D.B. Paul

Speech Recognition Using Hidden Markov Models

The Lincoln robust hidden Markov model speech recognizer currently provides state-of-the-art performance for both speaker-dependent and speaker-independent large-vocabulary continuous-speech recognition. An early isolated-word version similarly improved the state of the art on a speaker-stress-robustness isolated-word task. This article combines hidden Markov model and speech recognition tutorials with a description of the above recognition systems.

1. Introduction

There are two related speech tasks: speech understanding and speech recognition. Speech understanding is getting the meaning of an utterance such that one can respond properly whether or not one has correctly recognized all of the words. Speech recognition is simply transcribing the speech without necessarily knowing the meaning of the utterance. The two can be combined, but the task described here is purely recognition.

Automatic speech recognition and understanding have a number of practical uses. Data input to a machine is the generic use, but in what circumstances is speech the preferred or only mode? An eyes-and-hands-busy user—such as a quality control inspector, inventory taker, cartographer, radiologist (medical X-ray reader), mail sorter, or aircraft pilot—is one example. Another use is transcription in the business environment where it may be faster to remove the distraction of typing for the nontypist. The technology is also helpful to handicapped persons who might otherwise require helpers to control their environments.

Automatic speech recognition has a long history of being a difficult problem—the first papers date from about 1950 [1]. During this period, a number of techniques, such as linear-time-scaled word-template matching, dynamic-time-warped word-template matching, linguistically motivated approaches (find the phonemes, assemble into words, assemble into sentences), and hidden Markov models (HMM), were used. Of all of the available techniques, HMMs are currently yielding the best performance.

This article will first describe HMMs and their training and recognition algorithms. It will then discuss the speech recognition problem and how HMMs are used to perform speech recognition. Next, it will present the speaker-stress problem and our stress-resistant isolated-word recognition (IWR) system. Finally, it will show how we adapted the IWR system to large-vocabulary continuous-speech recognition (CSR).

2. The Hidden Markov Model

Template comparison methods of speech recognition (e.g., dynamic time warping [2]) directly compare the unknown utterance to known examples. Instead HMM creates stochastic models from known utterances and compares the probability that the unknown utterance was generated by each model. HMMs are a broad class of doubly stochastic models for nonstationary signals that can be inserted into other stochastic models to incorporate information from several hierarchical knowledge sources. Since we do not know how to choose the form of this model automatically but, once given a form, have efficient automatic methods of estimating its parameters, we must instead choose the form according to our knowledge of the application domain and train the parameters from known data. Thus the modeling prob-

Glossary

coarticulation	the effect of an adjacent phone on the current phone
CSR	continuous-speech recognition
decode	evaluation of $p(O \mid M)$
diphone model	left- or right-phone context-sensitive phone model
flat start	training initialization in which all states have the same parameter values
FSG	finite-state grammar
HMM	hidden Markov model
IWR	isolated-word recognition
mixture	a weighted sum of pdfs; the weights must sum to 1 and be non-negative
ML	maximum likelihood
MMI	maximum mutual information
monophone model	context-insensitive phone model
observation	(1) generation: the parameter emitted by the model; (2) decoding: the measurement absorbed by the model; may be discrete or continuous valued
pdf	probability distribution function: may be discrete (i.e., a probabilistic histogram) or continuous (e.g., a Gaussian or a Gaussian mixture)
perplexity	a measure of the recognition task difficulty: geometric-mean branching factor of the language
phone	the acoustic realization of a phoneme: a phoneme may be realized by one of several phones
phoneme	a linguistic unit used to construct words
RM	the DARPA 1000-word Resource Management CSR database [3]
SD	speaker dependent (train and test on the same speaker)
SI	speaker independent (train and test on disjoint sets of speakers)
TI-20	the Texas Instruments 20-word IWR database [4]
TI-105	the Texas Instruments 105-word IWR simulated-stress database [5]
tied mixture (TM)	a set of mixtures in which all mixtures share the same elemental pdfs
tied states	a set of states that are constrained to have the same parameters
triphone model	left- and right-phone context-sensitive phone model
VQ	vector quantizer: creates discrete observations from continuous observations (measurements) by outputting the label of the nearest template of a template set according to some distance measure
WBCD	word-boundary context dependent (triphones)
WBCF	word-boundary context free (triphones)

lem is transformed into a parameter estimation problem.

A.A. Markov first used Markov models to model letter sequences in Russian [6]. Such a model might have one state per letter with probabilistic arcs between each state. Each letter would cause (or be produced by) a transition to its corresponding state. One could then

train the model (estimate the transition probabilities) from some text, and use the model to compute the probability of a letter sequence or to generate letter sequences.

The theory of HMMs was developed in the mid-to-late 1960s by Baum, Eagon, Petrie, Soules, and Weiss [7–10]. HMMs were first used for speech recognition in the early 1970s by Jim Baker at Carnegie-Mellon University [11, 12], and shortly thereafter by Fred Jelinek's group at IBM [13]. The techniques were inconsistent with the prevailing philosophy of the time and were not widely understood until a 1983 tutorial by Levinson [14] became available. Since that time, many, if not most, speech recognition groups have switched to using HMMs.

In the Markov chain as used by Markov [6], the state sequence is visible to the observer through the observation (letter) sequence. The critical difference in HMMs is that each state transition will emit (or absorb) an observation according to some probability density function (pdf). The state sequence cannot be uniquely determined from the observation sequence and is therefore *hidden*.

In speech recognition, HMMs are used to model a nonstationary signal. However, they have been used in a variety of fields such as language, financial, and biological modeling. See Poritz [15] for a larger application list and bibliography.

2.1 The Model

An HMM M is defined by a set of N states, K observation symbols, and three probabilistic matrices:

$$M = \{\Pi, A, B\} \qquad (1)$$

where

$\Pi = \pi_i$ initial state probabilities

$A = a_{i,j}$ state transition probabilities

$B = b_{i,j,k}$ symbol emission probabilities.

A set of sample model topologies is shown in Fig. 1. The observation-symbol generation procedure for topology 1(a) is as follows:

(1) Start in state i with probability π_i.
(2) $t = 1$.
(3) Move from state i to j with probability $a_{i,j}$ and emit observation symbol $o_t = k$ with probability $b_{i,j,k}$.
(4) $t = t+1$.
(5) Go to 3.

