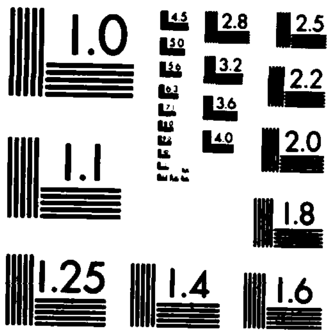
AD-A136 990 RESEARCH IN CONTINUOUS SPEECH RECOGNITION(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA R M SCHWARTZ ET AL. DEC 83 BBN-5485 N00014-81-C-0738 1/1

UNCLASSIFIED F/G 5/7 NL

END


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Bolt Beranek and Newman Inc.

Report No. 5485

Research in Continous Speech Recognition

Annual Report

December 1983

Prepared for:
Advanced Research Projects Agency

(15)

Report No. 5485

RESEARCH IN CONTINUOUS SPEECH RECOGNITION

Annual Report
1 October 1982 to 30 September 1983

Principal Investigator:
John Makhoul
(617) 497-3332

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

ARPA Order Nos. 4311/4707 Contract No. N00014-81-C-0738

Effective Date of Contract: 1 October 1981
Contract Expiration Date: 15 November 1983

This research was supported by the Advanced Research Projects Agency of the Department of the Defense and was monitored by ONR under Contract No. N00014-81-C-0738. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: BBN Report No. 5485 / 2. GOVT ACCESSION NO.: A136990	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): RESEARCH IN CONTINOUS SPEECH RECOGNITION	5. TYPE OF REPORT & PERIOD COVERED: Annual Report 1 Oct. 82 - 30 Sep 83
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Richard M. Schwartz, Yen-Lu Chow John Makhoul	8. CONTRACT OR GRANT NUMBER(s): N00014-81-C-0738
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Department of the Navy Arlington, VA 22217	12. REPORT DATE: December 1983
	13. NUMBER OF PAGES: 63
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of the document is unlimited. It may be released to the Clearinghouse, Dept. of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech recognition, phonetic recognition, Hidden Markov Model, phonetic context, acoustic-phonetic features, rule-based algorithms, network-decoding algorithms, Forward-backward algorithm, automatic training, triphone model, interpolated (detected) estimation, search strategy.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This annual report describes the work performed during the past year in an ongoing effort to design and implement a system that performs phonetic recognition of continuous speech. The general approach used it to develop a Hidden Markov Model (HMM) of speech parameter movements, which can be used to distinguish among the different phonemes. The resulting phoneme models incroporate the contextual effects of neighboring

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) cont

phonemes. One main aspect of this research is to incorporate both spectral parameters and acoustic-phonetic features into the HMM formalism.

TABLE OF CONTENTS

LIST OF FIGURES

1. OVERVIEW

1.1 Introduction

This annual report describes the work performed during the past year in an ongoing effort to design and implement a system that performs phonetic recognition of continuous speech. The general approach used is to develop a Hidden Markov Model (HMM) of speech parameter movements, which can be used to distinguish among the different phonemes. The resulting phoneme models incorporate the contextual effects of neighboring phonemes. One major aspect of this research is to incorporate both spectral parameters and acoustic-phonetic features into the HMM formalism.

Previous work on using acoustic-phonetic rules has yielded average phonetic recognition rates of 60-70%. We estimate that a minimum recognition rate of 80% is needed for a high-performance speech understanding system. We believe that the information needed to achieve the higher recognition accuracy is available in the speech signal, manifested in the form of spectral parameter sequences, as well as in the more global acoustic-phonetic features that are on the order of a phoneme. Current recognition systems make use of one or the other type of information. We believe that in order to achieve the level of performance that we desire, the recognition system must make use of both kinds of information in a coherent way.

1.2 Progress

During this past year, we completed a major milestone of the project, namely, the design and implementation of an initial phonetic recognition system on the VAX 11/780. Specifically, the following tasks were performed:

1. Designed and implemented a data structure to represent spectral probability densities and acoustic-phonetic features within the same formalism, and to allow extensive use of context-dependent phoneme models. Special attention has been paid to space/computation tradeoffs.

2. Designed and implemented a phonetic recognition system, referred to as the stack decoder. This program takes as input the acoustic representation of an unknown utterance and attempts to find the most likely phoneme sequence using a best-first search strategy.

3. Designed and implemented a system parameter training program, referred to as the forward-backward algorithm. Much effort has been spent on debugging and speeding up this program.

4. Developed software tools to debug the system interactively.

5. Carefully hand-labeled 110 utterances with phonetic transcriptions and phoneme segment boundaries. Upgraded and improved the labeling program to facilitate the labeling process. Recorded 250 new utterances for future use.

6. Performed an experiment with 110 utterances (5 minutes of speech) and obtained some initial recognition performance results.

In the following sections, we describe these tasks in greater detail. In Section 2, we present our basic system design and the underlying concepts. Details of our initial system implementation are given in Section 3. In Section 4, we describe an initial experiment that was performed to test the first implementation of the system, and conclusions are made to guide our research plans for the coming year.

2. SYSTEM DESIGN

In this section we describe the motivation for the approach we have taken to phonetic recognition. We also rev'ew the methods used and how they will fit together in tl final recognition system.

2.1 Combination of Methods

Phonetic recognition algorithms generally fall into one of two types. The first type, which we call _rule-based algorithms_, uses a set of heuristic acoustic-phonetic rules to segment speech into phoneme-sized units and then assigns phoneme labels to these segments. The second common type of phonetic recognition algorithms, which we call _network-decoding algorithms_, is usually based on a single parametric representation of speech sounds compiled into a network that represents alternate realizations of phonetic units. Given some input speech, the algorithm uses a dynamic programming search that attempts to find the optimal path through the network, based on a distance metric or a probabilistic error criterion. The advantages and disadvantages of each of the two types of algorithms are presented below.

2.1.1 Rule-Based Algorithms

Typically, the rules in a rule-based system are hierarchical. For example, the speech is first divided into sonorant and obstruent regions based on gross features, such as the amount of low-frequency or high-frequency energy. Then, within each of these regions, context-dependent rules are used to further subdivide the speech. These rules are most often based on explicit knowledge of the relation of certain acoustic-phonetic features to a particular articulatory gesture.

2.1.2 Network-Decoding Algorithms

Several phonetic recognition programs have been based on a network representation of speech (often known as a Hidden Markov Model). These methods do not segment the speech into phonetic units before labeling. Instead, they use a dynamic programming algorithm that considers all possible segmentations of the input. Associated with each segmentation is a best labeling and a corresponding score. The algorithm then chooses the segmentation and labeling that together result in the best score.

2.1.3 Comparison of Methods

The rule-based phonetic recognition system explicitly models the salient features of different phonemes by using those features that are best at making each phonetic distinction. By segmenting the speech before labeling, these algorithms avoid the expensive computation associated with dynamic programming. They can also make use of more global acoustic-phonetic features of phonemes that are difficult to represent in single-frame models. The primary drawback of these rule-based algorithms is that if decisions made in an early stage of the hierarchical analysis are incorrect (as they must be occasionally), decisions made in later stages are often not meaningful. While the processes that use the results of the phonetic recognition (such as a word matcher) can hypothesize various errors, they do not have the ability to score those errors accurately. Rather, they must rely on average confusion statistics to score these unlikely errors. Thus, a severe error in segmentation or labeling is quite likely. Such errors were found to be quite devastating in our speech understanding system.

In contrast, the hypothesize-and-test paradigm used by network-decoding algorithms typically computes detailed acoustic scores for a wider range of alternatives. While the correct phoneme will not always receive the highest score, it will at least always be considered, and will receive a score that is comparable to other scores assigned to other theories. Another way to describe such systems is that, by considering all reasonable alternatives, they degrade gracefully. Experience has shown that this feature is essential to a speech understanding system that combines many theories spanning different periods of input speech. A second very important feature of the network decoding algorithms is their capability for automatic training of the network model. The training method used is based on a bootstrapping technique. The training method used is based on a bootstrapping technique. A known sentence is given to the system along with a phonetic transcription. The system first finds the best alignment of the known input to the network. Then it uses the aligned input speech to update the network models, either by changing probability densities or by adding alternate paths in the network. In this way, large amounts of training speech can be incorporated into the network with relatively little effort.

The primary disadvantage of the network-decoding algorithms to date is that they use a uniform representation and metric for all frames of all phonemes in all contexts. The spectral parameters in a single frame are also not ideal because they are:

a. a high dimensional space

b. highly dependent on the speaker

c. do not include more global effects on the order of a phoneme.

Thus they do not directly model the known differences between phonemes. A second disadvantage associated with network decoding algorithms is that they typically require large amounts of computation.

2.1.4 Properties of a Unified Approach

As a result of our experience with various phonetic recognition methods, we have compiled a list of desired capabilities or properties for a phonetic recognition system that combines the advantages of the two basic methods. At this point we postulate that the performance of a phonetic recognition program is determined primarily by how well it models the acoustic realizations of different phoneme sequences. Below we list properties of a recognition algorithm which we feel will help achieve the goal of estimating models that are both complete and accurate.

1. The phonetic recognition algorithm chooses the phoneme sequence that is most likely, given the entire input specch signal. When used as a word matcher, it computes the score of the best alignment of the hypothesized words against the input speech.

2. The speech model incorporates both a spectral sequence model and the acoustic-phonetic features typically used in rule-based systems within the same formalism.

3. The system is able to be trained automatically with little human effort.

4. The system is amenable to retraining to a new speaker, or to multiple speakers, with only a short training period.

5. The system can be made to run faster by constraining the search significantly without degrading performance.

2.1.5 Block Diagram

Figure 1 illustrates the overall block diagram for the recognition system. The system can be divided into two major sections: the training component and the recognition component. The division between these sections is indicated by the horizontal dotted line. The training component uses two data bases. First, a small data base of carefully labeled speech is used to produce initial estimates of probability densities for spectral parameters and for acoustic-phonetic features (discussed further below). Then, a larger data base of phonetically transcribed (but not time-aligned) speech is used by the training algorithm to automatically improve the pdf estimates. The knowledge sources shown provide the training system with a model for the variability of speech parameters, and the effect of phonetic context, and a set of acoustic-phonetic feature definitions that have been found to be useful for making fine phonetic distinctions. These features are initially suggested by our experience with spectrum reading experiments. They are then further developed using our Acoustic-Phonetic Experiment Facility (APEF) [1] on a data base of phonetically labelled speech. These sources of human knowledge constrain the system along a smaller set of dimensions for modelling speech. The sources of knowledge are specifically <u>not</u> used to set thresholds or determine rules, but rather are used only as a structure on which the training system can build further knowledge.

The training system produces two sets of probability density functions (pdf's), for spectral parameters as a function of each frame in the phoneme, and for acoustic-phonetic features that are defined globally over a whole phoneme. These two sets of pdf's are then used by the network decoder to determine the most likely phoneme sequence corresponding to an input speech utterance.

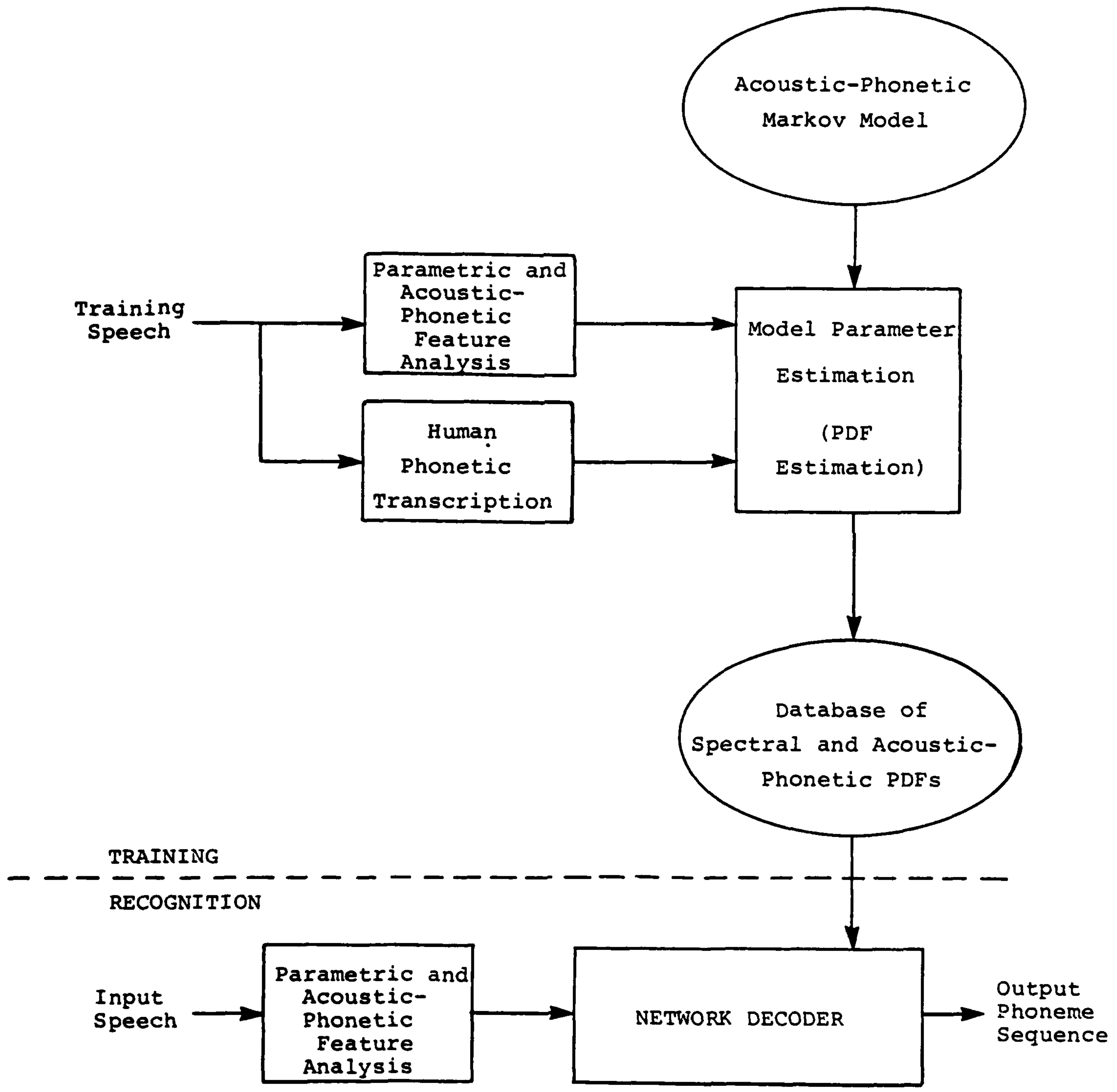

Acoustic-Phonetic Markov Model
Training Speech
Parametric and Acoustic-Phonetic Feature Analysis
Human Phonetic Transcription
Model Parameter Estimation (PDF Estimation)
Database of Spectral and Acoustic-Phonetic PDFs
TRAINING
RECOGNITION
Input Speech
Parametric and Acoustic-Phonetic Feature Analysis
NETWORK DECODER
Output Phoneme Sequence

FIG. 1. Block Diagram of phonetic-recognition system.

