

Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech

FREDERICK JELINEK, FELLOW, IEEE, LALIT R. BAHL, MEMBER, IEEE, AND ROBERT L. MERCER

Abstract—Most current attempts at automatic speech recognition are formulated in an artificial intelligence framework. In this paper we approach the problem from an information-theoretic point of view. We describe the overall structure of a linguistic statistical decoder (LSD) for the recognition of continuous speech. The input to the decoder is a string of phonetic symbols estimated by an acoustic processor (AP). For each phonetic string, the decoder finds the most likely input sentence. The decoder consists of four major subparts: 1) a statistical model of the language being recognized; 2) a phonemic dictionary and statistical phonological rules characterizing the speaker; 3) a phonetic matching algorithm that computes the similarity between phonetic strings, using the performance characteristics of the AP; 4) a word level search control. The details of each of the subparts and their interaction during the decoding process are discussed.

I. INTRODUCTION

MOST CURRENT attempts at automatic speech recognition are formulated in an artificial intelligence framework. A mini-universe of discourse is selected (e.g., repair of faucets [1], geophysical inquiry [2], or chess playing [3]) and described syntactically and semantically. This description deterministically limits the permissible set of utterances. A complex interaction of acoustic, syntactic, and semantic processors then directs the search for the spoken sentence. The search is often based on confidence, fit, and frequency metrics derived in an ad hoc intuitive manner.

In this paper we describe the structure of a linguistic statistical decoder (LSD) that will be implemented by the Speech Processing Group of the IBM Thomas J. Watson Research Center. It constitutes an attempt to model statistically various aspects of speech production and recognition and to use decision procedures familiar in the field of information transmission in order to recognize and transcribe speech. A statistics-based system was previously implemented by Dixon and Tappert [4], [5] at IBM, Raleigh, N.C.; this serves as the starting point for the efforts here.

Four problems of particular interest to information theorists will be introduced. None of these problems has received adequate analytic treatment in the literature. First, we must decode in a system in which the codewords are not under the control of the system designer. The information sequence is a sentence, and the corresponding transmitted codeword is the phonetic representation of the sentence,

which is completely determined by the phonology of the language and the speaker. Second, there is a probabilistic relationship between the information sequence and the codeword that is transmitted. This is due to the fact that even a single speaker can pronounce the same sentence in many different ways. Third, the number of symbols in a codeword is a random variable, since word lengths and sentence lengths are variable. Finally, the recognition channel commits insertion and deletion errors in addition to the usual substitution errors. The insertion and deletion errors are due to imperfect segmentation, and the substitution errors are due to imperfect classification.

Fig. 1 is a block diagram of the speech recognition system. The acoustic processor (AP) uses signal processing and pattern recognition techniques to extract phonetic information from the acoustic signal. In the AP that we will consider here, this phonetic information is in the form of a phonetic string. In general, the AP may provide more detailed information and may even provide probabilistic information. The linguistic decoder then uses the structure of the language—its vocabulary, phonological rules, syntax, and semantics—to produce an orthographic transcription of the phonetic string.

The aim of this work is to recognize carefully spoken utterances that are produced in a relatively noiseless environment. The utterances are drawn from a corpus, which at present consists of two million words of patent applications in the field of laser technology. The vocabulary is roughly 10 000 words. We use this corpus to construct a statistical model of the language, and this is used by the LSD during the decoding process. The details of this model are discussed in Section IV. Sections II and III describe the speaker and AP models and the estimation of probabilities for them. In Sections V and VI we explain the principles and the implementation of the LSD. Section VII deals with some refinements of the model and decoder.

II. FRONT END MODEL

Fig. 2 is a diagram of the speaker-AP model, which will be referred to as the front end. It consists of three parts—an essentially phonemic dictionary, a set of phonological rules that models the speaker, and a noisy channel that models the AP. The front end accepts orthographic text as input and produces as output a string of symbols from a phonetic alphabet $A = (a_1, a_2, \dots, a_K)$.

The phonemic dictionary and phonological rules are described in detail by Cohen and Mercer [6]. The phonemic dictionary contains a set of phonemic base forms for each word of the vocabulary. Each of these represents a basic pronunciation of the word from which a number of other

Manuscript received March 29, 1974.

F. Jelinek and R. L. Mercer are with the Computer Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y.

L. R. Bahl is with the Computer Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., and also the Department of Electrical Engineering and Computer Science, Columbia University, New York, N.Y.

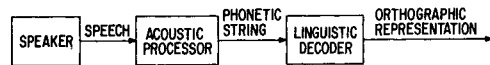


Fig. 1. Block diagram of speech recognition system.

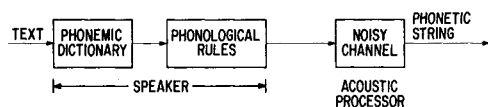


Fig. 2. Block diagram of front end model.

pronunciations are to be derived by the application of the phonological rules. For example, the word *library* has at least three phonemic base forms: /la¹brəri/, /la¹brəri/, /la¹beri/.