There are a number of possible variations on this model: $B = b_{i,k}$ depends only upon the source state and $B = b_{j,k}$ depends only upon the destination state. (These variations are tyings, described in section 2.6.) Another variation is substituting continuous observations for the discrete observations used in the above definition. We use $B = b_i(o)$ in our speech recognition systems where b_i is a Gaussian or Gaussian-mixture pdf dependent only upon the source state and o is an observation vector. (Our pdfs

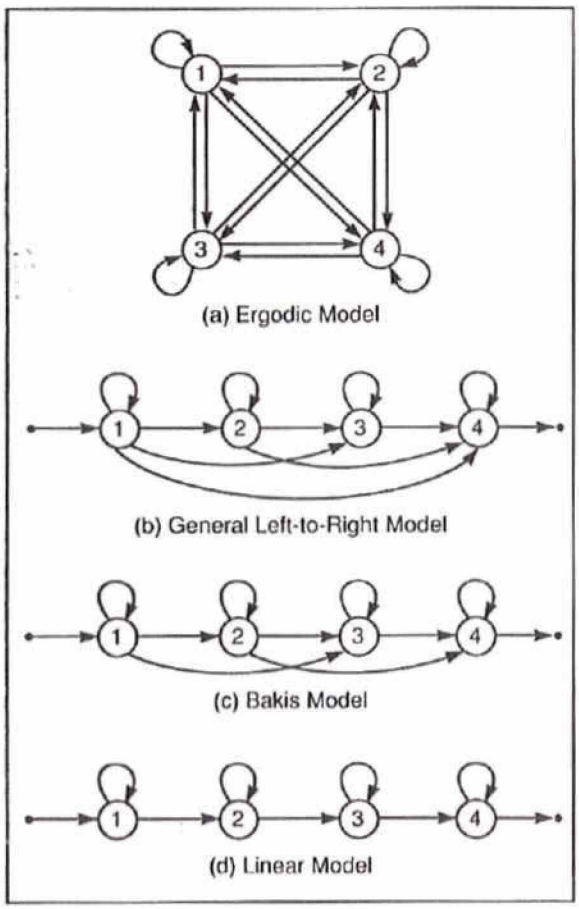


Fig. 1—Sample four-state HMM topologies.

will be described in detail later.)

To use this HMM, three basic algorithms must be understood: classification of an unknown observation sequence (recognition), training the models from a set of training data, and evaluation of the probability of an observation sequence. Classification is identification of an unknown observation sequence by choosing the most likely class to have produced the observation sequence. To perform this identification, one must compute the likelihood for a model and choose the most likely class. Training is estimation of the parameters of a model from a set of known training data (observation sequences). Evaluation of the probability of an observation sequence given a model (the *de code*) is a critical computation used in both training and classification.

The following discussion of the basic algorithms will use discrete-observation terminology. However, the discussion also applies to continuous observations by substituting $b_{i,j}(o)$ for $b_{i,j,k}$.

2.2 HMM Network Topologies

The topology of Fig. 1(a) is called an ergodic network because any state can be reached from any other state. This topology is inappropriate for speech recognition because speech consists of an ordered sequence of sounds. Instead we generally use a topology from the class of *left-to-right* models. In a left-to-right model a state, once left, cannot be revisited. (Equivalently, the states can be numbered such that the A matrix is upper-triangular.) This restriction forces an ordering on the state sequence—a transition can only stay in the same state or go to a higher numbered state. Figures 1(b) through 1(d) are all left-to-right models. Figure 1(b) is the general left-to-right topology. Figure 1(c), a *Bakis* model [16] (stay, move, or skip one), and Fig. 1(d), a *linear* (stay or move) model, are commonly used for speech recognition. The topologies 1(b) through 1(d) are progressively more restrictive: all start in state 1, finish in state N, and have additional disallowed transitions. In fact, they are just special cases of Fig. 1(a), in which some of the transition probabilities have been set to zero. The mathematics of all cases is identical, so no special attention need be paid to the restrictive cases. There is also a pragmatic trade-off: the more restrictive topologies generally require less training data and thus a simpler model may give better performance than a more complex model.

The topologies in Figs. 1(b) through 1(d) are drawn with network entry and end-of-data arcs. The entry arc emphasizes that paths in these topologies can only start in state 1. The exit arc is a special end-of-data arc that can be transited only at the end of data. In effect, there is a special end-of-data observation symbol for which $b_{i,j,k=end} = 0$ except for the final arc on which $b_{i=N,j=external,k=end} = 1$. This convention allows all states to be identical in form and anticipates concatenating phone models to create word models and concatenating word models to create sentence models (described in more detail in sections 5 and 6). Otherwise state N must have a self-transition probability of 1, which will make it behave differently from the other states.

The figures for the following algorithms will use the linear topology of Fig. 1(d) because it is the simplest practical topology for speech recognition. The equations presented below, however, are fully general.

2.3 Classification Using HMMs

HMMs use a maximum *a posteriori* likelihood classification (recognition) rule—choose the most likely class given the observations. For some observation sequence $O = o_1, o_2, \ldots, o_T$:

$$chosen\_class = \underset{class}{\operatorname{argmax}} \left[P(M_{class}|O) \right]. \qquad (2)$$

By Bayes rule,

$$p(M_{class}|O) = \frac{p(O|M_{class})\,p(class)}{p(O)}. \qquad (3)$$

But since the observation probability $p(O)$ is the same for all classes, it need not be computed, and the *a posteriori* likelihood is used instead of the *a posteriori* probability:

$$l(M_{class}|O) = p(O|M_{class})\,p(class). \qquad (4)$$

The optimal recognition-model parameters for each class are the same as the parameters for each class in the generation model. (However, in many cases, the generator is not an HMM or the generation model parameters are not available.) Evaluation of $p(O|M)$ will be discussed in section 2.4, and $p(class)$ allows us to apply constraints from higher-level knowledge sources such as a language model (word sequence probabilities) as will be described in section 5.