Each of these components will be discussed further in Section 3. In the remainder of this section, we outline briefly the basic methods and the major advances with respect to previously implemented network decoding algorithms.

2.2 Hidden Markov Model

The basic model of a phoneme that we use is a Hidden Markov Model (HMM) of spectral parameters as a function of time within the phoneme. In this subsection, we define the HMM and explain how it is used for the problem of phonetic recognition.

2.2.1 Definition

A simple example of a HMM is shown in Fig. 2. The HMM consists of a set of states in a Markov Chain. The probability of proceeding from any state to another state is given by a transition probability matrix. However, unlike the normal Markov Chain, which associates only one output symbol with each state, each state also has associated with it a probability for each output symbol or - in general - a probability density for the output. Assuming that the probability densities of the different states overlap, it is not possible to determine absolutely the sequence of states that produced a particular output sequence (thus the term "hidden").

In the example shown, there is a nonzero probability of proceeding between any two states, and the pdf's associated with the states are assumed to be independent. In modelling the acoustic realization of a phoneme, we can apply some reasonable

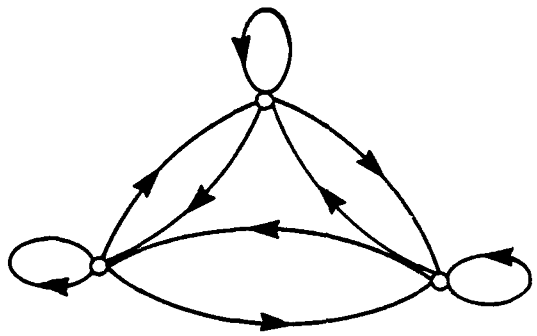


FIG. 2. Simple Hidden Markov Model.

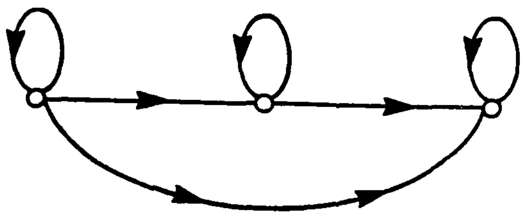


FIG. 3. Hidden Markov Model constrained to proceed from left to right.

constraints to the model that will allow its parameters (transition matrix and pdf's) to be more easily estimated. Figure 3 shows a simple example of a HMM in which the states are assumed to "flow" from left to right. That is, it is not possible to go from a later part of a phoneme to an earlier part of the same phoneme. The model does allow the deletion of the state at the center, under the assumption that if the person is speaking quickly, the target at the center of the phoneme may never be attained. The loops from a state back to itself (self-loops) allow for variable duration for any particular part of the phoneme. Finally, the pdf's associated with some of the transitions are "tied" together. This constraint allows the system to more easily estimate the fewer number of pdf's, while providing the flexibility to change the transition probabilities as needed.

2.2.2 Automated Training

The HMM structure affords a very powerful model for the parameters of speech as a function of time within a phoneme. However, it is also very important that the parameters of the HMM be easily estimated using a sufficient amount of speech such that the parameter estimates will be robust. First, using a small amount of carefully labeled speech (about 100 sentences), we determine an initial estimate of the spectral pdf's for each of the models for each phoneme. Initially, the pdf's are assumed to the same everywhere within a phoneme, and the transition matrix is assumed to be the same for all phonemes.

A large data base of phonetically transcribed speech is then used to estimate the pdf's, transition matrices, and the effect of phonetic context on these models more accurately. The algorithm used (termed the "Forward-Backward" or Baum [2] algorithm) first computes the probability of each possible alignment of the training speech given the model so far. Then, the new speech is used to update the model parameters in proportion to how well the particular alignment scored. This algorithm can be shown to determine the set of model parameters that maximizes the estimated probability of all of the training data. However, the maximum achieved is only guaranteed to be a local maximum - not a global maximum.

2.2.3 Network Decoder

Once this model has been sufficiently trained, another algorithm (similar to but not the same as the Viterbi algorithm) can be used to compute the probability of any sequence of phonemes, given a particular sequence of speech spectra. A search strategy is then used to find the most likely sequence of phonemes.

2.3 Phonetic Context

One of the major factors affecting the acoustic realization of a phoneme is the phonetic context. As a result, there have been many suggestions to model the acoustics of units larger than phonemes, such as diphones, demisyllables, syllables, etc. The larger the acoustic unit used, the more its acoustic realization will be independent of any neighboring units. However, even when the acoustic unit is the syllable or the whole word, both ends are still affected considerably by neighboring phonemes. All that can be said is that most of the interior parts (far from neighboring phonetic contexts) are unaffected. Each such proposed unit also has its own associated problems. For instance, if the unit is the syllable, a very significant problem is that in English there are about 10,000 different syllables, and therefore, it is impossible to gather detailed statistics about the likely acoustic realizations of each of them from a reasonable sized data base.

The different acoustic models proposed are, in fact, just trying to model the coarticulation effects of adjacent phonemes on each other. There is not necessarily any significant importance in the unit itself. Therefore we have chosen to return to a model of the acoustics of each phoneme, but to take into account the phonetic context in which it appears.

2.3.1 Triphone Model

As an approximation to modeling the phoneme in all possible phonetic contexts, we have decided to take into account the immediately preceding and following phonemes. We call this a triphone model, although it really only models the middle phoneme of three as a function of the two adjacent phonemes. It is expected that this model should account for almost all acoustic effects that are due to phonetic context. However, as discussed in the preceding section, using a larger unit results in a severe training problem, since there are many such larger units, and therefore no longer enough of each to develop a robust acoustic model. Due to the uneven distribution of any such units, however, there are enough samples of the more commonly occurring triphones. For those triphone contexts that have not occurred a sufficient number of times, we could use the model for the phoneme that depends on the phoneme to the left ("left diphone model") combined with the model that depends on the phoneme to the right ("right diphone model"). For those diphone contexts that have not occurred, we can use the model for the phoneme that is independent of context.

The method automatically uses information about adjacent phonemes only to the extent that it has seen examples of that context, and combines this information with less context-specific models for the phoneme in an optimal way. The specificity/robustness tradeoff is not eliminated, but is controlled by varying the amount of context used and the robustness to optimize performance.

2.3.2 Interpolated Estimation

One solution to the the problem of how to decide between the highly conditioned models and the general models with more training is a procedure called "Interpolated Estimation" [3]. Interpolated Estimation combines the individual pdf's obtained in the different contexts by taking a weighted average. The weights are set to reflect the importance of the different pdf's to the whole model and the degree of confidence one has in each pdf estimate based on the number of observations of that context.

2.4 PDF Estimation

The crucial step, then, is to estimate the conditional probability densities (pdf's) for the spectrum for each phoneme or phoneme sequence.

There are many methods available for estimating pdf's, but the method used most frequently is to assume a multivariate normal distribution. If we assume that each phoneme class can be represented by a normal distribution, and that all the covariance matrices are diagonal with all the variances equal (within each class and across all classes), then the Euclidean distance to each class is proportional to the log of the probability density for that class. Of course these assumptions are <u>not</u> all true. The distributions have different variances, and some features are much more useful than others. Therefore, some systems use a different covariance matrix for each class. Again, the tradeoff between specificity and robustness arises. Since there are fewer samples of each phoneme than of all phonemes, estimating a separate covariance matrix for each phoneme either requires more data, or will result in less robust pdf estimates. Taking this issue one step further, we note that most feature distributions are not typically Gaussian. Several systems, therefore, use non-parametric pdf estimates [4, 5, 6]. Two such estimators are described below.

Given many samples of each distribution in several dimensions, and a feature vector $\mathbf{x}$ from an unknown class, the problem is to estimate the probability density $p(\mathbf{x}|Ph_i)$ at $\mathbf{x}$. The simplest method entails first dividing the space into discrete bins -- typically by the use of a clustering algorithm. Then, for each class, the fraction of samples that fall into each bin determines the probability. For these probability estimates to be robust, we need several samples for each bin that is somewhat likely (the very unlikely bins, which correspond to other phonemes, are usually assigned an arbitrary low probability by padding them with one sample).

This discrete non-parametric pdf estimate has two major advantages: First, the pdf is not assumed to be Gaussian. Second, the computation needed to compute the probability density of a feature vector given all classes requires only that we find the nearest bin to the feature vector once. (This can be done quickly with a binary search if desired.) Then, the identity of the nearest bin is used as an index into the vector of precomputed probabilities for each class. This method requires several orders of magnitude less computation than a Gaussian probability computation.

There is one major problem with this approach. If the number of bins used to represent the space of feature vectors is small, then the pdf will not have enough resolution. For reference, consider the space of LPC vectors divided into 64 clusters. Speech that has been coded to 64 clusters (6 bits) is only barely intelligible. We must conclude that the fine phonetic distinctions have been eliminated. The pdf for any particular phoneme would span only 3 or 4 of the 64 clusters. If, on the other hand, we use about 1000 clusters (as indicated from our experience with coders) we now have a severe robustness problem. We could not hope to have enough data to estimate the probability of each of the 1000 bins for each phoneme. We can pad the bins carefully with an optimal number of fake samples (determined by a deleted estimation procedure), but this won't change the fact that our estimates are still, at best, poor.

Another non-parametric pdf estimation method uses all of the training data directly. Given an unknown $\underline{x}$, we compute the probability density directly from the data. This can be done using a Parzen window to compute the contribution of each training sample at the unknown, or more robustly, using a k-nearest neighbor (KNN) window. First, we find the K nearest data points. (K is typically made proportional to the square root of the number of training samples in the class.) Then, the probability is computed from the volume of the resulting window and the distribution of the neighbors within the window.

Since the number of points within the windows is kept fixed, the variance of the probability estimates is also fixed. The resolution then varies according to the amount of training data in each region. This method does not eliminate the trade-off between specificity and robustness, but at least it controls it in a definable way.

These non-parametric pdf estimators are more sensitive to the amount of training data than the Gaussian pdf estimator. But this sensitivity can be offset by the greater flexibility in the model. The major disadvantage of the more robust KNN pdf approach is the vast amount of storage and computation resources required to compute the distance to all of the training points in each distribution.

The solution that we have adopted takes the best of each method and combines them. The pdf is represented by a vector of probabilities - one for each of a large number of bins. However, the probability at each bin is estimated from as many samples as is deemed necessary to make the estimate robust -- rather than the very small number of samples that happen to fall within the particular bin. The pdf estimator used can be the Gaussian, the KNN estimator, or any other robust estimator. The storage and computation are also minimized.

2.4.1 Acoustic-Phonetic Features

There is often a distinction made between speech recognition systems that use detailed knowledge of the characteristics of speech sounds in order to recognize them, and systems that use only minimal specific information, but rather just measure the speech short term spectrum as a function of time and "grind out" numbers.

At BBN we have been trying to develop experience in using probabilistic methods combined with a detailed understanding of the nature of the variability of phonemes in order to improve the performance of phoneme recognition. In particular, we are attempting to determine the ways in which different phonemes vary in order to guide the probabilistic algorithms to find the optimal pattern recognition strategies. That is, to the extent that knowledge about the nature of speech exists, it can be used in the very powerful formalisms afforded by probabilistic methods. One key element in this effort is that we will be using acoustic-phonetic features, which have been heuristically developed over the years by speech researchers in a probabilistic formalism. We feel that this step is essential to the good performance of probabilistic pattern recognition methods. For example, a complete representation of the speech waveform or the power spectrum contains most of the information that is present in the speech. Theoretically, a probabilistic system, with sufficient training, should be able to develop optimal recognition algorithms. However, the dimensionality of the power spectrum is too high for accurate estimation of probability densities without an unreasonable amount of training data. On the other hand, by identifying a few acoustic-phonetic features, of the type typically used for reading spectrograms, most of the information contained in the speech spectrum can be represented in a much smaller number of dimensions, thus making it possible to develop accurate probabilistic models with a tractable amount of training data. In addition, there is some evidence that features of this type may vary less from speaker to speaker and thus result in systems that are closer to speaker independent.