The phonological rules account for variations in pronunciation due to surrounding words, differences in dialect and idiolect, rate of speech, etc. A typical phonological rule has the form $\alpha \rightarrow \beta/\gamma \text{ --- } \delta$, where α and β are strings over the phonetic alphabet A , and γ and δ are sets of such strings that define the context in which the rule can be applied. All rules are optional and apply on a single left-to-right pass through the base-form string.

The phonemic dictionary and the phonological rules comprise the speaker model in the following way. Given the string of words corresponding to a complete utterance, base forms for each of the words are selected with speaker-dependent probabilities. The phonological rules then apply to the base-form string with further speaker-dependent probabilities. Conceptually, this process generates all the possible pronunciations with associated probabilities. These are conveniently represented by a directed graph with a single starting node and a single ending node [7], such that any path through this graph traces out a possible pronunciation, with its associated probability obtained by forming the product of the probabilities on the branches. The actual application of the phonological rules to a base-form string is a very time-consuming process. Except for rules that operate across word boundaries, the rules are preapplied to the base forms to obtain a directed-graph representation of each word. To account for the rules that do involve word boundaries, the graphs have multiple starting and ending nodes that are labeled with the conditions that must be met by the preceding and following words, respectively. As an example, Fig. 3 shows the directed-graph lexical entries for *apprentice* and *sorcerer* and the resulting directed graph for *apprentice sorcerer* when preceded and followed by silence. Nodes that arise from application of the same set of phonological rules have the same probability distribution on their outgoing transitions. Those nodes in the graph through which all paths must pass are called confluent nodes. The phonology is such that, except for a few very short words, each directed-graph lexical entry has at least one confluent node.

The AP model is intended to account for the segmentation and classification errors that occur during the signal processing and pattern recognition process. The output alphabet of the AP is denoted by $B = (b_1, \dots, b_J)$. Because

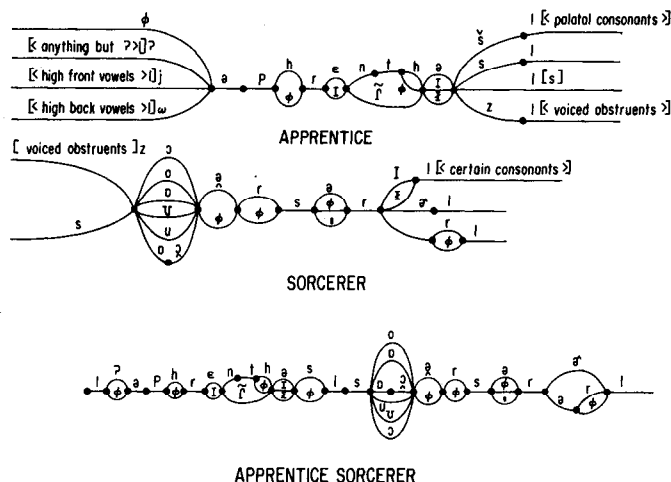
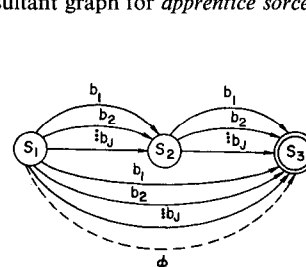
Fig. 3. Directed-graph lexical entries for *apprentice* and *sorcerer* and resultant graph for *apprentice sorcerer*.

Fig. 4. Simple example of channel model for AP.

of segmentation errors, the channel model must handle insertion and deletion errors as well as substitution errors. Models for such channels are discussed in detail by Bahl and Jelinek [8]. A simple example of such a model is one in which we associate with each input symbol $a_k \in A$ a probabilistic finite-state machine $F(a_k)$ that generates the possible outputs for the input a_k . Fig. 4 shows the state transition diagram of such a machine, which has initial state S_1 and final state S_3 . Associated with states S_1 and S_2 are probability distributions on their outgoing transitions, and associated with each transition is an output produced by that transition. Some transitions, called null transitions, have no associated output. These are drawn in dotted lines and labeled with a ϕ in the state transition diagram. In the example of Fig. 4, the null transition from state S_1 to state S_3 represents a deletion, the remaining transitions from S_1 to S_3 via S_2 produce two outputs and, therefore, represent insertions. To accurately reflect the types of errors made by the AP, it may be necessary that the machines $F(\cdot)$ be associated not with single symbols from A but with strings of symbols from A . It may, of course, also be necessary to consider machines with more than three states.

Both the directed-graph lexical entries and the probabilistic finite-state machines $F(\cdot)$ are Markov chains. Given a word sequence $W_{1,m} = w_1, w_2, \dots, w_m$, we can construct a composite Markov chain that generates all of the possible front end sequences for $W_{1,m}$ by first concatenating the directed-graph lexical entries for the words $w_1, w_2, \dots, w_m$ and then embedding in each transition the finite-state machine $F(\cdot)$ associated with the label of that transition.