2.4 *Evaluation of p(O|M)*

Numerical evaluation of the conditional probability $p(O|M)$ from Eq. 4 is vital to both the training and recognition processes. (This computation is commonly called a decode due to a similar operation used in communication theory.) This is the probability that any path (i.e., the sum of the probabilities of all paths) through the network has generated the observation sequence O. Thus for the set of possible state sequences $\{S\}$

$$p(O|M) = \sum_{\{S\}} \left(\pi_{s_1} \prod_{t=1}^{t=T} a_{s_t, s_{t+1}} \, b_{s_t, s_{t+1}, o_t} \right) . \quad (5)$$

The complexity of this computation is on the order of N^T ($O(N^T)$) and therefore completely intractable for any nontrivial model.

Fortunately, there is an iterative method of evaluation that is far more efficient than the direct method:

$$\alpha_1(i) = \pi_i \qquad 1 \le i \le N \quad (6.1)$$

For $t = 1, 2, \ldots, T$:

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{i,j} b_{i,j,o_t} \qquad 1 \le j \le N \quad (6.2)$$

$$p(O|M) = \sum_{i \in \{\text{terminal states}\}} \alpha_{T+1}(i) . \quad (6.3)$$

This same procedure is shown graphically in Fig. 2(a) for the linear topology of Fig. 1(d). The lower-left lattice point is initialized to 1 since the topology defines state 1 as the start (Eq. 6.1). All other lattice points are initialized to 0. The paths for this topology can either stay in a state (the horizontal arcs) or move to the next state (the diagonal arcs). In each case, the arc probability is the product of the source lattice-point probability $\alpha_t(i)$, the transition probability $a_{i,j}$, and the observation probability b_{i,j,o_t}. Each lattice point at time $t+1$ sums the probabilities of the incoming arcs (Eq. 6.2). Thus the probability at each lattice point is the probability of getting from the start state(s) to the current state at the current time. The final probability is the sum of the probabilities on the exit states (Eq. 6.3). (There is only one exit state in topology 1[d].) This operation is the *forward decoder*. Time-synchronous (i.e., all states are updated at the same time) left-to-right models can be computed in place by updating the state probabilities in reverse (state number) order.

The *backward decoder* starts at the exit states and applies the observations in reverse order (Fig. 2[b]):

$$\beta_{T+1}(i) = 1 \qquad i \in \{\text{terminal states}\} \quad (7.1)$$

For $t = T, T-1, \ldots, 1$

$$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} b_{i,j,o_t} \beta_{t+1}(j) \qquad 1 \le i \le N \quad (7.2)$$

$$p(O|M) = \sum_{i=1}^{N} \pi_i \beta_1(i) . \quad (7.3)$$

The probability of each lattice point is the probability of getting from the current state and time to the exit state(s) at time $T+1$. This decoder produces the same result as the forward decoder and can also be computed in place for a time-synchronous left-to-right model. While either decoder can be used for classification, usually only the forward is used because the computation can begin before all of the observations are available. However, both are required for the forward-backward training algorithm described in section 2.5. Both decoders are $O(N^2T)$ for ergodic models and $O(NT)$ for typical speech models, which is practical to compute for the models used in speech recognition.

The combined decoders have the property

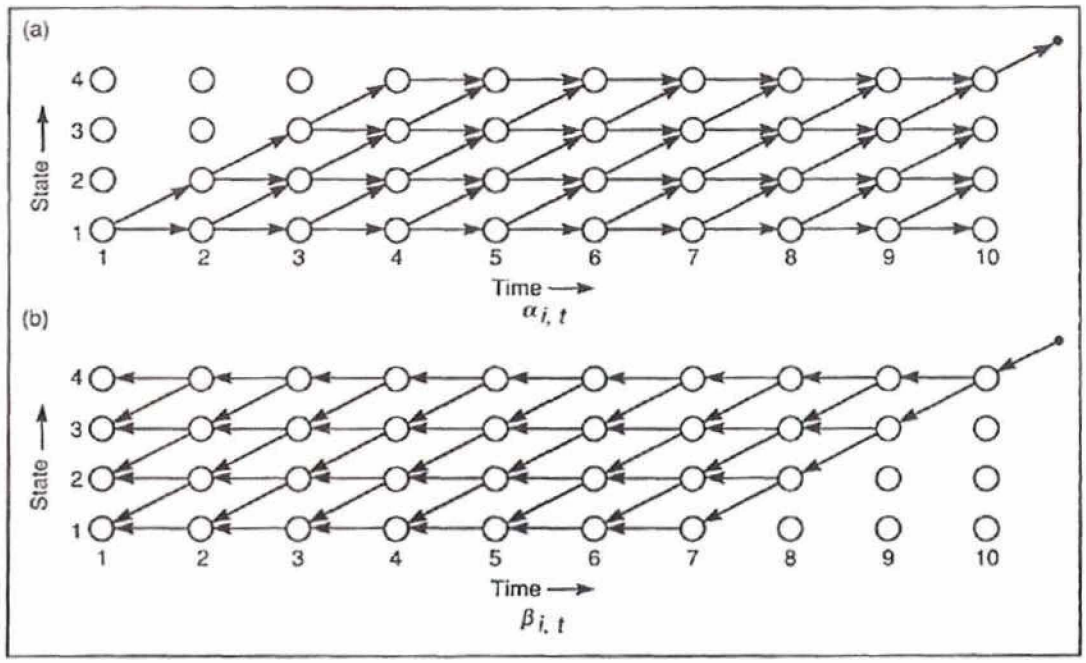


Fig. 2—Decoder lattices for four-state linear model of Fig. 1 (d): (a) forward decoder lattice, (b) backward decoder lattice.

that the sum of the products of the forward and backward probabilities of all states at any given time is $p(O|M)$:

$$p(O|M) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \qquad 1 \le t \le T+1. \quad (8)$$

Eqs 6.3 and 7.3 are special cases of this equation for $t = T+1$ and $t = 1$, respectively.