Below we present the procedure for using acoustic-phonetic features in the recognition process.

1. Identify those phonetic confusions that are most common in the network decoder using the spectral parameters.

2. Using the Acoustic-Phonetic Experiment Facility (APEF), we will determine a set of features that seems to distinguish those phonemes well. These experiments are carried out on a modest sized data base of carefully labeled speech. This would be the same subset of the data base used to derive the initial statistics for the spectral parameter HMM's.

3. The specification of the most useful features are inserted into the network by associating them with the phonemes that are to be distinguished. The probability densities derived using APEF are also inserted as the initial pdf's for these features.

4. The forward backward algorithm is used to jointly train both the spectral pdf's and the new acoustic-phonetic feature pdf's. The feature pdf's can also be made to depend on phonetic context in the same way as the spectral pdf's.

5. Using a procedure similar to that used to the one that determines the weights between different context models (interpolated estimation), determine the optimal weights to use to combine the probabilities computed from spectral parameters and from acoustic-phonetic features.

In the decoder, the feature pdf's will be used in a somewhat different manner than the spectral pdf's. When a phoneme is proposed (during the search for the best phoneme sequence), the decoder first scores the new phoneme using the context-dependent spectral parameter model. Then the newly scored theory is placed back on the stack according to its score. When this theory comes to the top of the stack again it is rescored using the feature pdf's, and again put on the stack. By using this two step procedure, we avoid computing scores due to the feature pdf's unless they will be necessary.

3. SYSTEM IMPLEMENTATION

In this section, we describe in detail our implementation of the system for phonetic recognition of continuous speech. Major topics of interest include data structure design, the training program for automatic estimation of speech parameters, the recognition front end that finds the optimal or near-optimal phoneme sequence for an unknown utterance, and the many software tools developed for debugging and data examination. Various practical issues relating to implementation, including those of computation and storage, are also discussed. Fig. 4 shows the system configuration. Here, the forward-backward algorithm takes training speech (including speech parameters and phonetic transcription) and estimates new system parameters. These parameters are then used by the stack decoder to produce an "optimal" phoneme sequence from an unknown utterance.

3.1 Data Structure

As mentioned earlier, one of the major goals of this research is to incorporate both spectral parameters and acoustic-phonetic features in the Hidden Markov Model (HMM) formalism in our speech recognition system. To this end, we have designed and implemented a HMM data structure that we believe is powerful enough for this purpose. It also has the flexibility to allow us to model phoneme contextual effects.

In Section 2.2 we presented a short introduction to the theory of Markov chains. Here, we will discuss in detail the structure of the specific phonetic HMM that was implemented.

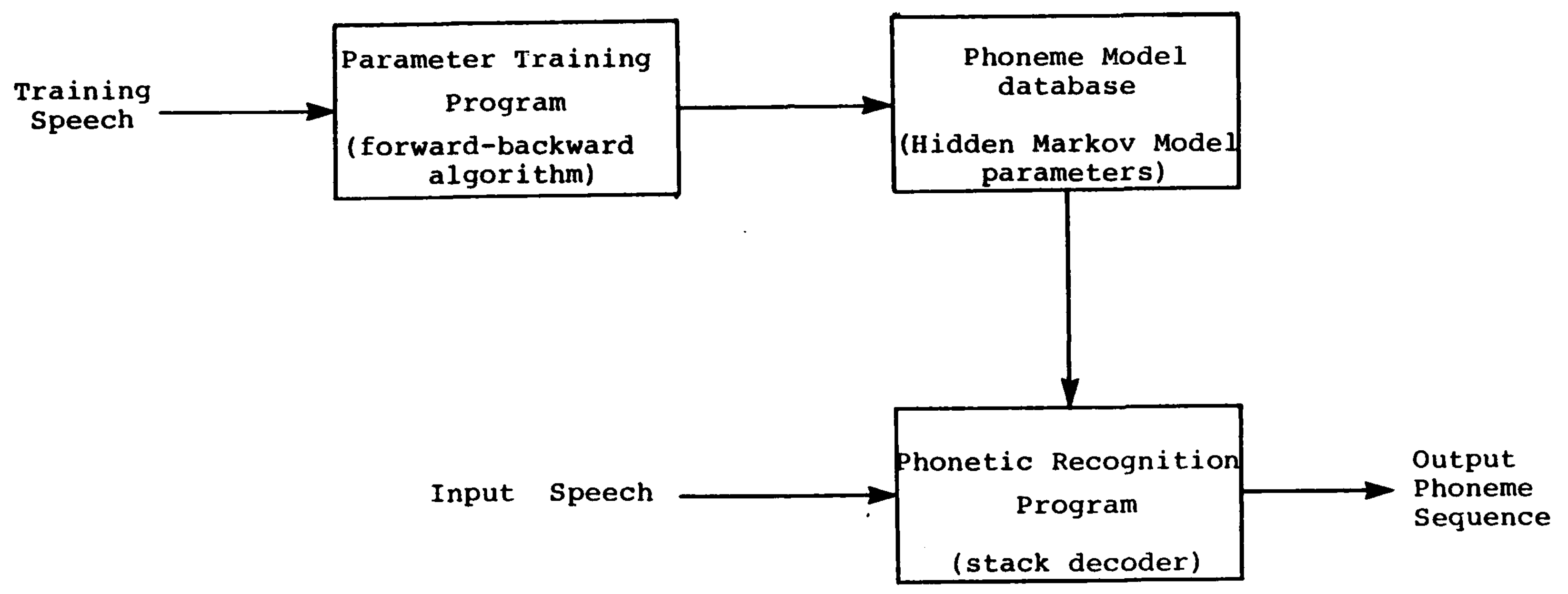

Training Speech
Parameter Training Program (forward-backward algorithm)
Phoneme Model database (Hidden Markov Model parameters)
Input Speech
Phonetic Recognition Program (stack decoder)
Output Phoneme Sequence

FIG. 4. Phonetic Recognition System

Figure 5 shows the model that is actually used in our current system. This model is uniform (has the same structure) across all phonemes. We believe that this same structure is adequate to represent distinct phonemes that differ in both temporal and acoustic characteristics, and yet powerful enough to model the often rapid acoustic transitions within a specific phoneme.

Now let us take a closer look at the model structure. This Markov model, like all Markov chains, is made up of a finite number of nodes (corresponding to states) and arcs (corresponding to transitions). Associated with each state is a probability density function (pdf) for spectral parameters (usually represented by an alphabet of a set of prototypical spectra). The arcs are transitions to and from other states (representing other pdf's). Associated with each transition is a transition probability. Together, these states and the transitions represent a stochastic process that characterizes the acoustic phenomena of a phoneme.

Figure 6 shows the data structure for the Markov model that is implemented. In this implementation of the Markov model, each node contains the following information: an index to a specific pdf (pdf number) associating a probability density function to the node, a list of nodes that can follow this node, and a list of transitions to use in getting to those nodes. As an example, consider node 1. Node 1 is tied to pdf #1, and can be followed by itself (self loop) using transition 3. It can also be followed by node 2, using transition 4, and node 3, using transition 5. Node 2 is tied to pdf #1 (the same pdf as that of node 1), and node 3 is tied to pdf #2. This means that in going to node 2, pdf #1 is used, and to node 3, pdf #2 is used. A node that is tied to pdf #0 (for example, node 10) means that no input

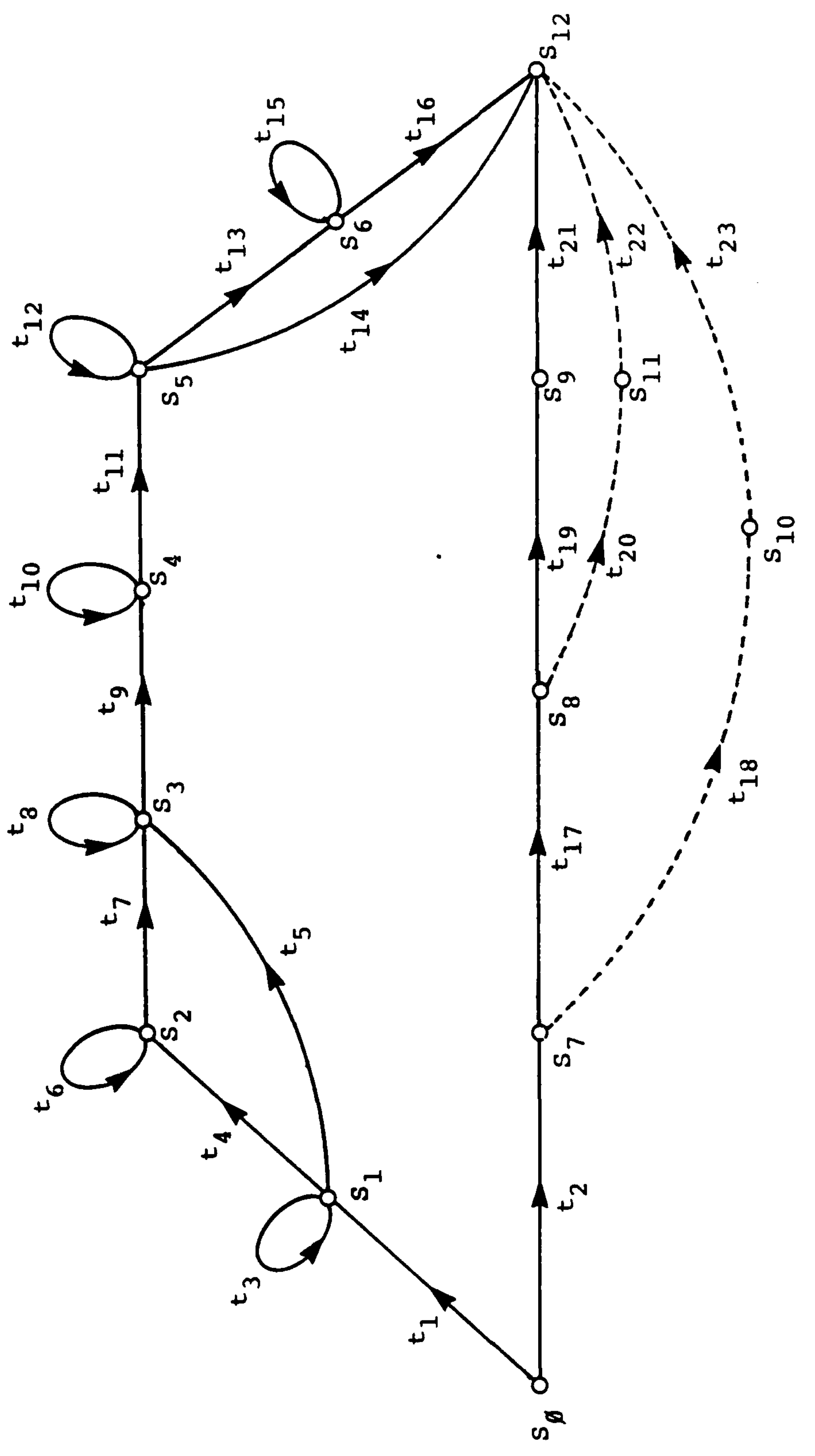


FIG. 5. Phoneme hidden Markov Model currently implemented.