III. ESTIMATION OF PROBABILITIES FOR FRONT END MODEL

To carry out the decoding process, we must know 1) the probabilities of the transitions in each directed-graph lexical entry, and 2) the probabilities of the transitions in each of the probabilistic finite-state machines $F(\cdot)$. One way of determining these probabilities is to transcribe by hand a large amount of speech data for each speaker. Regarding this transcription as the true input of the AP, one can in principle estimate the probabilities associated with the machines $F(\cdot)$ by comparison with the actual AP output for the same utterance. One could also estimate the probabilities associated with the directed-graph lexical entries by determining which path in the directed graph of the utterance is identical to the transcription. The amount of speech that must be transcribed to make reliable determinations in this manner is very large. Since hand transcription of speech is a very slow and expensive process, this makes the approach an unattractive one. Furthermore, different transcribers often produce different transcriptions for the same utterance, which leads us to question the consistency of this approach.

We now propose a method for estimating probabilities that does not involve hand transcription. Baum, Petrie, Soules, and Weiss [9] have described an iterative method for obtaining the maximum-likelihood estimate of the parameters of a Markov chain, given a sample of the output generated by the chain. Each iteration can be carried out by using the algorithm of Bahl, Cocke, Jelinek, and Raviv [10]. This method can be applied to the problem. We proceed as follows. Text is read by the speaker and processed by the AP, which produces a phonetic string. The composite Markov chain for this text is constructed by the method described in the previous section. The manner in which this chain is constructed places constraints on the probabilities associated with its transitions. For example, each subchain corresponding to a particular machine $F(a_k)$ must have the same probabilities. The method of Baum *et al.* may be applied to obtain maximum-likelihood estimates of the probabilities on the composite Markov chain. The probabilities associated with each subchain $F(a_k)$ can then be obtained by combining the probabilities associated with each occurrence of this subchain in the composite chain. This method requires some initial crude estimates of the probabilities, which can be obtained by examining a small amount of data.

Although the probabilities for the phonemic dictionary and phonological rule models will certainly be speaker dependent, those for the AP model will be much less so because the AP itself is trained to the individual speaker. Once probabilities have been estimated for many different speakers, it will be relatively easy to estimate probabilities for a new speaker by choosing as an initial estimate those of the known speaker who is dialectally most similar.

IV. LANGUAGE MODEL

The aim of any speech recognition system is to determine for an acoustic signal A and a language L that sequence of words $W \in L$ which maximizes the *a posteriori* probability $\Pr\{W|A;L\}$. Now $\Pr\{W|A;L\} = \Pr\{W|L\} \cdot$

$\Pr\{A|W;L\}/\Pr\{A|L\}$. The front end model provides us with $\Pr\{A|W;L\}$, and it is the function of the language model (LM) to provide us with $\Pr\{W|L\}$. The term $\Pr\{A|L\}$ in the denominator is independent of W and, therefore, need not be known.

We employ a probabilistic finite-state model to define the language L and provide us with the probabilities $\Pr\{W|L\}$. The probabilities associated with the model will be derived from the laser patent corpus. It should be noted that the decoding procedure is in no way dependent on the finite-state nature of the LM; it asks only for the probabilities $\Pr\{W|L\}$, caring nothing about how they are obtained.

We are well aware of the opinion held by linguists that finite-state models cannot account for the performance or competence of speakers of a natural language. The following arguments can be advanced in favor of finite-state models.

Current linguistic research and even many speech recognition efforts are primarily concerned with semantic interpretation, while the aim here is simply transcription. We know of no syntactic-semantic theory that is sufficiently developed to support a real corpus as large as ours. Furthermore, most of the arguments against finite-state models have been directed at word n -gram models in which the possible state transitions are determined by observing the $(n+1)$ -grams that occur. For finite-state models in which the states are not word n -grams and in which the state transitions are determined by more sophisticated techniques, these criticisms are unfounded. No research into the adequacy of more general finite-state models has been done. With the availability of large computers, this appears to be an opportune moment to make such an investigation. Finally, one more advantage is that while there are many efficient computational techniques known for finite-state models, few techniques are available for models of any other type.