The above full decoders compute the probability of any state sequence that generates the observations. The full decoders are theoretically optimum for the classification (recognition) task. The suboptimum *Viterbi decoder*, $p_v(O|M)$, is often used because it computes the probability of the best state sequence. (The full decoder does not produce the single best path—it uses all possible paths.) In the Viterbi decoder the maximum of the entering probabilities is used in Eq. 6.2 rather than the sum. If the backpointers (pointers that indicate highest-probability entering arc) are recorded, the best path can be constructed by tracing back from the best exit state (the Viterbi backtrace). This best path, which assigns observations to arcs, aligns the data to the model. Viterbi decoder-based speech recognizers usually produce similar recognition results to the full decoder-based systems.

2.5 *Training the Model*

Given an HMM topology and a set of training observations, it is possible to train (or choose an optimum set of parameters for) the models. The *forward-backward*, or *Baum-Welch*, reestimation algorithm trains the HMMs according to a maximum-likelihood (ML) criterion:

$$\underset{M}{\text{maximize}}\ P\left(O_{training}|M\right). \quad (9)$$

This is an expectation-maximization, or estimate-maximize (EM), algorithm: the expectation phase aligns the training data to the model and the maximization phase reestimates the parameters of the model. The expectation phase consists of computing the probability of traversing a lattice arc at time t given the observation sequence O:

$$p_{arc}(i,j,t) = \alpha_t(i)a_{i,j}b_{i,j,o_t}\beta_{t+1}(j)/p(O|M). \quad (10)$$

The α term is the probability of getting from the start state(s) to state i at time t, the a term is the probability of transitioning from state i to state j, the b term is the probability that the observation o_t occurs on the transition from i to j, and the β term is the probability of getting from state j at time $t+1$ to the exit(s) of the network. The $p(O|M)$ term serves to normalize p_{arc} such that

$$\sum_{i=1}^{N}\sum_{j=1}^{N} p_{arc}(i,j,t) = 1 \qquad 1 \le t \le T \qquad (11)$$

due to Eq. 8.

The maximization phase consists of estimating the model parameters to obtain a new model, $\overline{M}$:

$$\overline{\pi_i} = \sum_{j=1}^{N} p_{arc}(i,j,t=1) \qquad 1 \le i \le N \qquad (12.1)$$
$$= \frac{\alpha_1(i)\beta_1(i)}{p(O|M)}$$

$$\overline{a_{i,j}} = \frac{\sum_{t=1}^{T} p_{arc}(i,j,t)}{\sum_{j=1}^{N}\sum_{t=1}^{T} p_{arc}(i,j,t)} \qquad 1 \le i,j \le N. \qquad (12.2)$$

For a discrete-observation system

$$\overline{b_{i,j,k}} = \frac{\sum_{t\in\{o_t=k\}} p_{arc}(i,j,t)}{\sum_{t=1}^{T} p_{arc}(i,j,t)} \qquad \begin{matrix} 1 \le k \le K, \\ 1 \le i,j \le N. \end{matrix} \qquad (12.3)$$

For a continuous-observation system with Gaussian pdfs

$$\overline{\mu_{i,j}} = \frac{\sum_{t=1}^{T} p_{arc}(i,j,t)\,o_t}{\sum_{t=1}^{T} p_{arc}(i,j,t)} \qquad 1 \le i,j \le N \qquad (12.4)$$

$$\overline{\Sigma_{i,j}} = \frac{\sum_{t=1}^{T} p_{arc}(i,j,t)\,o_t o_t^{tr}}{\sum_{t=1}^{T} p_{arc}(i,j,t)} - \overline{\mu_{i,j}}\,\overline{\mu_{i,j}}^{tr} \qquad (12.5)$$
$$1 \le i,j \le N$$

where o_t is the observation vector, μ is the mean vector, tr denotes vector transpose, and Σ is the correlation matrix. (A Gaussian pdf is, of course, defined by its μ and Σ.)

The interpretation of these equations is very simple. The term $p_{arc}(i,j,t)$ is simply a probabilistic count of the number of times the arc from state i to j is traversed at time t. Thus Eq. 12.1 is the number of times the path starts in state i, Eq. 12.2 is the number of times arc i,j is traversed divided by the total number of departures from state i, and Eq. 12.3 is the number of times the symbol k is emitted from arc i,j divided by the total number of symbols emitted from arc i,j. Similarly, Eqs. 12.4 and 12.5 are just (arc) weighted averages for computing the mean vector μ and covariance matrix Σ.

The above equations assume one training token (a token is an observation sequence generated by a single instance of the event, such as a single instance of a word). The extension to multiple training tokens simply computes the sums over all tokens, which maximizes

$$\prod_{i=tokens} p(O_i|M).$$

The proof of this reestimation procedure [8, 10, 14] guarantees that $p(O|\overline{M}) \ge p(O|M)$. This training procedure acts much like a gradient hill climb with automatic step sizing—it starts with an initial set of parameters and improves them with each iteration until it reaches a local maximum.

Inspection of the forward-backward algorithm shows it to be a collection of local ML optimizations. Each individual parameter—the transition probabilities leaving a single state or the observation pdf from a single arc—is set to its ML value given the data alignment from the expectation phase. The proof also guarantees that the total effect of any subset of these local optimizations is $p(O|\overline{M}) \ge p(O|M)$. Thus one can choose to train only some of the parameters while leaving the others fixed.

The forward-backward algorithm simultaneously considers all possible alignments of the data to the model (paths), each weighted by its probability. The *Viterbi training* procedure uses

a Viterbi decoder to derive the counts from the backtrace (section 2.4). (The counts p_{arc} are now 1's or 0's and fit into the same reestimation equations.) This procedure, unlike the forward-backward algorithm, considers only the best path through the model. As a result, it is much more sensitive to the initial model parameters.

The restricted topologies can be viewed as fully connected models with some of the transition probabilities set to zero. The forward-backward and Viterbi training algorithms maintain these zero probabilities because $a_{i,j} = 0$ implies $p_{arc}(i, j, t) = 0$ for all t and thus the numerator of Eq. 12.2 and $\overline{a_{i,j}}$ must also be zero. The training algorithms can set a transition or symbol emission probability to zero, but once zero it remains zero.

There are two other training methods available for HMMs: gradient hill climbing and simulated annealing. The gradient method [14], which must be modified to bound the values of the probabilities by zero and one, is computationally expensive and requires step-size estimation. Simulated annealing has also been tested as a training method [17]. It demands far more computation and discards the initial model parameters—which convey useful information into the models by helping to choose the local maximum.