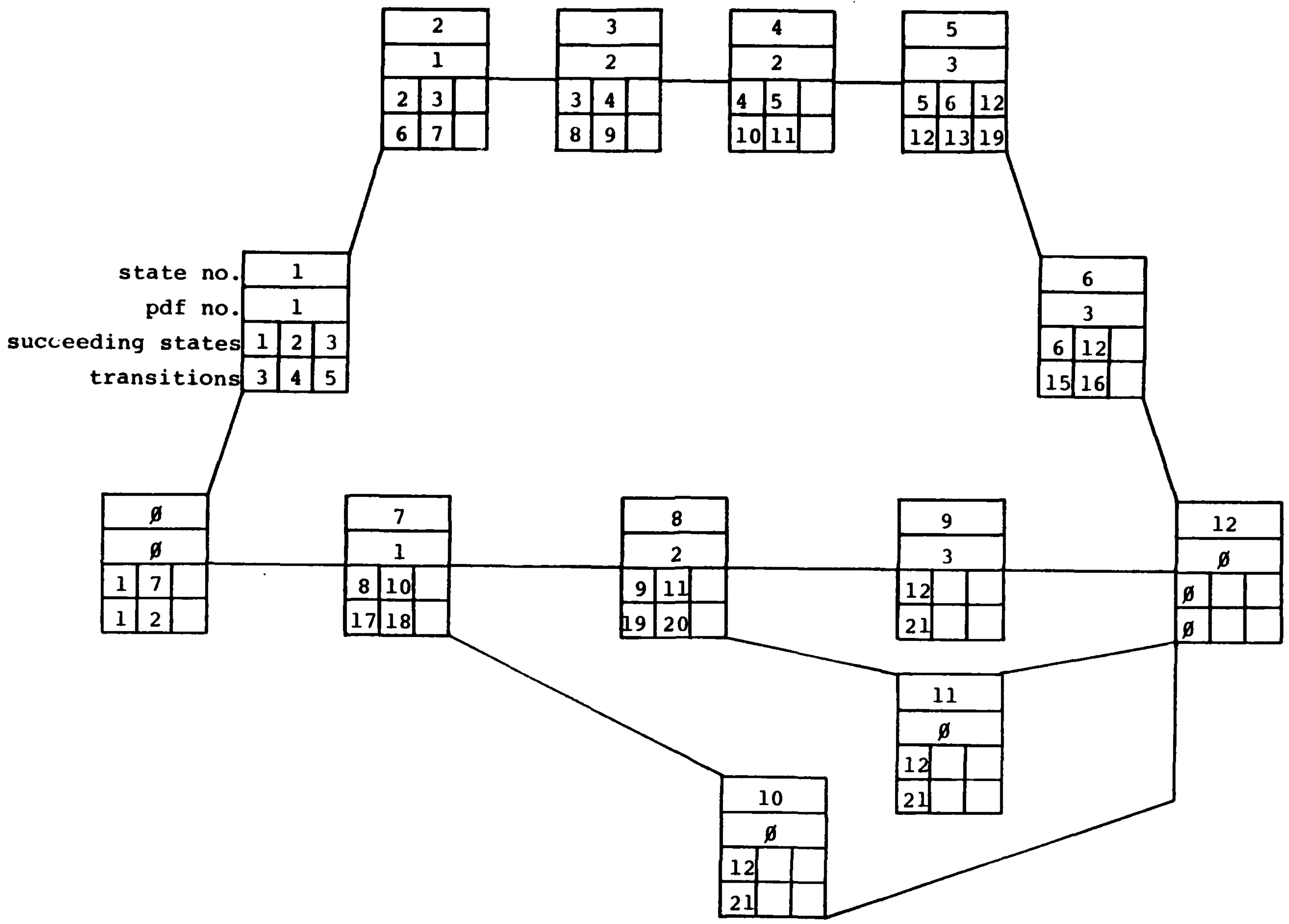


FIG. 6. Implementation of the phoneme HMM structure.

spectrum is used in going to that node. The fact that there are a total of 3 pdf's is rather arbitrary: we can have as many (or as few) as we like. However with too many pdf's one would run into the problem of not having enough training, and with too few the Markov model would in general not be powerful enough to model the acoustic variations within a phoneme. Using three pdf's is a reasonable first try since, in general, each phoneme can be thought of as having 3 parts, the beginning, the middle, and the end. Each pdf can be thought of as modelling each of those parts. Also, the fact that states are tied together (different states use the same pdf) allows us enough flexibility to model pdf durations.

The above structure defines a Markov chain that models the duration of a spectral pdf by having successive states tied together (have the same spectral pdf), and by using self-loops (a state can follow itself). Although this may be powerful enough in most cases to model spectral duration, it does place some constraint on the shape of the duration pdf that is estimated. We have designed our data structure for a Markov model node to allow us to model duration explicitly by using pdf's of duration. Each node, then, has a duration pdf associated with it. To reduce computation and storage, we have chosen to use self-loops for duration, for the time being.

This completes our HMM structure definition. Associated with each of the definition for a phoneme is a probability structure for the pdf's of spectral parameters and transition probabilities. This probability structure is defined in our program to be a single entity that can be referenced by the name of a phoneme. This structure is declared with the following fields:

1. Nocc: number of occurrences of a particular phoneme.

2. Pdf: pdf of spectral parameters referenced by a pdf number (there are a total of 3 such pdf's)

3. Transprob: an array of transition probabilities referenced by a transition number.

It is worthwhile to remember that while there are many such probability structures (one for each phoneme), there is only one Markov model. The HMM defines the structure of the finite state automaton whereas the probability structure provides the parameter values of this automaton.

With the above Markov model and probability structure definitions for a phoneme, we can generalize the structure to that of a triphone, i.e., a phoneme in the context of a particular left phoneme and right phoneme. In our system, a triphone, then, can include many different models that incorporate various levels of context information. In particular, in our current implementation, we have models for (1) phoneme (unconditioned on context), (2) left diphone (phoneme in the context of a left phone), (3) right diphone (phoneme in the context of a right phone), (4) left class (phoneme in the context of a class of phonemes to the left), (5) right class (phoneme in the context of a class of phonemes to the right), (6) left-right class (phoneme in context of a class of phonemes on both sides), and (7) triphone (phoneme in context of a particular left and right phoneme). Identical models of different triphones are tied together. This means that only one copy of a particular context model exists in physical memory. For example, the triphones P[IY]L (for "peel") and P[IY]R (for "peer") all reference the same diphone model for P[IY], and the same phoneme model for [IY], although these are two different triphones. We will describe how these different context models can be combined in some optimal way for scoring in a later section.

3.2 Markov Model Definitions

In this section, we formulate a mathematical definition for the hidden Markov Model (HMM). We shall assume that the underlying Markov chain has N states q_1, $q_2, \ldots q_N$ and the observations are drawn from an alphabet, S, of M prototypical spectra, $s_1, \ldots, s_M$.

The underlying Markov chain can then be specified with a parameter set (Π, A, B), where:

$$\Pi = (\Pi_1, \Pi_2, \ldots, \Pi_N) \qquad (1)$$

is an initial state distribution,

$$A = [a_{ij}], \qquad 1 \le i, j \le N \qquad (2)$$

is a state transition matrix, with a_{ij} being the probability of transiting to state q_j given current state q_i, that is,

$$a_{ij} = \text{prob}(q_j \text{ at } t+1 \mid q_i \text{ at } t) \qquad (3)$$

and

$$B = [b_{jk}], \qquad 1 \le j \le N,\ 1 \le k \le M \qquad (4)$$

is an output symbol probability matrix, with b_{jk} being the probability of observing symbol s_k given current state q_j, that is,

$$b_{jk} = \text{prob}(s_k \mid q_j) \tag{5}$$

Hence, a hidden Markov model, M, is identified with the parameter set (Π, A, B).

To use hidden Markov models to perform speech recognition we must solve two specific problems: (i) computing the probability of an observation sequence, which is used to decode an utterance for recognition purposes, and (ii) estimating the model parameters, for training the models. Both problems are based on a sequence, $\underline{x}$ of observations x_1, $x_2, \ldots, x_T$ where each x_t for $1 \leq t \leq T$ is some $s_k \varepsilon S$.

The training problem is simply that of determining the model parameters A,B, given a training sequence $\underline{x}$ such that the probability prob($\underline{x}$|M) (probability of $\underline{x}$ given the model) is maximized. The recognition problem is the following: given the HMM parameters (A, B) and an observation sequence $\underline{x}$, we want to determine the model M (phoneme sequence) such that the probability Prob(M|$\underline{x}$) is maximized.

3.3 Training Program

The theory of speech parameter training with a hidden markov chain as the underlying model using forward-backward algorithm was first proposed by Baum, and it is known as the Baum-Welch algorithm. Whereas dynamic programming finds the single most likely path through the model network with global consideration of all possible states, the forward-backward algorithm determines the probabilities of all the state sequences that are possible within a given time frame (sum over all paths).

One major advantage in modeling a speech signal as a probabilistic function of a hidden Markov chain is that the HMM allows for automatic training of its parameters. The training method used is based on a bootstrapping technique. A known sentence is given to the system along with a phonetic transcription. The system first aligns the known input speech in all possible ways to the network. Then it uses these alignments to update the network models by changing probability densities. A major advantage of this technique is that the training could be done automatically without human interference, and that it allows the network to be trained on large amounts of speech with relatively little effort. A drawback of this type of algorithms is that they typically require large amounts of computation. However, computation is becoming increasingly faster and cheaper, while human labor is becoming more and more expensive and unavailable. Therefore we still prefer computer-intensive to labor intensive methods.

3.3.1 Forward-backward Algorithm for Markov Model Parameter Estimation

As mentioned in the previous section, the training problem is that of determining the HMM parameters A, B, given a training sequence $\mathbf{x}$ such that prob($\mathbf{x}$|M) is maximized.

One could, in principle, compute prob($\mathbf{x}$|M) by computing the joint probability prob($\mathbf{x}$,$\underline{t}$|M) for each state sequence $\underline{t}$ of length T, and summing over all state sequences. This would be computationally intractable. Instead there is an efficient method for computing prob($\mathbf{x}$|M). We can define a function $\alpha_t(i)$ for $1 \leq t \leq T$ as prob($x_1 x_2 \ldots x_t$, and at state q_i|M). And we have the following recursive relationship for the "forward probabilities".

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]\ b_i(x_{t+1}) \qquad 1 \le t \le T$$
$$b_i(x_{t+1}) \equiv b_{ik} \text{ iff } x_{t+1} = s_k \tag{6}$$

Similarly, we define another function $\beta_t(j)$ = prob(x_t+1, x_t+2,...,$x_t|q_j$ at t, M). We set β_T (j)= 1 and then use the backward recursion

$$\beta_t(j) = \sum_{j=1} a_{ij}b_j(x_{t+1})\beta_{t+1}(j) \tag{7}$$

to compute the "backward probabilities." Then,

$$\text{Prob}(\underline{x}|M) = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j) = \alpha_T(N) \tag{8}$$

for any t, $1 \le t \le T-1$. Equation (6) and (7) formulate what is known as the forward-backward algorithm.

The problem of training a model, that is, finding a global maximum for Prob($\underline{x}$|M), does not have a simple solution. However, Prob($\underline{x}$|M) can be locally maximized. Global maximization involves searching for all local maxima over the entire search space and finding the maximum, whereas a local maximum is any one of those local maxima. We can use the forward and backward probabilities to formulate a solution to the problem of training (finding a local maximum) by parameter estimation. Given some estimates of the parameter values we can compute the expected number of transitions γ_{ij}, from q_i to q_j, conditioned on the observation sequence an it is just

$$\gamma_{ij} = \sum_{t=1}^{T} \alpha_t(i)a_{ij}b_j(x_{t+1})\beta_{t+1}(j)/\alpha_T(N) \tag{9}$$

Then, the expected number of transitions γ_i, out of q_i, given $\underline{x}$, is

$$\gamma_i = \sum_{j=1}^{N} \gamma_{ij} = \sum_{t=1}^{T} \alpha_t(i)\beta_t(i)/\alpha_T(N) \tag{10}$$

The ratio $\frac{\gamma_{ij}}{\gamma_i}$ is then an estimate of the probability of state q_j, given that the previous state was q_i. This ratio may be taken as a new estimate, $\bar{a}_{ij}$, of a_{ij}. That is

$$\bar{a}_{ij} = \frac{\gamma_{ij}}{\gamma_i} = \frac{\sum_{t=1}^{T} \alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \alpha_t(i)\ \beta_t(i)} \tag{11}$$

Likewise, we can get a new estimate of b_{jk} as the frequency of occurrence of s_k in q_j relative to the frequency of occurrence of any symbol in state q_j. Stated in terms of the forward and backward probabilities, we have

$$\bar{b}_{jk} = \frac{\xi_{jk}}{\gamma_i} = \frac{\sum_{t:x_t=s_k} \alpha_t(j)\beta_t(j)/\ \alpha_T(N)}{\sum_{t=1}^{T} \alpha_t(j)\ \beta_t(j)/\ \alpha_T(N)} = \frac{\sum_{t:x_t=s_k} \alpha_t(j)\beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j)\ \beta_t(j)} \tag{12}$$

Both γ_{ij} and ξ_{ij} are what we'll refer to as probabilistic counts.

In reality, instead of a single observation sequence $\underline{x}$, we'll have many sequences (corresponding to many distinct utterances) $\underline{x}^{(1)}$, $\underline{x}^{(2)}, \ldots \underline{x}^{(L)}$ that make up the training set. Then

$$\gamma_{ij}^{(\ell)} = \sum_{t=1}^{T^{(\ell)}-1} \alpha_t(i) a_{ij} b_j(x_{t+1}^{(\ell)}) \beta_{t+1}(j)/\ \alpha_T^{(\ell)}(N) \tag{13}$$

$$\alpha_T^{(\ell)}(N) = \mathrm{Prob}(\underline{x}^{(\ell)}|M)$$

for output sequence $\underline{x}^{(\ell)}$, and for the entire training set,

$$\gamma_{ij} = \sum_{\ell=1}^{L} \gamma_{ij}^{(\ell)} \tag{14}$$

and

$$\bar{a}_{ij} = \frac{\gamma_{ij}}{\gamma_i} = \frac{\sum_{\ell=1}^{L} \gamma_{ij}^{(\ell)}}{\sum_{\ell=1}^{L} \gamma_i^{(\ell)}} \tag{15}$$

Likewise,

$$\xi_{jk} = \sum_{\ell=1}^{L} \xi_{jk}^{(\ell)} = \sum_{\ell=1}^{L} [\sum_{t:x_t=s_k} \alpha_t(j)\beta_t(j)/\alpha_T^{(\ell)}(N)] \tag{16}$$

and

$$\bar{b}_{jk} = \frac{\sum_{\ell=1}^{L} \xi_{jk}^{(\ell)}}{\sum_{\ell=1}^{L} \gamma_i^{(\ell)}} \tag{17}$$

The forward-backward reestimation procedure is as follows:

1. Given some initial parameter values.
2. Use PH(phonetic transcription) - x(observation sequence) pairs to obtain new counts from current parameter values. Do this on all of the training data.
3. Estimate new parameter values from the accumulated counts. Replace current parameter values with new parameter values.
4. If parameter values have converged - stop, otherwise continue with step 2.

