = \sum_{i=1}^{L} \sum_{t=1}^{n_i} \delta(x_{it}, X) P(s_{it-1} = S, s_{it} = S', x_i) \qquad \text{(A.12)}$$

where $\delta(\ ,\ )$ denotes the Kronecker delta function. The quantities $N(S, S')$, $N(S)$, $\hat{q}_1(X \mid S\, S')$, and $\hat{P}_1(S' \mid S)$ are then derived from $N(S, S', X)$ by formulas (A.6) and (A.7). The obtained values of $\hat{q}_1$ and $\hat{P}_1$ are used as the new guessed Markov source probabilities in the next iteration step on the same data $\mathfrak{X}$. The process continues until no substantial changes in these guessed probabilities take place. Again, the "smoother" probabilities (A.8) should be used for the final Markov source model.

The difference between Viterbi and FB extraction is that the latter uses all the available information in its attempt to determine the source statistics that would account best for the data. For instance, if the Viterbi decision between two state paths is close, this fact as well as the identity of other than the most probable path is thrown away. On the other hand, the FB extraction carefully records, through its use of formula (A.12), all contributions of the various transitions in direct proportion to their a posteriori probability. Unfortunately, while Baum *et al.* [14], [15] proved that after every iteration the F-B derived probabilities $\hat{q}_i$ and $\hat{P}_i$ account for the data better than the preceding ones $\hat{q}_{i-1}$ and $\hat{P}_{i-1}$ do,[58] the continued improvement resulting from iteration can only be shown to lead to a local optimum that is conditioned on the initial guess $\hat{q}_0$, $\hat{P}_0$.[59]

Acknowledgment

The concepts and results discussed in this paper are the work of many people. The author would like to gratefully acknowledge the contribution of all his colleagues in the IBM Speech Processing Group who are engaged in a dedicated and cooperative effort to build a continuous speech recognizer. A directory of research involvement with the presented material seems appropriate:

L. R. Bahl–Statistical extraction and decoding algorithms (Sections IV through VIII).

J. K. Baker–Viterbi decoding (Section VI).

R. Bakis–Word-based channel model and automatic training (Sections IX and X).

P. S. Cohen–Phonology and Phonetics (Section III).

N. R. Dixon–Phone-based acoustic processor (Section II)

R. L. Mercer–Phonetic rule implementation (Sections III and IV), formulation of decoding concepts.

H. F. Silverman–Phone-based acoustic processor (Section II), computer and hardware support.

C. C. Tappert–Modular word-based channel model (Sections IX and X).

L. R. Bahl, J. K. Baker, P. S. Cohen, and R. L. Mercer–have carried out the experiments described in Sections VI and VII.

The author has also benefited from discussions with other group members engaged in nonstatistical work: J. M. Baker, S. K. Das, M. Fitzgerald, B. Lewis, and R. Riekert.

Finally, he would like to express his thanks to P. S. Cohen and R. L. Mercer for their many helpful suggestions regarding his manuscript, and to C. Solanto who typed it and put up patiently with the author's handwriting and many revisions.

[58] The source $\hat{q}_i$, $\hat{P}_i$, generates the observed data $\mathfrak{X}$ with higher probability than does the source $\hat{q}_{i-1}, \hat{P}_{i-1}$.

[59] On the other hand, it is shown in Section VII that an immense improvement in recognizer performance is attainable when probabilities $\hat{q}$, $\hat{P}$ rather than $\hat{q}_0, \hat{P}_0$ are used.

References

[1] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Dev.*, vol. 13, pp. 675-685, Nov. 1969.

[2] K. S. Zigangirov, "Some sequential decoding procedures," *Probl. Pered. Inform.*, vol. 2, no. 4, pp. 13-25, 1966.

[3] R. Bakis, "Continuous-speech word recognition via centisecond acoustic states," presented at the 91st Meeting Acoustical Society of America, Washington, DC, Apr. 1976.

[4] W. Klein, R. Plomp, and L. C. W. Pols, "Vowel spectra, vowel spaces, and vowel identification," *J. Acoust. Soc. Amer.*, vol. 48, pp. 999-1009, 1970.

[5] J. Lyons, *Introduction to Theoretical Linguistics.* Cambridge, England: Cambridge Univ. Press, 1969.

[6] H. F. Silverman and N. R. Dixon, "A parametrically controlled spectral analysis system for speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 369-381, Oct. 1974.

[7] N. R. Dixon and H. F. Silverman, "A description of a parametrically controlled modular structure for speech processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 87-91, Feb. 1975.

[8] N. R. Dixon and H. F. Silverman, "A general language-operated decision implementation system (GLODIS): Its application to continuous speech segmentation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. *ASSP-24*, 1976, to appear.

[9] P. S. Cohen and R. L. Mercer, "The phonological component of an automatic speech recognition system," in D. R. Reddy, Ed., *Invited Papers of the IEEE Symposium on Speech Recognition, April 15-19, 1974.* New York: Academic Press, 1975, pp. 275-320.

[10] C. C. Tappert, work in progress.

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

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

[13] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. *IT-20*, No. 2, pp. 284-287, March 1974.

[14] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, no. 6, pp. 1554-1563, Dec. 1966.

[15] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171, 1970.

[16] F. Jelinek, L. R. Bahl, and R. L. Mercer, "The design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 250-256, May 1975.

[17] 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, July 1975.

[18] L. R. Bahl, J. K. Baker, P. S. Cohen, N. R. Dixon, F. Jelinek, R. L. Mercer, and H. F. Silverman, "Preliminary results on the performance of a system for the automatic recognition of continuous speech," to be presented at the Proc. 1976 IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Philadelphia, PA, Apr. 1976.

[19] N. Abramson, *Information Theory and Coding.* New York, McGraw-Hill, 1963.

[20] F. Jelinek, *Probabilistic Information Theory*, New York: McGraw-Hill, 1968.

[21] R. G. Gallager, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[22] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. and Cybernetics*, vol. SSC-4, pp. 100-107, July 1968.

[23] D. R. Reddy, "Speech recognition by machine: A review," this issue, pp. 501-531.

[24] L. Shockey and R. Reddy, "Quantitative analysis of speech perception," in Fant, Ed., *Proceedings of the Speech Communication Seminar.* New York: Wiley, 1975.