The ML criterion is not the only possible training criterion. Several other criteria that consider the incorrect as well as the correct words—such as maximum mutual information (MMI) [18]—have been proposed. These criteria generally require significantly more computation than the forward-backward algorithm and have not shown improved speech recognition performance on any but very small tasks. A philosophically related procedure called corrective training [19], which modifies ML models based upon possible confusions obtained from a modified recognizer, has provided modest performance improvements.

2.6 Tied States, Null Arcs, and Null States

A very useful tool for HMMs is *tied states*. Tied states are a set of states in which all states have the same transition probabilities and observation pdfs. States may also be partially tied by tying only some of the state parameters. This allows us to constrain the models such that, for example, all instances of a particular phone have the same model and this model is trained on all instances of the phone in the training data. It also reduces the number of parameters that must be estimated from the necessarily finite amount of training data.

Another useful tool is the *null arc*, which does not emit an observation symbol. A single null arc is equivalent to arcs from all immediately preceding states to all immediately following states with appropriate tyings on the transition probabilities. (Successive null arcs are more complicated but are an extension of the single null arc.) The null arc is associated with the state rather than its surrounding states, and thus may be more convenient and require fewer parameters than the equivalent network without the null arcs.

A similarly useful tool is the *null state*, which has no self-transition and only null exiting arcs. It is a path redistribution point that may be used to induce tyings on the previous states with a simplified organization. For example, if models of the form shown in Fig. 1(c) were concatenated, null states might be placed at the junctions.

2.7 Discrete Observations

Some tasks, such as the letter sequence task used by Markov [6], inherently use a finite alphabet of discrete symbols. Many other tasks, including speech recognition, have continuous-valued observations (measurements). A *vector quantizer* (VQ) [20] can be used to convert the measurements into discrete observations. A typical VQ used in speech recognition contains a set of spectral vector templates (sometimes called the codebook) and outputs the label of the closest template according to a distance measure. This operation converts a sequence of spectral vectors into a sequence of best template labels, which is the discrete-observation sequence.

2.8 Continuous Observations

The preceding sections describe the discrete observation and continuous observation with single-Gaussian-pdf-per-state models. The Gaussian pdf, while not the only continuous-observation pdf for which convergence of the forward-backward algorithm has been proved, has simple mathematics and is the most commonly used continuous-observation pdf for HMM speech recognition. Only the Gaussian pdf will be discussed here. The single Gaussian model has the disadvantage that it is a unimodal distribution. A multimodal distribution can be obtained from a *Gaussian mixture*, or weighted sum of Gaussians (G):

$$\sum_i c_i G(o, \mu_i, \Sigma_i)$$
$$\sum_i c_i = 1, \quad c_i \geq 0. \tag{13}$$

The Gaussian mixture can be redrawn as a subnet of single Gaussian (per state) states by using null and tied states and is therefore a convenience, but not a fundamental extension to HMMs.

Recently a new form of mixture has emerged in the speech recognition field: the *tied mixture* (TM) [21–23]. Gaussian tied mixtures are a set of mixtures that share the same set of Gaussians. The traditional discrete-observation system used for speech recognition—a VQ followed by a discrete-observation HMM—is a special case of a TM system. It is a pruned TM system in which only the single highest-probability Gaussian is used. (A TM system can also be viewed as an extension of a discrete-observation system in which all observation symbols are used all of the time, but weighted according to their probabilities.) The pdf of the TM system combines the generality of the nonparametric (probabilistic histogram) pdf of the discrete-observation system with the direct access (to the measurements) of the continuous-observation system and the pdf smoothing of the Gaussian pdf systems. Unlike the traditional discrete-observation system, TM systems allow simultaneous optimization of both the Gaussians, which correspond to the VQ spectral templates, and the weights, which correspond to the discrete pdf.

2.9 Pragmatic Issues

The preceding sections describe the basic techniques of HMMs. In addition, several pragmatic issues must be addressed if one wishes to use HMMs. The first is numerical underflow. After several probabilities have been multiplied, the numbers become so small that most computers will underflow. (The Lincoln Laboratory system routinely produces likelihoods on the order of 10^{-2000}.) Thus these systems must store the numbers in a log format or use scaling.

Second, for best performance the complexity of the model must be matched to the amount of training data. Tying states or parameters is one method of reducing the number of free variables to be trained. The form of the observation pdf must also be chosen with consideration for this limitation. Frequently, the parameters are smoothed with less-detailed but better-trained models. Deleted interpolation [24] is an HMM-based method that automatically determines the smoothing weights. Smoothing of phonetic models will be described in section 6.2

Finally, a discrete-observation HMM has a missing observation problem. A test token may result in an observation symbol that was not seen during training of the class. The symbol's probability will have been set to zero during training and thus the class probability will go to zero. To prevent this problem, a lower limit is placed upon the values of the observation pdf. This limit prevents a small number of previously unobserved symbols from eliminating the correct class from consideration.

3. Basic HMM Isolated-Word Recognition

Normally we speak in continuous sequences of words called sentences. In continuous speech, there is considerable variation due to the sentence structure (prosodics) and interaction between adjacent words (cross-word coarticulation). In addition, no clear acoustic mark-