Figure 7 provides an illustration of the system training procedure.

3.3.2 Deleted Estimation

In the previous sections we discussed using HMM's of phonemes in context as the basis of our recognition/training

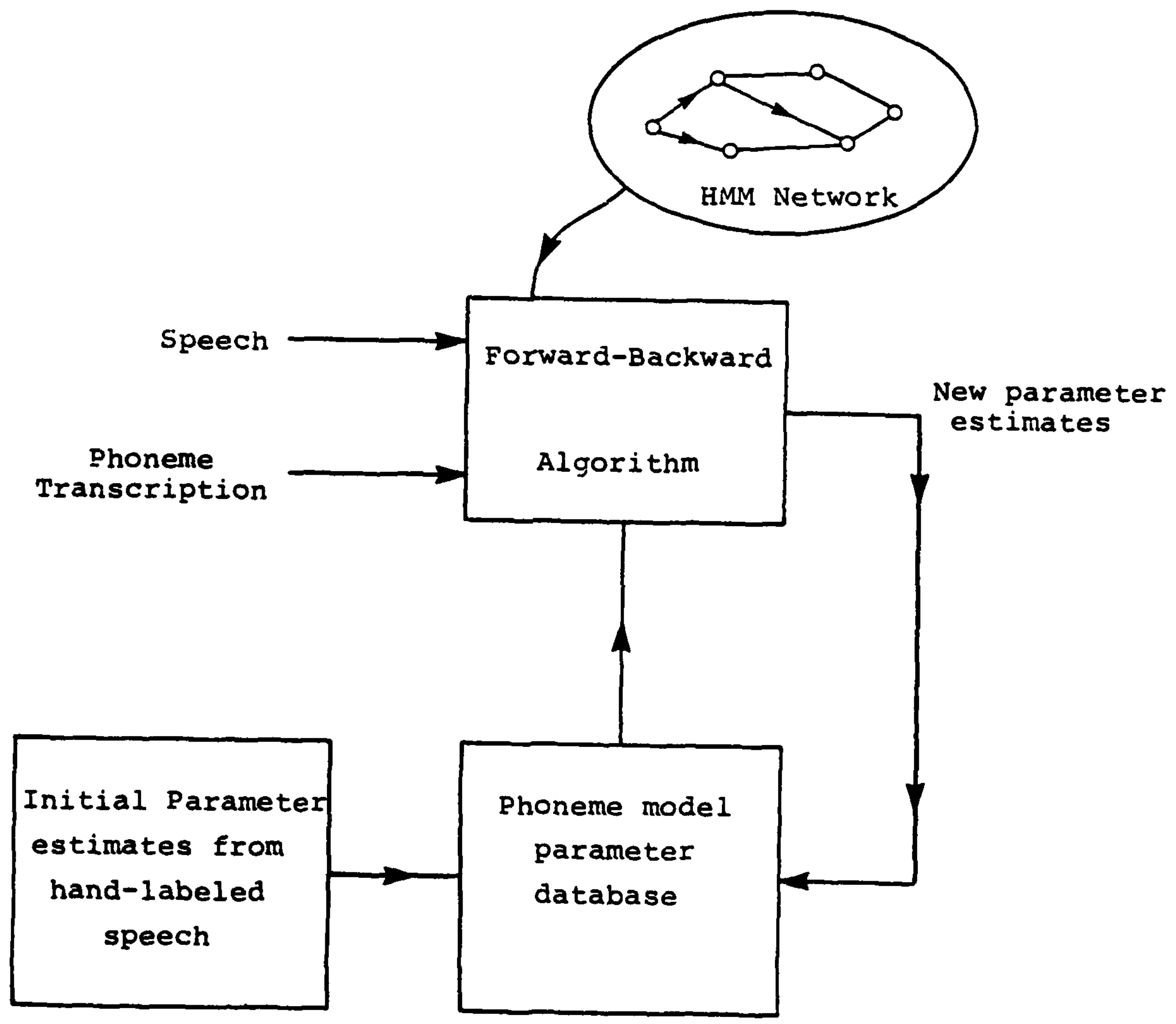

HMM Network
Speech
Phoneme Transcription
Forward-Backward Algorithm
New parameter estimates
Initial Parameter estimates from hand-labeled speech
Phoneme model parameter database

FIG. 7. Forward-Backward Training Algorithm.

algorithm. However, we know that while there will be sufficient training data for some context models, there will not be enough for others. One solution to the dilemma between the desire to use highly conditioned models and the practical restriction of a finite training set is a procedure called "Deleted Estimation". This method combines the individual pdf's of the various contextual models by taking a weighted average. The weights are set to reflect the importance of the different pdf's to the whole model and the degree of confidence one has in each pdf estimate, based on the number of observations of that context. The pdf of the speech parameters x_t at time t, given a particular phoneme with all possible left and right contexts can, then, be estimated from:

$$\begin{aligned} &P(x_t|\text{pdf\#,phoneme, left context, right context}) = \qquad (18) \\ &w_1(\text{pdf\#},n_1)\ p(x_t|\text{pdf\#, phoneme}) + \\ &w_2(\text{pdf\#},n_2)\ p(x_t|\text{pdf\# phoneme, left phoneme class}) + \\ &w_3(\text{pdf\#},n_3)p(x_t|\text{pdf\#,phoneme, right phoneme class}) + \\ &w_4(\text{pdf\#},n_4)p(x_t|\text{pdf\#, phoneme, left phoneme}) + \\ &w_5(\text{pdf\#},n_5)p(x_t|\text{pdf\#, phoneme, right phoneme}) + \\ &w_6(\text{pdf\#},n_6)p(x_t|\text{pdf\#, phoneme, left phoneme class, right phoneme class}) + \\ &w_7(\text{pdf\#},n_7)p(x_t|\text{pdf\#, phoneme, left phoneme, right phoneme}) + \\ &\qquad\vdots \end{aligned}$$

where x_t is the speech parameter at time t, $w_i(\text{pdf\#},n_i)$ is the weight assigned to the pdf of the ith context, and n_i is the number of occurrences of the phoneme in the ith context. We must have

$$\sum_i w_i(\text{pdf\#}, n_i) = 1 \tag{19}$$

for all pdf#.

We have 3 pdf's for each phoneme model in our current implementation, where pdf#1 can be thought of as associated with the left part of the phoneme, pdf#2 the middle, and pdf#3 the right. So, when using pdf#1 in scoring an input spectrum for both training and recognition, the weights should reflect that fact and should favor the left context models more. The pdf in the middle may be determined primarily from the unconditioned context (w_1), but the right end (pdf#3) would depend more on the models in right context, if there are enough samples.

The procedure for automatic training of the w_i's is as mentioned before, deleted estimation. In practice, this algorithm is combined with a method of dividing training data into smaller blocks and training on alternate blocks. This method is called "jackknifing". In this procedure, the data is first divided into N (usually 4) blocks that are equal in size. Given some initial pdf estimates and weight estimates, first we train the pdf's (using forward-backward algorithm) on a subset of the training set containing N-1 blocks of the data until the pdf's converges to obtain new pdf's; then train the remaining block with the new pdf's on the weights (using forward-backward algorithm) and obtain weight counts (to see how well the trained pdf's model the unseen data block). This is the deleted estimation procedure. Perform this procedure on all possible permutations of the N blocks of data and obtain a new set of weight estimates from the weight counts accumulated over all the pieces on which deleted estimation was performed. This is a single pass of the jackknifing algorithm. With the new estimate of the weights, we start the entire jackknifing procedures over again until the pdf's converge. This might take from a few passes to several iterations, depending on the size of the training set. With the final weight estimates, we now train the pdf's on the entire training set until they converge. This entire process might take as many as tens of iterations of the forward-backward algorithm. Needless to say, deleted estimation combined with jackknifing is expensive computationally. In the following sections, we'll look at ways of speeding up the forward-backward algorithm.

3.3.3 Implementation Issues

Initially, we had designed the data structure so that all of the system parameters (probability densities) would be stored in memory to minimize I/O and speed up system running time. However, to do so would require 40-80 Megabytes of main memory (more than normally available). Careful examination of the algorithm revealed that a relatively large amount of computation is performed using a fraction of the data. Therefore, the programs have been altered to store the phoneme spectral pdf's on a file. The pdf's are read in as they are needed. Since this amount of memory is small (for a sentence), for a small number of phoneme context models, and the computations performed are significant, the overhead involved is small. This phenomenon may no longer be true, however, if we use the full range of phoneme context models available. We may have to examine more closely I/O vs computational issues and to look for ways to speed up the system further. Currently, the forward-backward algorithm reads in all the pdf's for all the phonemes in an utterance as the utterance is being trained.

The computation of the probabilistic counts in the forward-backward algorithm requires the forward probabilities α_t as well as the backward probabilities β_t for all time t. One method of computing the counts is to compute the α's and the β's separately for all time t and then compute the counts. Storing both α's and the β's for all input frames for a typical training utterance requires a large amount of memory. To save on storage, and possibly computation we have in our system implemented the following: the α's are computed first in a forward pass for all time; then in the backward pass, the β's are computed a single frame at a time, along with the counts. As we go back in time, the β's for the previous time is updated to become the current β's. This is done for all input time frames. This way, only two "rows" of β's need to be stored at any time. This practically eliminates the storage needs of the backward probabilities.

Our current implementation provides two algorithms for speeding up the forward-backward algorithm. One is to use phoneme segmentation to nail down phoneme boundaries and allow $\pm$ N frames from the proposed segmentation in the forward-backward iteration. The second method uses windowing to narrow down the range of phonemes for α and β computation. At each input time, in the loop of the forward-backward algorithm, the program computes the phoneme in which the maximum α occurred, and then use this information to limit the computation at the next time frame to a few phonemes bordering the phoneme of maximum α. This speeds up the algorithm by (1) reducing the computation of α's and β's, and (2) reducing the overhead of memory page faulting associated with accessing α's in a large array. An order of magnitude savings in computation has been realized.

In our implementation we are also concerned with the problem of numerical underflow caused by the fact that small numbers like α (they are probabilities) are multiplied together over may time frames. We have solved this problem by normalizing the α's (and the β's) by the maximum at each time frame and keeping track of this normalization factor over time. Since a pdf count involves dividing a small number ($\alpha.\beta$) by a slightly larger number $\alpha_T(N)$, the "smallness" often cancels out, resulting in a relatively large number. All of this is taken care of properly.

Currently, the forward-backward program has the following capabilities:

1. Allows the user to specify the number of phoneme contextual models to use to allow for a certain degree of control so different experiments can be tried.

2. The user can specify a pdf file containing the pdf structures to use in the training. The program checks for consistency of the file with program data declarations (for example, the number of spectral clusters used and declared must be the same).

3. The user can also specify a count file for accumulating probabilistic counts for training.

4. The pdf's in two pdf files can be combined with user-specified weighting.

5. Allows forward-backward training of the pdf's.

6. Allows forward-backward training of the weights on different context models.

7. Include I/O for reading and writing of pdf's, weights, pdf counts, and weight counts.

8. With the three functional capabilities above (5-7), the user can perform deleted estimation and jackknifing.

9. Allows estimation of new pdf's in a pdf file from the counts in a count file. Provide pdf padding and smoothing capabilities.

10. Several capabilities for debugging: allows user to examine various pdf and count structures, as well as the weights. Can perform diagnostic test on the pdf structures in a pdf file.

Since all the functions have been implemented on a modular basis, adding a new function can be done with ease. The fact that the program is controlled by a top level command interpreter provides smooth function integration.

3.4 Phonetic Recognition Program

The phonetic recognition program in our phonetic recognition system is called the stack decoder because (i) it attempts to "decode" an unknown utterance by walking a HMM network representation of speech; and (ii) it uses a stack structure to order competing theories.

Network-decoding algorithms in general do not segment the speech into phonetic units before labeling. Instead, they use a dynamic programming type of algorithm that considers all possible segmentations of the input. Associated with each segmentation is a best labeling and a corresponding score. The algorithm then chooses the segmentation and labeling that together result in the best score.

3.4.1 Stack Decoder

In section 3.3, we provided a solution to the problem of HMM parameter estimation. In this section, we will address the problem of utterance classification for recognition. Given the phoneme Markov model parameters and an observation sequence $\underline{x}$, the problem is to find a phoneme sequence $\underline{PH}$ such that the aposteri probability $\mathrm{Prob}(\underline{PH}|\underline{x})$ is maximized. Baye's rule gives us

$$\mathrm{Prob}(\underline{PH}|\underline{x}) = \frac{P(\underline{x}|\underline{PH})\cdot P(\underline{PH})}{P(\underline{x})} \tag{20}$$

$P(\underline{x})$ is the unconditioned probability of the observation sequence and is independent of the phoneme sequence hypothesis $\underline{PH}$. $P(\underline{PH})$ is the unconditioned probability of the phoneme sequence, and $P(\underline{x}|\underline{PH})$ is the probability of the observation sequence given the phoneme. The problem is then to find $\underline{PH}$ such that the joint probability

$$P(\underline{PH},\underline{x}) = P(\underline{x}|\underline{PH})\cdot P(\underline{PH}) \tag{21}$$

is maximized.