The states of the model are vectors of the form $S = (w_{-1}, \dots, w_{-k}, s_{-1}, \dots, s_{-l}, c_{-1}, \dots, c_{-m})$, where $w_{-1}, \dots, w_{-k}$ are the k most recent words, $s_{-1}, \dots, s_{-l}$ are the parts of speech of the $l (\geq k)$ most recent words, and $c_{-1}, \dots, c_{-m}$ are the representatives of the m content classes (defined later) immediately preceding word w_{-k} . The purpose of including the most recent parts of speech in the states of the model is to make use of local grammaticality, while the purpose of the content classes is to use some information about long-range semantic content. As an example, consider the sentence: *The future of America depends on the availability of cheap energy.* Using D for determiner, N for noun, P for preposition, V for verb, and A for adjective, the corresponding part-of-speech sequence is $DNPVNVPDNPAN$. If we assume that the only words that belong to content classes are *future* and *depend*, then for $k=1, l=2, m=1$, the resulting state sequence would be (ϕ, ϕ, ϕ, ϕ) , $(\text{The}, D, \phi, \phi)$, $(\text{future}, N, D, \phi)$, $(\text{of}, P, N, \text{future}(N))$, $(\text{America}, N, P, \text{future}(N))$, $(\text{depends}, V, N, \text{future}(N))$, $(\text{on}, P, V, \text{depend}(V))$, $\dots$, $(\cdot, \cdot, N, \text{depend}(V))$, where $S_0 = (\phi, \phi, \phi, \phi)$ is the initial state and $(\cdot, \cdot, N, \text{depend}(V))$ is a terminal state. Associated with a transition from state S_i to S_j is a word and its part of speech and the probability of that transition, which is denoted by $P(S_j | S_i)$.

We propose to obtain the transition probabilities by statistical analysis of the corpus. Thus it is necessary to label the corpus with parts of speech and to determine for it an appropriate set of content classes. Currently we are using a classification system of 29 parts of speech that have been obtained by conglomeration of the 133 parts of speech used by the Harvard Multiple-Path Syntactic Analyzer for English [12]. The part-of-speech labeling of the corpus is carried out by a probabilistic algorithm described elsewhere [13]. The algorithm agrees with human labeling roughly 98 percent of the time.

In the state description we have included the k most recent words explicitly. However, some words have significant statistical influence over a range of text; such words are called content words. Only adjectives, adverbs, nouns, and verbs are considered as possible content words. Since content words are of interest for their long-range effect, grammatical inflection due to number, tense, etc., is considered unimportant. Content words are, therefore, divided into equivalence classes called content classes, and one member of a class acts as a representative of that class.

For example, light (verb) represents the class {light (verb), lighted (verb), lighting (verb), lights (verb), lit (verb)}, while light (noun) represents the class {light (noun), lights (noun)}. Not all adjectives, adverbs, verbs, and nouns have long-range statistical influence. We have adopted the following procedure to determine those which do. After the text has been labeled with parts of speech, each of the potential content words is replaced by its representative. For each pair of representatives we determine the number of times they occur together within a specified span. If for any representative the co-occurrence statistics are significantly different from the average, then the class it represents is taken as a content class.

Once the text is labeled and the content classes determined, one may estimate the probabilities of transitions by determining the state sequence for the text and counting the transitions that actually occur. In estimating the probabilities, transitions that have never occurred should not necessarily be assigned zero probabilities. We are investigating several ways of estimating the probabilities of such transitions.

V. DECODING PRINCIPLES

We now consider the decoding of the AP output on a sentence-by-sentence basis. Given the AP output $Y_{1,N} = y_1, y_2, \dots, y_N$, we wish to find that sequence of LM state transitions $S_{0,M} = S_0, S_1, \dots, S_M$ such that $\Pr\{S_{0,M}\} \cdot \Pr\{Y_{1,N} | S_{0,M}\}$ is maximized. State S_0 must be a sentence initial state and state S_M a sentence final state in the LM.

Corresponding to any state sequence $S_{0,m} = S_0, S_1, \dots, S_m$, there is a unique word sequence $W_{1,m} = w_1, w_2, \dots, w_m$ determined by the state transitions. The probability that the LM assigns to $S_{0,m}$ is denoted by $P(S_{0,m}) = P(S_1 | S_0) \cdot P(S_2 | S_1) \cdot \dots \cdot P(S_m | S_{m-1})$.

Let $G(S_{0,m})$ be the graph embodying all possible phonetic realizations of the word sequence $W_{1,m-1}$ when followed by w_m , which provides the necessary phonological right context. Let g be a particular path through the graph

(denoted by $g \in G(S_{0,m})$), and let $u(g)$ be the probability associated with g as determined by the speaker model. Now for a given front end output $Y_{1,n} = y_1, y_2, \dots, y_n$ and a state sequence $S_{0,m}$, we define the conditional probability $Q(Y_{1,n} | S_{0,m})$ to be the probability that input $W_{1,m-1}$ (when followed by w_m) produces the output $Y_{1,n}$. Clearly,

$$Q(Y_{1,n} | S_{0,m}) = \sum_{g \in G(S_{0,m})} \Pr\{Y_{1,n} | g\} u(g). \quad (1)$$

Let $Q(Y_{1,N} | S_{0,m})$ denote the $(N + 1)$ -vector

$$\{Q(\phi | S_{0,m}), Q(Y_{1,1} | S_{0,m}), Q(Y_{1,2} | S_{0,m}), \dots, Q(Y_{1,N} | S_{0,m})\}.$$

An efficient method for computing $Q(\cdot | \cdot)$ is given in Bahl and Jelinek [8].