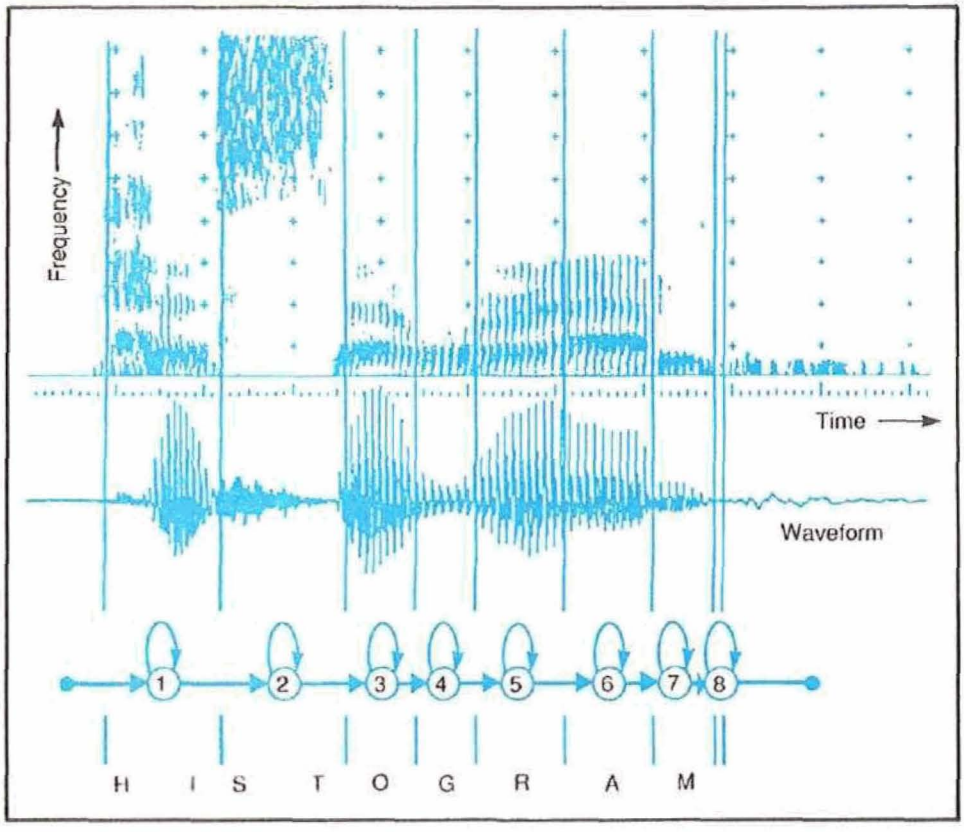


Fig. 3—Viterbi decoder alignment for the word "histogram."

ers delineate the word boundaries. Therefore, it is simpler to recognize isolated words. The word pronunciations are much more uniform and the boundaries are much easier to locate than in continuous speech. Therefore, isolated-word recognition will be described first.

In performing recognition upon a naturally occurring signal, such as human speech, we do not have access to the generator and therefore are unable to determine its structure directly. (It is almost certain that humans do not use HMM models. Even if the speech generator does not use HMMs, we have found that we can model the speech signal fairly well with HMMs.) Thus we must guess the topology of the model. For IWR Bakis and linear models have been found to work well, but the number of states N needs to be determined.

The speech signal itself is not a good observation. However, work in psychoacoustics and voice coding has found the short-term spectral envelope to be a good way to represent the speech. Thus most speech recognition systems use some representation of the spectral envelope, computed at 10-to-20-ms intervals as the basic measurement. A continuous-observation system uses this measurement directly, but a discrete-observation system must use a VQ to convert each measurement into one of the set of observation symbols. This preprocessor is called the *front end*.

The basic system outlined here uses *word models*. Word models treat the word as the fundamental unit to be modeled as a sequence of sounds. Each word model is completely independent—it does not make use of the fact that words are built from a small inventory of phonemes.

To train the system, one records several tokens of each word in the vocabulary and processes them with a front end to create the observation sequences corresponding to each

token. The model might be initialized to a *flat* start (all states have the same initial parameters) and trained by using the forward-backward algorithm. The training data need only be identified by its word label (orthographic transcription)—no detailed internal marking is required. The training will dynamically align each training token to the model and customize the states and arcs.

The recognition process must first choose a probability for each word. Usually, all words are assumed to have equal probability, but unequal probabilities are used in some applications. Each unknown word is then processed with the front end and passed to the HMM, which chooses the most likely word according to Eq. 4. If the likelihood is too low, the recognizer can reject the utterance. (Rejection can help to eliminate out-of-vocabulary words, poor pronunciations, or extraneous noises.)

A sample decode for the word "histogram" is shown in Fig. 3. An eight-state linear model for the word was trained from a flat start and was used to perform a Viterbi decode. Vertical lines have been drawn on a spectrogram of the word to show the location of the state transitions. The figure shows how the model tends to place each stationary region into a single state; however, the /h/ and /l/—which are very dissimilar—were lumped into state 1. This occurred because the training procedure is a collection of local optimizations and the topological constraints—the model was unable to split state 1 into two states during training. The decode also attempted to discard state 8 but could not because no skip path was provided. State 8 probably modeled an alternative pronunciation and the highest likelihood was obtained by minimizing the amount of time spent in the state.

4. The Lincoln Robust IWR

Our initial goal was to improve the accuracy of speech recognition in the cockpit of a modern high-performance aircraft. The pilot is in a noisy environment and under task, mental, and physical stress, all of which affect speech and degrade recognition performance. (The same factors occur in a wide variety of military and civilian environments.) These factors cause a variety of effects in speech—loud speech, the Lombard effect [25], careless speech, fast speech, or even a speech block. These effects, in turn, cause a number of acoustic effects such as changed spectral tilts, changed formant (vocal-tract resonance) positions, changed energies, changed timings, and changes in the phonology [26, 27].

We felt that the VQ required by a discrete-observation system would impose a barrier between the signal and the HMM. Thus we chose a continuous-observation HMM for our robustness work. The following sections describe our front end, our baseline system, and our robust IWR system.

4.1 The Lincoln Front End

The Lincoln Laboratory IWR and CSR systems use a pseudo-filter-bank [28] front end. This front end produces a 100-per-second mel-cepstral observation vector by the following signal processing. The input speech can be sampled at rates between 8 and 20 kHz.