In theory, the solution can be found by doing an exhaustive search by trying all possible phoneme sequences (of all possible alignments). Obviously, this is computationally infeasible. Instead, the stack decoder that is implemented employs a best-first strategy, using a stack structure to order competing theories using a particular scoring paradigm. The type of computation for the decoder is similar to that for the forward probabilities α's in the forward-backward algorithm, where the scores for all possible paths that end in a particular state are added together.

In the forward-backward algorithm, the α's are computed with the phonemes of an entire training sentence concatenated together, so that all warpings of the speech input through the concatenated model are allowed. In the decoder, a single phoneme is proposed and scored at a time. This is called the incremental match. If during decoding of a theory $PH_1,\ldots PH_i$ is being extended by many i+1th phonemes, PH_{i+1}, PH'_{i+1}, $PH''_{i+1},\ldots,$ then only the incremental match calculation beyond PH_i need to be done for each new phoneme. This saves on computation as well as storage when scoring a new theory. Each theory then actually includes a list of possible ending time and corresponding scores. See Fig. 8.

3.4.2 Algorithm development

a. Search Strategy

The stack decoder implemented in our system uses a best-first strategy to find the best-scoring phoneme sequence. This search strategy differs from any other search strategies by the order in which theory nodes are expanded. In the breadth-first search, nodes are expanded in the order they are generated. In the depth-first search, the most recently generated nodes are expanded first. And the uniform-cost best-first search expands the highest scoring theory node first. These are all blind search methods which do not take into account how close a theory is in getting to the end.

What the decoder uses then is heuristic search combining the best-first search strategy with an evaluation function. This evaluation function scores a theory by combining the score so far with the expected score of the theory in getting to the end of an utterance. Without an evaluation function, longer theories will

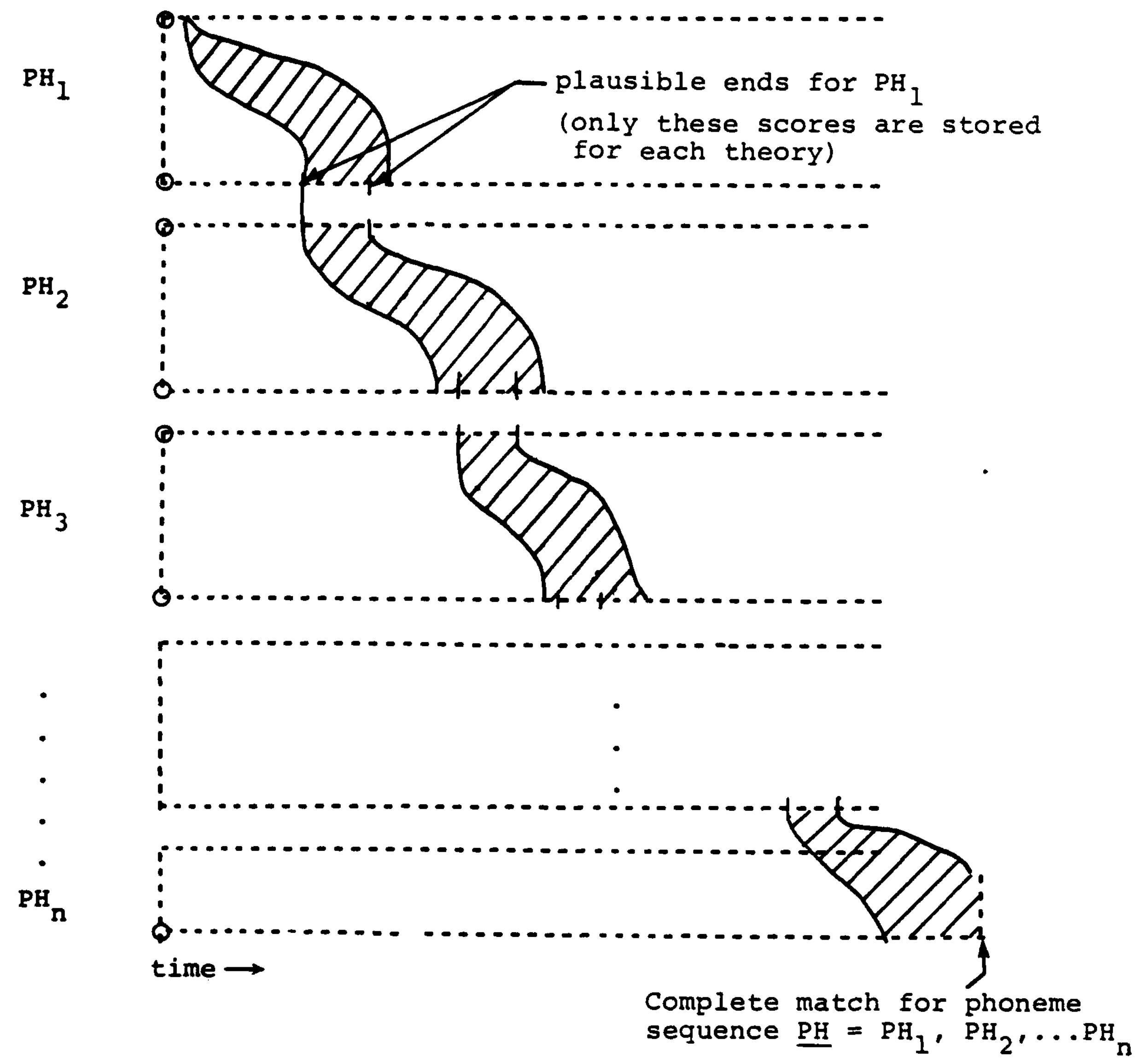

PH$_1$
plausible ends for PH$_1$
(only these scores are stored for each theory)
PH$_2$
PH$_3$
PH$_n$
time →
Complete match for phoneme sequence $\underline{PH}$ = PH$_1$, PH$_2$,...PH$_n$

FIG. 8. Incremental matching procedure for a theory.

always get lower score, so that eventually all theories will have be evaluated, a phonemenon called thrashing. This way the search essentially becomes exhaustive, and thus computationally expensive.

What we like to do is to allow the optimal path to stay near the top of the list of theories to expand. What we would like is an evaluation function that increased on the optimal path (and hopefully decreased on non-optimal paths).

What is used then, is an evaluation function:

$$\Lambda(\underline{PH}) = Pr(\underline{PH}) \cdot \frac{\sum_t Prob(\underline{x}_t|\underline{PH})}{p^*(\underline{x}_t)\,\delta^t} \tag{22}$$

The numerator is simply the joint probability prob($\underline{PH}$,$\underline{x}$). $p^*(\underline{x}_t)$ is equivalent to unconditioned probability of the input up to time t, and we estimate it from a first-order Markov probability of the input:

$$p^*(\underline{x}_t) \approx p(\underline{x}_t|\underline{x}_{t-1}) \tag{23}$$

Along the optimal path,

$$\frac{Prob(\underline{x}_t|PH)}{p^*(\underline{x}_t)} \approx 1 \tag{24}$$

and along non-optimal paths,

$$\frac{Prob(\underline{x}_t|PH)}{p^*(\underline{x}_t)} << 1 \tag{25}$$

See Fig. 9.

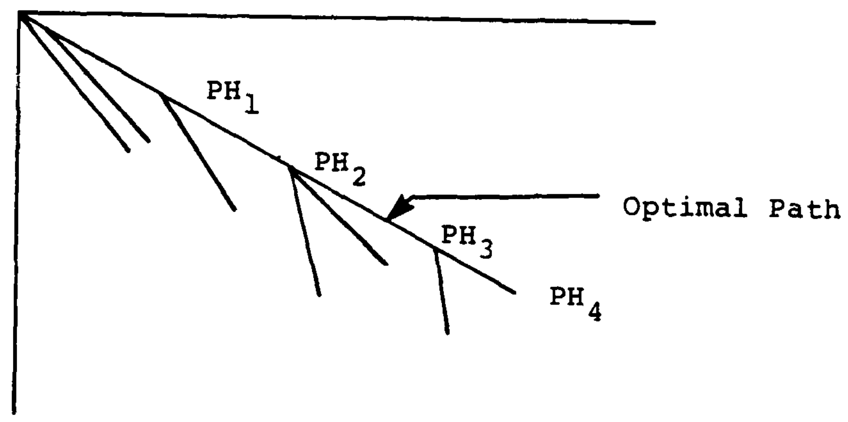


FIG 9a. Best-first search. The score for the optimal path decreases with the length of the theory.

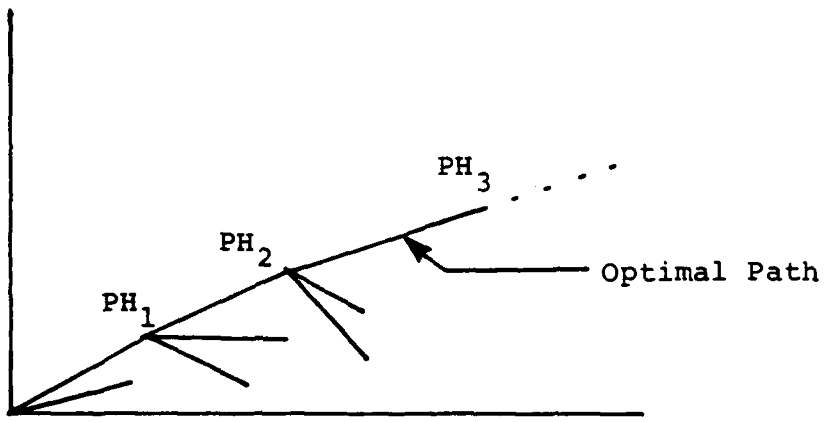


FIG. 9b. Best-first search with an evaluation function. The score for the optimal path increases slightly with the length of the theory.

δ is a constant that allows us some control over how the optimal path should behave. Choosing δ carefully, we can make $\Lambda(\underline{PH})$ go up slowly along the optimal path. If we multiply $\Lambda(\underline{PH})$ by $p^*(\underline{x}_T)\delta^T$, we get

$$\Pr(\underline{PH}) \cdot \sum_t \text{Prob}(\underline{x}_t|\underline{PH}) \cdot \frac{p^*(\underline{x}_T)\delta^T}{p^*(\underline{x}_t)\delta^t}$$

Now, $p^*(\underline{x}_T)\delta^T$ is the expected value for the entire sentence; and $p^*(\underline{x}_t)\delta^t$ is expected value for sentence up to x_t. So $p^*(\underline{x}_T)\delta^T/p^*(\underline{x}_t)\delta^t$ is the expected value for remainder of the sentence.

b. Stopping condition for a theory

When a theory $\underline{PH}_i$ is taken off the top of stack and extended by a new phoneme hypothesis PH_{i+1}, we need to know when the incremental matching procedure can be terminated. Hence, a stopping condition has to be met.

This is done by looking at the score of the terminal state $\alpha_t(s_N)$ for phoneme PH_{i+1} as a function of time t and detecting a sharp fall in the scores. Actually, $\sum_j \alpha_t(s_j)$ is the score used for stopping (a more robust score). In our program, decoding is terminated when the change in the average log slope of $\sum_j \alpha_t(s_j)$ has exceeded a certain threshold. The 15 terminal state scores $(\delta_t(S_N), t = t_1, t_2, \ldots, t_{15})$ centering around the maximum $\alpha_t(s_N)$ is kept for the theory $PH_1\ PH_2 \ldots PH_{i+1}$. This gives a tremendous saving in storage. See Fig. 10.