We assign to $S_{0,m}$ the likelihood $L(S_{0,m})$ given by

$$L(S_{0,m}) = P(S_{0,m}) \sum_{n=0}^N Q(Y_{1,n} | S_{0,m}) \cdot \Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}. \quad (2)$$

The term $P(S_{0,m})$ is, of course, the *a priori* probability of the state sequence $S_{0,m}$. $\Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}$ is the probability that the remainder of the output sequence $Y_{n+1,N}$ is produced by w_m (w_{m-1} merely provides the necessary left context) followed by some sequence of words ending in a period that can be produced from state S_m . Since we cannot know exactly how much of the output sequence $Y_{1,N}$ is accounted for by input $W_{1,m-1}$, it is necessary to sum over all possible values of n in (2). It should be noted that whereas $P(S_{0,m})$ involves the first m words, $Q(Y_{1,n} | S_{0,m})$ uses only the first $m - 1$ words, with w_m providing the right context for the directed-graph lexicon (DGL). Assuming that the pronunciation of the final word (which must be sentence terminal punctuation) is complete silence, the likelihood of (2) for a complete sentence $S_{0,M}$ becomes $P(S_{0,M}) \cdot \Pr\{Y_{1,N} | S_{0,M}\}$, which is exactly what we are trying to maximize. In fact,

$$\begin{aligned} \Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\} \\ = \sum_i \Pr\{Y_{n+1,N}, I = i | Y_{1,n}, w_{m-1}, w_m, S_m\} \end{aligned}$$

where I is a random variable whose value is the sentence length. When S_m is a terminal state, the only contribution to the sum is that for $I = m$, and thus $\Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}$ reduces to δ_{nN} (Kronecker delta).

In practice, the probabilities $\Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}$ cannot be obtained exactly, but we can make a reasonable approximation by neglecting the dependence of $Y_{n+1,N}$ on w_{m-1}, w_m, S_m . It is possible to make a simple Markov model for the AP output and estimate its parameters by direct observation. Such a model will provide us with probabilities $\Pr\{Y_{n+1,N} | Y_{1,n}\}$, which we use instead of $\Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}$ in (2). With this approximation, the likelihood function is, of course, no longer exact for incomplete sentences.

The LSD uses the stack algorithm of sequential decoding [14], [15] to find the most likely state sequence corresponding to a given front end output $Y_{1,N}$. Each partially decoded sequence $S_{0,m}$ is represented in the stack by an entry of the

form $(S_{0,m}; L(S_{0,m}))$. Entries in the stack are arranged in descending order of their likelihoods. We define the successors of an entry $(S_{0,m}; L(S_{0,m}))$ to be the entries $(S_{0,m+1} = S_{0,m}; S_{m+1}; L(S_{0,m+1}))$ and $(S_{0,m'} = S_{0,m-1}; S_m'; L(S_{0,m'}))$, where S_{m+1} is chosen such that $P(S_{m+1} | S_m) = \max_S \{P(S | S_m)\}$, and S_m' is chosen such that $P(S_m' | S_{m-1}) = \max_S \{P(S | S_{m-1}) | P(S | S_{m-1}) < P(S_m | S_{m-1})\}$. We assume that no two transitions out of S_{m-1} have identical probabilities. If they do, then the usual minor adjustment must be made in the definition of S_m' . Occasionally no S_m' exists, in which case there is only one successor.

Decoding proceeds as follows.

- 0) Initialize the stack to contain the single entry $(S_{0,1}; P(S_{0,1}) \cdot \Pr\{Y_{1,N}\})$, where $S_{0,1} = S_0, S_1$ is chosen such that $P(S_1 | S_0) = \max_S P(S | S_0)$.
- 1) Let $(S_{0,m}; L(S_{0,m}))$ denote the entry at the top of the stack, i.e., the one having maximum likelihood. If S_m is a final state, then we stop and accept $W_{1,m}$ corresponding to $S_{0,m}$ as the most likely input. Otherwise, we replace $(S_{0,m}; L(S_{0,m}))$ by its successors, reorder the stack, and repeat step 1.