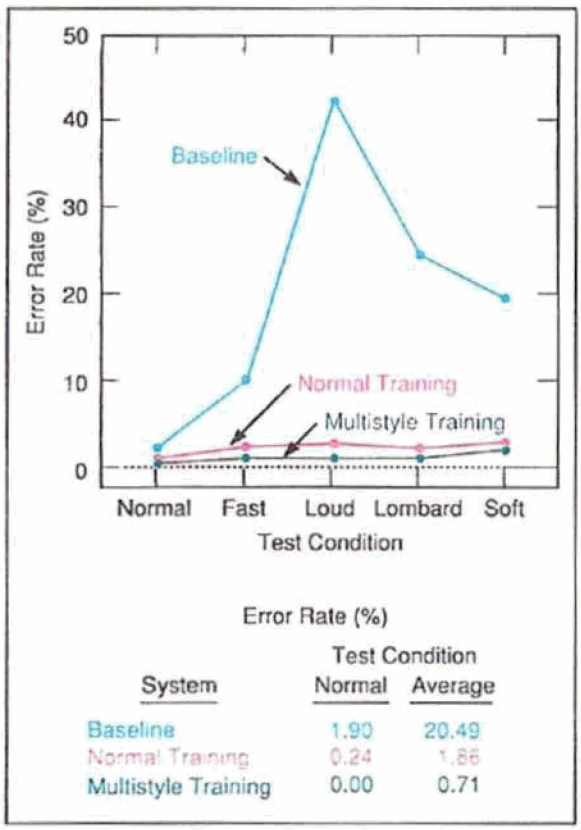
(1) Apply a 20-ms Hamming window and zero pad to 256 or 512 points.
(2) FFT (→ complex spectrum).
(3) Magnitude squared (→ power spectrum $S(f)$).
(4) Preemphasize $S(f) = S(f)\left(1+\left(\frac{f}{500Hz}\right)^2\right)$.
(5) Mel-bandpass weighted summations (→ mel-power spectrum) using constant area triangular filters: 100-Hz spacing 0.1 kHz to 1 kHz, 10% above; width = 2× spacing.
(6) Convert mel-power spectrum to db ($db(x) = 10 \log_{10}(x)$).
(7) Modified cosine transform [28] (→ mel-cepstrum c_i).
(8) Truncate the mel-cepstrum to the desired number of components.
(9) Advance signal 10 ms and repeat from step 1.

The mel-scale is a nonlinear frequency scale [29], consistent with several phenomena ob-

served in human hearing, that can be approximated by a linear scale below 1 kHz and a logarithmic scale above 1 kHz. The mel-cepstrum of speech is also a relatively orthogonal feature set. Thus the mel-cepstrum has both good psychoacoustic properties and good pattern recognition properties. The mel-cepstrum is a transform of the speech spectral envelope—which carries most of the desired information in the English and the European languages. (Pitch is phonemic in some other languages.) The truncation in step 8 removes mel-cepstral components that can be affected by the speaker's pitch.

4.2 The TI Simulated-Stress IWR Database

Since large amounts of truly stressed speech data are difficult to obtain, we used a speaker-dependent simulated-stress database recorded at Texas Instruments (TI) [5]. Although the speakers were asked to speak in a variety of styles, the acoustic changes that occurred are similar to those that occur during real stress [26, 27]. This database uses a 105-word aircraft vocabulary and is thus frequently called the *TI-105* database. It contains eight speakers —five male and three female, each of whom produced a full set of training and test utterances. The training portion consists of five normally spoken tokens of each word and the test portion consists of two tokens of each word, spoken in the following styles: normal, fast, loud, Lombard (speakers wearing headphones with noise in the earpieces), and soft. (There is also a shout condition that was so extreme that it was largely ignored.) The database is sampled at 8 kHz. TI also used the same database to work on the stress robustness problem [5]).



Error Rate (%)

System	Test Condition Normal	Average
Baseline	1.90	20.49
Normal Training	0.24	1.86
Multistyle Training	0.00	0.71

Fig. 4—Speaker-dependent Lincoln robust IWR performance on the TI-105 simulated-stress database.

4.3 The Robust IWR Algorithms

We started with a baseline system with the following features:

(1) 10-state linear-topology word HMMs
(2) 1 model per word
(3) trained diagonal-covariance Gaussian pdfs
(4) 12th-order mel-cepstral observation: c_1–c_{12} (the energy term c_0 is not used)
(5) trained from a flat start by the forward-backward algorithm
(6) Viterbi decoder in recognizer
(7) single-state beginning and ending background models
(8) training and recognition open endpoint (the word endpoints are found by the decode)

A 10-state configuration was chosen as a compromise between too few states for long words and too many states for short words. Although some of the template-matching IWR systems use multiple templates per word to model variation, only one model per word was used here because HMMs are stochastic models and thus are inherently capable of modeling variation.

Diagonal covariance (variance) Gaussians were used because we did not have enough training data for full covariance. The energy term was not used due to normalization difficulties. The flat start and the Viterbi decoder were chosen because they were simpler than the alternatives and the background models allowed us to explicitly recognize the background in the decoding process.

The recognition performance of this baseline system is shown in Fig. 4. The normal speech error rate of 1.9% is fairly good (compared to the state of the art when the tests were performed), but the error rates on the styles are quite high with an average error rate over all five test conditions of 20.5%.

From this baseline system we developed two robust systems, one using a perceptually motivated distance measure, and a second using a tied (or grand) variance [30–32]. These systems feature

(1) tied variance (trained or fixed)
(2) observations augmented with temporal difference (*delta*) mel-cepstra (includes Δc_0 term)
(3) automatic number of states
(4) one stage of Viterbi training for initialization
(5) adaptive background model during recognition

Five tokens of each word were found to be insufficient to train a variance vector per state, and thus the variance was tied. Under conditions of insufficient data, a state variance can become too small; in fact, we observed cases in which a component of the variance vector became zero. (The trainer was operating properly—this increased the likelihood of the training data.) The perceptually motivated distance measure [31, 32], which is a manually chosen fixed variance, provides perceptual and pragmatic speech knowledge while it simultaneously avoids the variance training problem.

Formant motion as well as position is important in speech, and the inclusion of the delta mel-cepstra made this information more available to the system [33]. The fixed number of states in the baseline system gave too many states to short words and too few to the long words. Choosing the number of states per word to be *avg_length_in_frames*/3, where the length is determined by an endpointing algorithm, provided appropriate model sizes for all words. A better initial set of parameters for training can be obtained by dividing the observations between the endpoints of each training token into number-of-states equal-duration sections, assigning each section to its corresponding state, and averaging the observations assigned to each state. This *Viterbi start* gives each state a more equal role in the modeling than does the flat start and is equivalent to one iteration of Viterbi training from a forced backtrace. Finally, an adaptive background model was used during recognition, since the recognition background may differ from the background observed during training.