c. Theory collision

During decoding, whenever two theories end in the same state

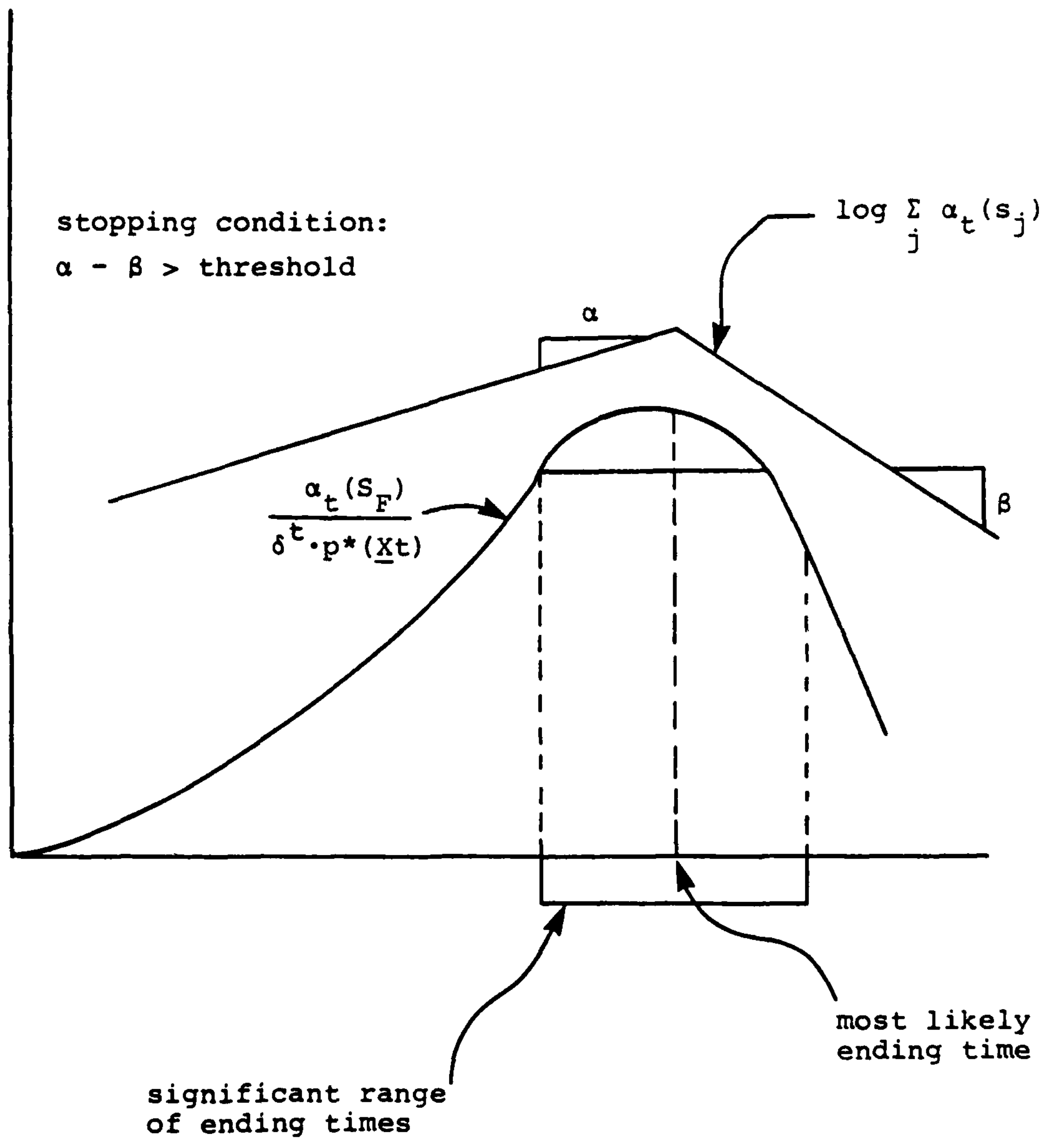

stopping condition:
$\alpha - \beta >$ threshold
$\log \sum_j \alpha_t(s_j)$
α
β
$\frac{\alpha_t(S_F)}{\delta^t \cdot p^*(\underline{X}t)}$
most likely ending time
significant range of ending times

FIG. 10. Stopping condition for a theory.

(as defined by the amount of phoneme context models used), we would like to keep only one of them on stack and eliminate the other one. For example, consider the two paths, $\underline{PH} = PH_1,...PH_n$ and $\underline{\widetilde{PH}} = \widetilde{PH}_1,...\widetilde{PH}_m$, both have equivalent ending time. Suppose that the full context model of a phoneme is used (triphone), and that the phoneme model states for the two paths is the same (so that $PH_{n-1} = \widetilde{PH}_{m-1}$ and $PH_n = \widetilde{PH}_m$), and $\Lambda(\underline{PH}) \geq \Lambda(\underline{\widetilde{PH}})$. Then any extension of $\underline{\widetilde{PH}}$ will be inferior to the corresponding extension of $\underline{PH}$. The theory $\underline{\widetilde{PH}}$ can therefore be eliminated.

3.4.3 Implementation Issues

a. Theory extension control strategy

A stack decoder using best-first search strategy in general performs the following logical steps:

1. Initialize the stack with a null theory ($\underline{PH}=\emptyset$) and giving it a score of unity.
2. Take the best theory off the top of the stack. If end of the utterance, stop. This is the optimal answer. Else continue with Step 3.
3. Extend the best theory and place it on stack. Go to Step 2.

Fig. 11 illustrates the above steps with a flow chart.

In our implementation, step 2 is actually more complicated than what is described. Due to the fact that we use phoneme right context models, a straightforward scoring strategy would be to score a phoneme (α computation) in all possible extensions of the right context. In general, this requires a large amount of a computation. We can accomplish essentially the same thing, but

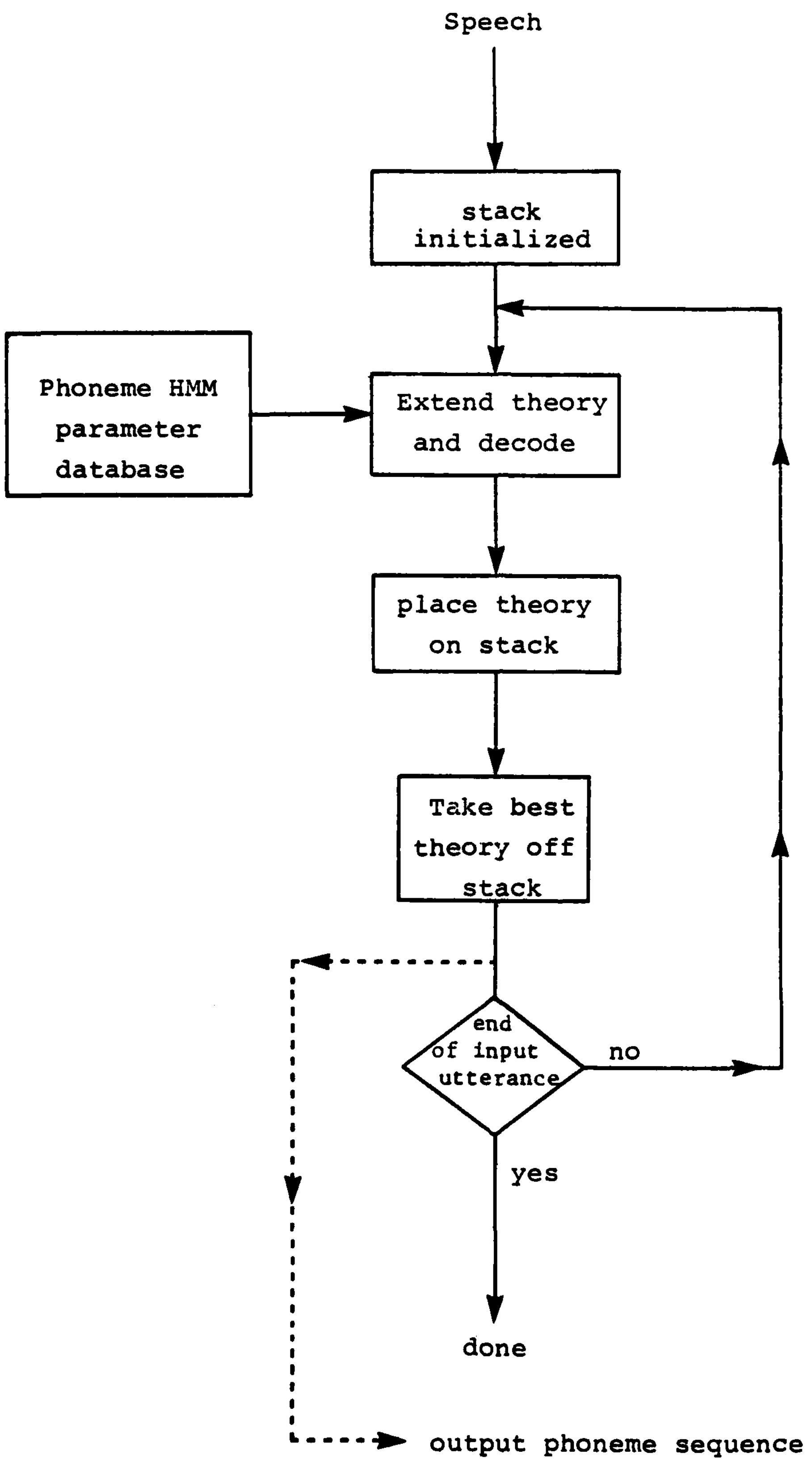


FIG. 11. Stack Decoder Algorithm.

with a tremendous saving in computation by doing the following: we keep two kinds of theories, "fake" theories, and "real" theories, on stack. Fake theories are theories that are scored with phoneme transition probabilities (no acoustic scores). Real theories are acoustically scored, using only the left context models. Whenever a real theory comes to the top of the stack, it is extended to fake theories by all the possible phonemes that could follow. The scores for these fake theories are the scores for the real theory multiplied by the phoneme transition probability. Only when a fake theory comes to the top of the stack again is it acoustically scored. When a fake theory comes to the top of the stack, it is actually scored twice. First it is scored using the full range of left and right context models (since it knows about a right phoneme). Then the theory is extended so that the right phoneme is scored using left context models only. The scored theory is then placed on the stack. The benefit of the above algorithm is twofold. First, by giving the fake theories the benefit of the doubt (not scoring them acoustically), only those theories with a reasonable a priori probability will come to the top of the stack and become real theories. Second, by scoring using only the left context models, we can get a pretty good idea of how well the acoustics match without going to the full contextual effects, which would require hypothesizing many more theories (one for each right context) and scoring them. Only those theories that have reasonable acoustic scores based on a reduced model set will ever be considered again. Together, these two methods for extending and scoring theories provide large savings in computation with little loss in performance. See Fig. 12 for an illustration.

b. Theory tree and stack structures

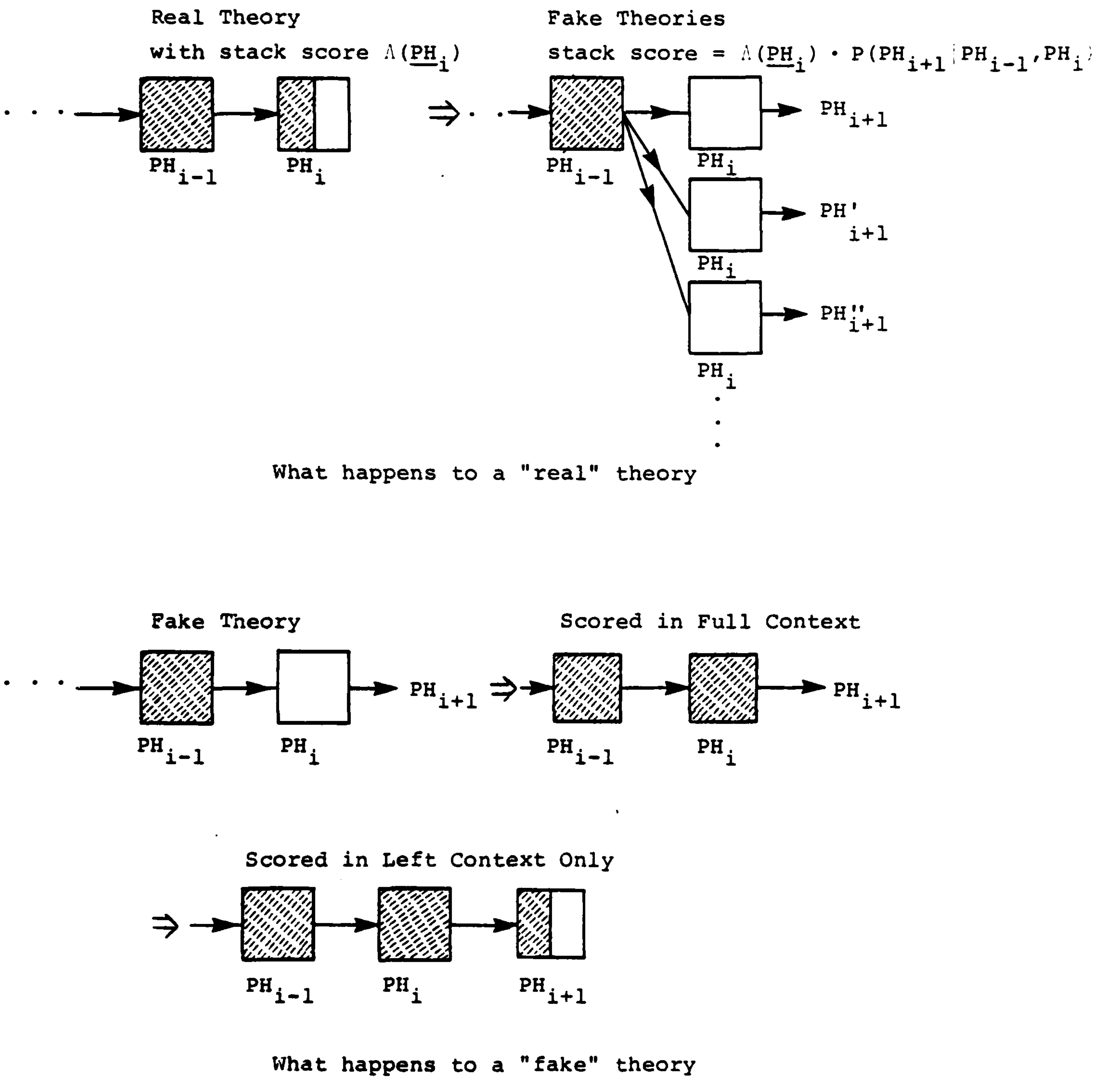

Real Theory
with stack score $\Lambda(\underline{PH}_i)$
Fake Theories
stack score = $\Lambda(\underline{PH}_i) \cdot P(PH_{i+1} | PH_{i-1}, PH_i)$
PH_{i-1}
PH_i
PH_{i+1}
PH'_{i+1}
PH''_{i+1}
What happens to a "real" theory
Fake Theory
Scored in Full Context
Scored in Left Context Only
What happens to a "fake" theory

FIG. 12. Theory extension control strategy.