VI. DECODER IMPLEMENTATION

The decoding algorithm of the previous section is feasible only if an efficient technique can be devised for evaluating the likelihoods $L(S_{0,m})$. The probabilities $P(S_{0,m})$ are easily obtained from the LM using the recursion $P(S_{0,m}) = P(S_{0,m-1}) \cdot P(S_m | S_{m-1})$. The probabilities $\Pr\{Y_{n+1,N} | Y_{1,n}, w_{m-1}, w_m, S_m\}$ have been approximated by $\Pr\{Y_{n+1,N} | Y_{1,n}\}$, which need to be computed only once for the output sequence $Y_{1,N}$. This leaves the evaluation of the probabilities $Q(Y_{1,N} | S_{0,m})$. An algorithm for computing these probabilities is discussed in [8]. The complexity of the calculation is roughly proportional to the product of the number of nodes in $G(S_{0,m})$ and N , the number of symbols in the output sequence. When computing $Q(Y_{1,N} | S_{0,m+1})$ and $Q(Y_{1,N} | S_{0,m'})$, it is not necessary to deal with the complete graphs for $G(S_{0,m+1})$ and $G(S_{0,m'})$ but only with the parts that differ from $G(S_{0,m})$. Let us denote by $G^+(S_{0,m})$ the subgraph of $G(S_{0,m})$ up to the rightmost confluent node and by $Q^*(Y_{1,n} | S_{0,m})$ the probabilities

$$Q^*(Y_{1,n} | S_{0,m}) = \sum_{g \in G^+(S_{0,m})} \Pr\{Y_{1,n} | g\} u(g),$$

$$n = 0, 1, \dots, N.$$

The probabilities $Q^*(Y_{1,N} | S_{0,m})$ are obtained as a by-product of the computation of $Q(Y_{1,N} | S_{0,m})$. To compute the probabilities $Q(Y_{1,N} | S_{0,m+1})$ and $Q(Y_{1,N} | S_{0,m'})$, one need only know $Q^*(Y_{1,N} | S_{0,m})$ and those portions of the graphs $G(S_{0,m+1})$ and $G(S_{0,m'})$ that differ from $G^+(S_{0,m})$. Therefore, we store the vector $Q^*(Y_{1,N} | S_{0,m})$ with the stack entry for $S_{0,m}$.

Fig. 5 is a block diagram of the LSD. The decoding process is controlled by the word level search controller (WLSC), which maintains the likelihood stack. Each entry in the stack contains the path $S_{0,m}$, the likelihood $L(S_{0,m})$, the probability $P(S_{0,m})$, and the vector $Q^*(Y_{1,N} | S_{0,m})$.

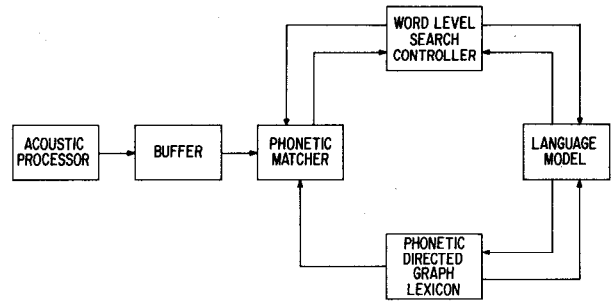


Fig. 5. Block diagram of LSD.

One iteration in the decoding process is as follows. The WLSC locates the path $S_{0,m}$ having the highest likelihood. The successors of $S_{0,m}$ must now be determined. To do so the WLSC sends the last three states of path $S_{0,m}$ to the LM. The LM determines S_{m+1} and the probability $P(S_{m+1} | S_m)$ and returns this information to the WLSC. Simultaneously the LM sends the word triplet w_{m-1}, w_m, w_{m+1} corresponding to the state sequence $S_{0,m+1}$ to the DGL. The DGL uses these to determine the rightmost confluent node R of $G(S_{0,m})$ and $G^+(S_{0,m+1})$, which is that part of $G(S_{0,m+1})$ that lies to the right of R . The phonology is such that in virtually all cases R can be determined from w_{m-1} and w_m . At most, the DGL will need to know w_{m-2} , which could be provided by the WLSC through the LM if desired. The DGL sends $G^+(S_{0,m+1})$ to the matching algorithm. The matcher also receives $Q^*(Y_{1,N} | S_{0,m})$ from the WLSC and computes $Q(Y_{1,N} | S_{0,m+1})$ and $Q^*(Y_{1,N} | S_{0,m+1})$, both of which it returns to the WLSC. The WLSC can now complete the entry for $S_{0,m+1}$ and insert it in the stack. The procedure for computing the entry for the other successor $S_{0,m'}$ is similar. The WLSC terminates the decoding process when the top of the stack corresponds to a complete sentence.

The LM that we have described generates word and part of speech pairs, and consequently the decoder finds the most likely sequence of words and parts of speech for a given utterance. The likelihood of a word sequence can be obtained by summing up the likelihoods of all the state sequences that give rise to that word sequence. The decoder could then operate on these likelihoods to find the most likely word sequence. We believe, however, that the loss of performance caused by using state sequence likelihoods rather than word sequence likelihoods will be negligible.