The theory tree (representing theory propagation) in our decoding network is made up of theory nodes (representing phoneme sequences) that are linked together. Each node in this search tree, declared as a single structure in our program, contains the following information:

phone:	phoneme number identifying the node.
nextphone:	phoneme number identifying the following context.
parent:	pointer to the parent node for trace back.
son:	pointer to a son node.
brother:	pointer to a brother node. All brothers have the same parent.
lefttime:	starting time of when the scores for the terminal state of the theory is kept (there are 15 such scores for a theory, starting from left time).
time_max:	time of the occurrence of the maximum score for the terminal state.
score:	a list of terminal state scores ($\alpha_t(S_N)$, $t = t_1, t_2, \ldots t_{15}$) for the theory.
entry_on_stack:	pointer to the location of the theory on the stack.

Similarly, we have a structure declared for the stack. The stack is declared as an array, and each entry of this array contains a score to order the theories and a pointer to the theory tree to identify the stack entry with a particular theory hypothesis.

In our implementation, the stack is organized as a heap so that the search time for the best theory increases proportionally only to the log of the total number of theories (vs a linear search time for linear search).

c. Program Functional Capabilities

The current implementation of the stack decoder allows a wide range of functional capabilities. It allows the user

1. to specify the number of phoneme context models used.
2. to specify a pdf file that contain the Markov model parameters to be used in decoding.
3. to decode an utterance and write out the decoded phoneme sequence.
4. to constrain recognition to a user-specified phoneme transcription to obtain a segmentation.
5. I/O capabilities to read in pdf weights and other tables.
6. to set various decoding parameters: (1) set debugging flag for interactive debugging; (2) set a flag to use a version of short-fall density scoring; (3) allows a limited breadth-first search by giving all fake theories a benefit, so that all will be scored acoustically; (4) specify an offset to raise the theory scores; (5) allows the options of using log slope difference or probability threshold as stopping conditions, and to set corresponding parameters and thresholds; and (6) allow use of theory collision algorithm.
7. to examine the pdf structures for a specific triphone, as well as the weights for the pdf models.

When decoding an utterance in debug mode, the program allows the user to interactively examine and if he wishes, to control the decoding process. Specifically, the user can

1. look at N best theories on stack. For each theory the program will type out the phoneme string, their stack scores, and the location on stack.
2. examine the 15 terminal scores of a theory by specifying a stack position.
3. look at the time alignment (phoneme boundaries) of a theory.
4. examine all theories ending with a particular phoneme sequence.
5. redo a theory that seems confusing, giving the user a chance to run it the system debugger.
6. restrict the decoding to a particular phoneme sequence.

Like the forward-backward program, the decoder is implemented modularly using command interpreter whenever possible. This permits easy integration of new function modules and creates a user-friendly programming and debugging environment.

4. EXPERIMENTS

This section describes a set of experiments conducted to get a preliminary idea for how well our initial phonetic recognition system performs. First, we will provide a brief description of our database. We will then describe the experimental procedure, actual system configuration and parameter values for this experiment, the type of signal processing that was done, and the performance that was achieved.

4.1 Database

Currently our database contains 110 different sentences (about 5 minutes of speech) by a single speaker, carefully hand-labeled. This means that the researcher has indicated for each utterance what the phonemes are and where each begins and ends. The entire set of these 110 utterances have been employed for our experiment. Out of this set of 110 utterances we used a set of 100 utterances for training, and the remaining set of 10 utterances for testing. In data gathering, speech was first recorded with a microphone, and sampled at 20 kHz. We are currently collecting an additional half an hour of speech to be used in the immediate future to further our research goals.

4.2 Markov Model Structure

In our system, for this experiment, each phoneme is represented by a Markov model of spectral states and transitions to and from these states. The spectral pdf's associated with each state are dependent on the location within the phoneme (See Section 3.1). The duration statistics of the transitions are modeled explicitly by pdf's of duration, rather than by the first-order Markov probability of repeating the same state (self-loop). The phoneme model parameters depend only on the phoneme (rather than on any phonetic context, such as diphone or triphone). This model is, in many respects, simpler than the full system being developed. Therefore, initial performance was expected to be very poor, but this simple system has served as an initial debugging testbed.

4.3 Signal Processing

For signal processing, LPC analysis is performed to extract 14 log-area-ratio (LAR) coefficients once every 10 ms on the entire signal spectrum. This results in a 14-dimensional parameter vector for each frame of speech. These LAR vectors are then quantized to one of 50 spectral clusters. These clusters are prototypical spectra which span the entire spectral space, and were derived with a clustering algorithm from 1.5 minutes of speech. The probability density functions of these spectral clusters are estimated and used in scoring for phonetic recognition.

4.4 Recognition Performance

In each experiment, two tests are performed. In the first test 10 test sentences outside the training set are used. This set is called test set A. In the second tests, the test sentences are taken from a set of 10 sentences that are within the training set. This set is called test set B.

Figure 13 provides a summary of the results of the experiments. It gives the recognition error rate as a function of the training set size. The dotted line represents the error rate for the experiments where the test set is outside the training set (test set A). The solid line is the result for the experiments where the test set is within the training set (test set B).

Starting from the left of the figure, points 1 and 1' are the results of the experiment where spectral pdf's are based strictly on initial statistics (hand-labeled data) with no forward-backward training. The initial statistics are derived automatically by a program that takes the labeled speech and computes a matrix of phoneme-spectral cluster counts, and then normalizes the counts to obtain the initial pdf's. For each phoneme, pdf's #1, #2, and #3 are set equal to the pdf obtained from the initial statistics for that phoneme (see section 3.1) The duration pdf's are user-specified exponential functions. The error rate for test set A (outside training) is 64%, and that for test set B (inside training) is 62%.

Points 2 and 2' show the error rates when forward-backward training is done on a single sentence (within set B) starting from the pdf's based on the initial statistics from the 100 sentences. The error rate for test set A (outside training) was very high at 82%. This results is not surprising given that a very small training set of one sentence is likely to be very different from a random set of test sentences. The error rate for testing on the same sentence was 0%, as it should be.

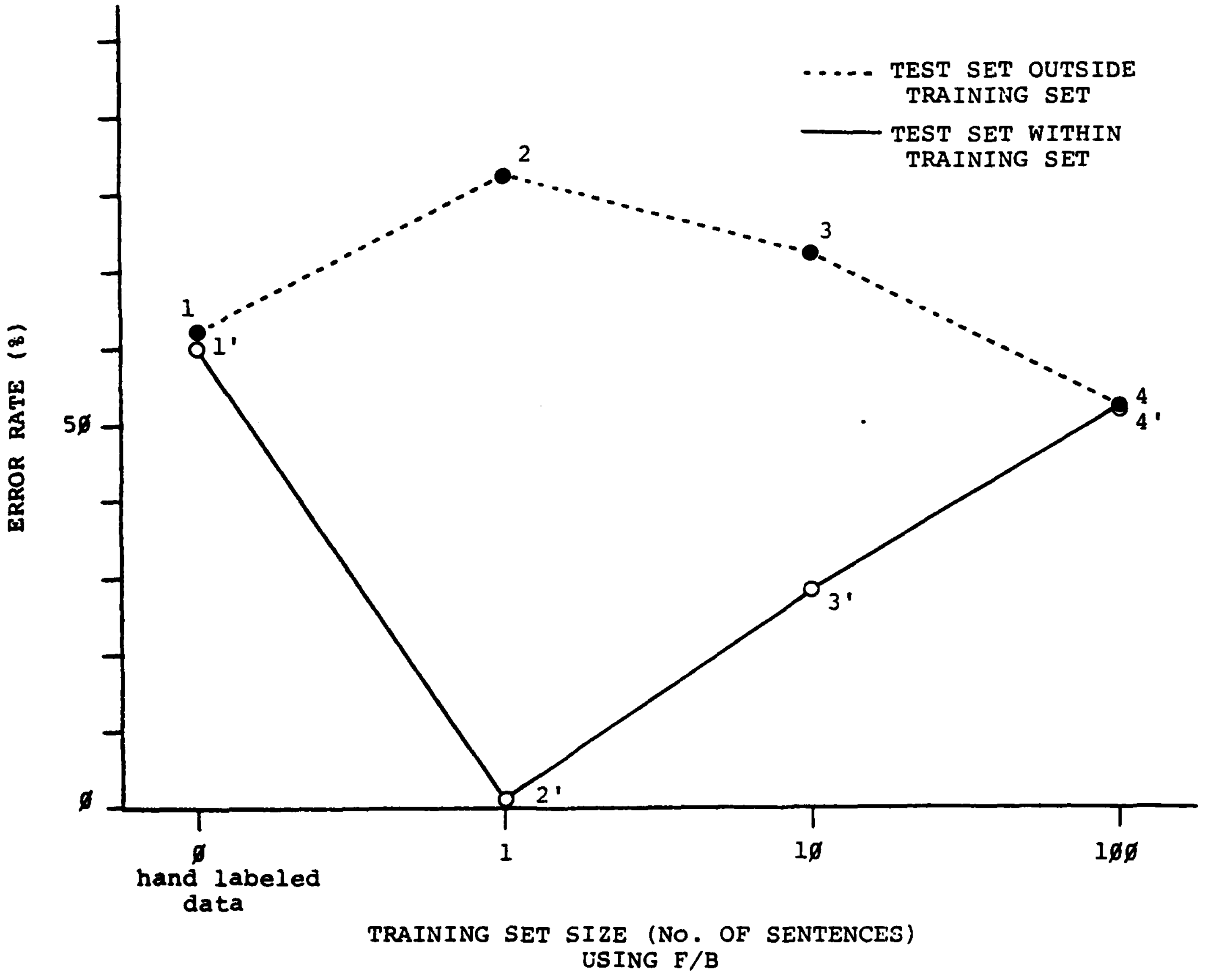


FIG. 13. Phonetic recognition performance results.

Points 3 and 3' show the results of the experiment where forward-backward training is done on a set of 10 sentences (set B) starting with the initial statistics of experiment 1. The error rate for test set A (outside training) is quite high, at 70%, and that for test set B (within training) is 28%.

Finally, points 4 and 4' are the results of the experiment where forward-backward training is performed on the entire set of 100 sentences. Again, two tests are done. In the first case, test set A (outside training) is used, and in the second, test set B (within training) is used. The error rates obtained for both cases turned out to be very close, at 55%. However, this coincidence may have been an artifact of the random differences between test sets.

4.5 Conclusion

From the results of this experiment, we can draw some conclusions as to what needs to be done to improve the performance of our current phonetic recognition system. The conclusions are as follows: (1) A 50-cluster spectral representation of speech is clearly not enough; (2) a more appropriate speech analysis method is needed; and (3) phoneme context-dependent information must be used. Below we describe each of these points in more detail.

First of all, 50-cluster representation of speech does not provide enough spectral resolution to distinguish among the different phonemes, since there are as many phonemes as there are clusters. Much research is needed to find optimal number of clusters as well as cluster pdf smoothing algorithms to achieve both resolution and robustness for speech recognition purposes.

Secondly, computing 14 LAR's on the entire 10 kHz bandwidth with no spectral scaling is not appropriate since most of the useful information in the spectra reside in the lower few kilohertz. We propose using some warping function on the spectra so that more attention is paid to those spectral bands that contain more useful information.

Lastly, using unconditioned models of phonemes for recognition yields too much variability in the pdf's that model the phonemes (especially near the phoneme boundaries). Use of phonetic context (by using phoneme models that incorporate effects of neighboring phonemes) should solve this problem and allow us to make finer phonetic distinctions.

In the coming year, our research topics include the ones discussed above, as well as incorporating features into the HMM network.

REFERENCES

1. R. Schwartz, "Acoustic-Phonetic Experiment Facility for the Study of Continuous Speech," _IEEE International Conference on Acoustics, Speech and Signal Processing_, Philadelphia, PA, April 1976, pp. 1-4.

2. L.E. Baum, J.A. and J.A. Eagon, "An Inequality with Applications to Statistical Estimations for Probabilistic Functions of Markov Processes and to a Model of Ecology," _Amer. Math Soc. Bulletin_, Vol. 73 1967, pp. 360-362.

3. L.R. Bahl, F. Jelinek, and R.L. Mercer, _Continuous Speech Recongition: Statistical Methods_, North Holland Publishers, Amsterdam, 1982, .

4. W.A. Woods, et al., "Speech Understanding Systems: Final Report," Vols. I-V, BBN Report No. 3438, AD Nos. I: AO35165; II: AO35166; III: AO35167; IV: AØ35277; V: AO35278,, Dec. 1976.

5. L.R. Bahl, J.K. Baker, P.S. Cohen, A.G. Cole, F. Jelinek, B.L. Lewis, and R.L. Mercer, "Automatic Recognition of Continuously Spoken Sentences from A Finite State Grammar," _IEEE International Conference on Acoustics, Speech and Signal Processing_, Tulsa, OK, April 1978, pp. 418-421.

6. S.E. Levinson, L.R. Rabiner, M.M. Sondhi, "Speaker Independent Isolated Digit Recognition Using Hidden Markov Models," _IEEE International Conference on Acoustics, Speech and Signal Processing_, Boston, MA, April 1983, pp. 1Ø49-1Ø52.