VII. REFINEMENTS

One possible shortcoming of the LM is that it does not take advantage of any phonetic similarity between different words. The LM can be modified in the manner shown in Fig. 6. Fig. 6(a) shows a portion of the state transition diagram of the LM. All the transitions indicated have words whose phonetic representation starts with the phonetic sequence corresponding to *non*. The phonetic representations of *nonsense* and *nonsensical* agree even further as do those of *nonagenarian* (*N*) and *nonagenarian* (*A*). Fig. 6(b) shows a portion of a modified transition diagram that produces the same words with the same

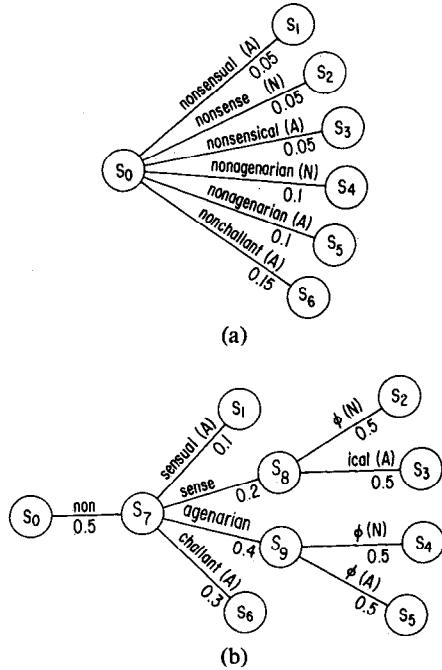


Fig. 6. Modification of LM state transition diagram to take advantage of phonetic similarity.

probabilities as Fig. 6(a), but takes advantage of the phonetic similarity of the words. The advantages of such a modification are twofold. First, the time used in computing $Q(Y_{1,N} | S_{0,m})$ is reduced because we need to compute these for *non* only once and use them for any word that starts with *non*. Second, the actual order of search may be altered beneficially, e.g., in Fig. 6(a), the transition S_0, S_6 will be investigated before either of the transitions S_0, S_4 or S_0, S_5 , while in Fig. 6(b) S_7, S_9 (which is a combination of S_0, S_4 and S_0, S_5 in Fig. 6(a)) is investigated before S_7, S_6 (which corresponds to S_0, S_6 in Fig. 6(a)).

Although we have described the decoding algorithm as replacing the top stack entry with its successors, it may be better to consider replacing the top several entries with their successors simultaneously. Furthermore, it may be worthwhile to construct more than two successors for each stack entry.

Another possible modification to the decoding algorithm would be to allow a two-way search, i.e., a search in which we decode the received utterance simultaneously in the forward and the reverse directions. To construct a reverse decoder we must of course build a reverse LM which produces the sentences for the language in reverse order. This reverse model can be easily obtained for any finite-state machine by reversing the arrows in the state transition diagram and replacing the probability $P(S_j | S_i)$ by ${}^rP(S_i | S_j) = P(S_j | S_i) \cdot P(S_i) / P(S_j)$, where $P(S_i)$ is the probability that a path through the state transition diagram will include S_i . The initial states of the reverse machine are the final states of the original machine and *vice versa*. The forward machine defined by $P(S_j | S_i)$ and the reverse machine defined by ${}^rP(S_i | S_j)$ are then isomorphic under sequence reversal. We will use a left superscript r to indicate

quantities related to the reverse model. A state sequence ${}^rS_{m_1,M}$ in the reverse machine corresponds to the transitions $S_{m_1} \leftarrow S_{m_1+1} \leftarrow \dots \leftarrow S_M$, where S_M is a final state. The speaker and channel models are similarly reversible, since they too are finite state. Once the reverse models have been obtained, a reverse decoder that decodes from the end of the received sequence $Y_{1,N}$ can be implemented in exactly the same manner as the forward decoder of Sections V and VI. We can combine the forward and reverse decoders to give us a two-way decoder. This decoder would have two stacks, one for the forward paths and one for the reverse, and we would extend the top stack entry that has the greater likelihood. When extending a forward path, we would not only construct the two successors discussed in Section V, but also those successors that connect this forward path to a reverse path. Let S_{0,m_1} be a forward path and ${}^rS_{m_2,M}$ be a reverse path and let us assume that the transition $S_{m_1} \rightarrow S_{m_2}$ is possible in the LM. Then we would construct the successor $S_{0,m_2} = S_{0,m_1} S_{m_2}$ and determine the likelihood for the complete path $S_{0,M} = S_{0,m_1} S_{m_2} {}^rS_{m_2,M}$ from the entries for S_{0,m_2} and ${}^rS_{m_2,M}$. The *a priori* probability of the complete path is given by

$$P(S_{0,M}) = \frac{P(S_{0,m_2}) \cdot P({}^rS_{m_2,M})}{P(S_{m_2})}. \quad (3)$$

The probability $\Pr \{Y_{1,N} | S_{0,M}\}$ is obtained as

$$\Pr \{Y_{1,N} | S_{0,M}\} = \sum_{n=0,N} \Pr \{Y_{1,n} | W_{1,m_2}\} \cdot \Pr \{Y_{n+1,N} | {}^rW_{m_2+1,M}\} \quad (4)$$

where W_{1,m_2} and ${}^rW_{m_2+1,M}$ are the word sequences corresponding to S_{0,m_2} and ${}^rS_{m_2,M}$. The probabilities on the right side of (4) are, of course, easily obtained by extending $Q^*(Y_{1,n} | S_{0,m_2})$ and ${}^rQ^*(Y_{1,n} | {}^rS_{m_2,M})$, respectively. The likelihood of the complete path $S_{0,M}$ is obtained by multiplying $P(S_{0,M})$ and $\Pr \{Y_{1,N} | S_{0,M}\}$, which have been obtained from (3) and (4). Similarly when extending a reverse path, we would try to connect to partial forward paths.

It would, of course, require considerable computation to check which partial reverse paths are connectable to a forward path being extended. A simple heuristic technique for limiting the search is as follows. For any $S_{0,m}$, let $n_{\max}(S_{0,m})$ be that value of n (presumed unique) which maximizes $\Pr \{Y_{1,n} | S_{0,m}\}$. Then $n_{\max}(S_{0,m})$ is the most likely right boundary for $S_{0,m}$. Similarly, we can define $n_{\max}({}^rS_{m_2,M})$ to be the most likely left boundary for ${}^rS_{m_2,M}$. When trying to connect S_{0,m_1} to ${}^rS_{m_2,M}$, we check if $n_{\max}({}^rS_{m_2,M}) - n_{\max}(S_{0,m_1}) \leq e$, where e is a small integer. If this condition is not met, we do not try to connect the two paths. To facilitate in checking this condition, we can keep for each $n = 0, 1, \dots, N$, a list of all forward paths $S_{0,m}$ such that $n_{\max}(S_{0,m}) = n$. Similar lists can be kept for reverse paths. To decide if $S_{0,m}$ should be connected to any reverse path, we consider only those reverse entries that are in the reverse path lists for $n_{\max}(S_{0,m}) + n'$, $n' \leq e$. The basic idea here is that $n_{\max}(S_{0,m})$ gives us the most likely right boundary for $S_{0,m}$, and we should try to connect $S_{0,m}$ only to those reverse paths that have a likely left boundary close to

$n_{\max}(S_{0,m})$. At any time we need only save one complete path—i.e., the one with the highest likelihood—and if its likelihood is higher than the partial paths in both stacks, decoding is terminated.

ACKNOWLEDGMENT

The authors are grateful for the many suggestions they have received from the members of the Speech Processing Group, especially P. Cohen. They also wish to express their thanks to Dr. John Cocke for many stimulating discussions.

REFERENCES

- [1] D. E. Walker, "Speech understanding through syntactic and semantic analysis," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, 1973, pp. 208–215.
- [2] W. A. Woods, J. Makhoul, J. Wolf, and P. Rovner, "Organizing a system for continuous speech understanding," presented at the 85th Meeting of the Acoustical Society of America, Boston, Mass., Apr. 1973.
- [3] D. R. Reddy, L. D. Erman, R. D. Fennell, and R. B. Neely, "The hearsay speech understanding system," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, 1973, pp. 185–193.
- [4] N. R. Dixon and C. C. Tappert, "Intermediate performance evaluation of a multi-stage system for automatic recognition of continuous speech," Tech. Rep. RADC-TR-73-16, 1973.
- [5] C. C. Tappert, N. R. Dixon, and A. S. Rabinowitz, "Application of sequential decoding for converting phonetic to graphemic representation in automatic recognition of continuous speech (ARCS)," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 225–228, June 1973.
- [6] P. S. Cohen and R. L. Mercer, "The phonological component of an automatic speech-recognition system," in *Proc. IEEE Symp. Speech Recognition*, 1974, pp. 177–188.
- [7] R. Alter, "Utilization of contextual constraints in automatic speech recognition," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 6–11, Jan. 1968.
- [8] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions, with applications to speech recognition," to appear in *IEEE Trans. Inform. Theory*.
- [9] 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. Statist.*, vol. 41, no. 1, pp. 164–171.
- [10] 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, pp. 284–287, Mar. 1974.
- [11] N. Chomsky, *Syntactic Structures*. New York: Humanities, 1957.
- [12] S. Kuno, "The multiple-path syntactic analyzer for English," Mathematical Linguistics and Automatic Translation, Computation Lab. Harvard Univ., Rep. NSF-8, 1963.
- [13] L. R. Bahl and R. L. Mercer, "Part-of-speech assignment by a statistical decision algorithm," IBM Res. Rep., to appear.
- [14] F. Jelinek, "A stack algorithm for faster sequential decoding of transmitted information," IBM Res. Rep., RC-2441, Apr. 1969.
- [15] —, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, pp. 675–685, Nov. 1969.