We also improved performance by *multi-style training*, which includes samples of stressed speech in training [34, 30]. (No token was used simultaneously for both training and testing.) This approach allowed the trainer to focus on the consistent aspects while avoiding the stress-variable aspects of the speech.

These robust IWR systems yielded significant improvements over the baseline system, both with and without multi-style training. The results for the perceptually motivated distance measure system are shown in the second and third plots in Fig. 4. The average error rate is reduced from 20.5% to 1.9% for normal training and 0.7% for multi-style training. The normal test-style error rates are also reduced from 1.9% to 0.24% and 0%, respectively. Thus the robustness enhancements and the multi-style training helped both the robustness to speaker stress and the normal speech performance.

As an independent test of the robust IWR system, we tested it on the Texas Instruments 20-word (*TI-20*) database [4], which uses a 20 normally spoken isolated-word vocabulary from 16 speakers. Our first run on this database produced a 0.06% (3/5120) error rate. The best-known previous result was 0.2% (10/5120) [4]. After a small amount of work on the database, the error rate was reduced to 0%. These results

showed that our system performance was also very good on an independent database.

The standard HMM systems have a dying-exponential state-duration model due to the probability on the self-loop. This model is not very realistic for speech, so we investigated some stronger duration models [31]. Two basic approaches were investigated: subnets [35, 31] and explicit duration modeling [36, 37]. Subnets, where each state is replaced by a small network with pdfs tied over all states, were found somewhat promising but doubled the number of states in the network. The explicit duration models did not help due to training and normalization difficulties and significantly greater computation requirements than the standard models. Durations are a function of the speaking rate and other factors. The function is segment dependent, nonlinear, and not well understood. Thus a weak model was found preferable to a poorly trained strong model.

We also tested two other approaches to improving the stress robustness and/or general performance: stress compensation and a second-pass discriminator. Since we were not able to quantify the speaker stress from the signal, the stress compensator used the spectral tilt c_1 to control a mel-spectral compensator [38, 39]. The second-pass discriminator, using pairwise discrimination techniques on information obtained from the Viterbi backtraces, compared the top few choices from the HMM [40, 41]. Both techniques provided some additional improvement; however, both were tested only in the IWR environment. The compensator might, but the discriminator does not appear to extend to the CSR environment.

We investigated supervised stress and speaker adaptation of the models during recognition. Two forms of supervision are possible: correct-incorrect or the identity of the correct word. In the first case, the system can only adapt on the correctly recognized tokens; in the second case, the system can adapt on all tokens. We found limited improvement in the first case, but dramatic improvement in the second case [42]. Thus, in environments where the user will correct recognition errors, the system is able to adapt very effectively.

As a prelude to CSR, we converted the IWR from word models to triphone models. Each word model was composed of a sequence of triphone (context-dependent phone) models (see section 6 for details), which caused a slight increase in the error rate [32]. We also clustered the triphones (using a χ^2-like criterion) and found that the number of triphones could be reduced by one-third before the performance began to suffer [32]. Another study that clustered states of our word models reached a similar conclusion [43].

5. Basic HMM Continuous-Speech Recognition

HMM CSR is performed by using three hierarchical modeling layers: a language-model layer, a word-model layer, and a phone-model layer. The language model contains all of the constraints on the word sequences that the speaker might utter. This language model contains the syntax (structure) and the semantics (meaning) of the language, including all context factors (e.g., who the participants are, what has just happened). This stochastic language model is expressed as

$$p(w_i) = p_{lang}(w_{i-1}, \ldots, w_i | w_1). \qquad (14)$$

The probabilities express the likelihood of each possible utterance. For example, the *finite-state grammar* (FSG) is a simple language model that can be expressed as a finite-state network with words on each arc. (An FSG is simple to use and may be adequate for limited domains, but it is not a good model of general human language.) There are many other classes of language models, and language modeling is a field of research in its own right.

The recognition difficulty of a language is measured by its perplexity [24]:

$$entropy = \mathrm{E}\left[-\sum_i p_{lang}(w_i) \log_2 \left(p_{lang}(w_i) \right) \right] \qquad (15)$$

$$perplexity = 2^{entropy}. \qquad (16)$$

which can be interpreted as the weighted geometric-mean branching factor. While this number does not reflect the acoustic confusability of the vocabulary, it is the best single number in common usage for indicating the difficulty of the recognition task. Fluent English has a vocabulary of over 80,000 words [44] and one assumption-laden estimate of the word perplexity of general English is 142 [45].

Each arc in the language model is a word. The word-modeling layer replaces each word arc with a network of phones describing the ways in which a word may be pronounced. The phone network is created by looking up the *phoneme* pronunciation in a dictionary and using phonetic rules to convert the phoneme sequence into a *phone* network. The system now becomes a network of phones. Each phone is modeled by a small HMM network in which each arc models the acoustic events. These HMM networks replace the phone arcs in the word models to create our final recognition network. The combined network is extremely sparse with braided paths.

All probabilities of the combined network are trained from data by algorithms appropriate to the type of model used at each level. In practice, a large amount of tying reduces the parameters to a manageable number. Recognition is performed by finding the best word path through this network. The network can be very large or infinite—thus it may be necessary to interpret the network during the recognition operation. The size of the network also requires pruned-search strategies to reduce the computation to practical levels.

6. The Lincoln Robust CSR

Our initial goal had been robust CSR for the aircraft environment and the robust IWR was an intermediate goal. The initial development of our CSR system was performed in the context of a robust CSR task [32], but after a short period, our task changed to large-vocabulary normally spoken CSR using the DARPA Resource Management database. Our current CSR system retains many of the robustness features from our earlier work and probably also retains the robust performance of our earlier systems.

6.1 The DARPA Resource Management Database

The DARPA-supported CSR development sites—Bolt Baranek & Newman (BBN) [46], Carnegie-Mellon University [47, 48], Lincoln Laboratory, MIT [49, 50], and SRI [51]—have been using the DARPA Resource Management (RM) database (AT&T is a guest site also using the database [52]). As a result of this shared database and the sharing of our results, our systems incorporate a combination of techniques developed at our own as well as the other sites.

The RM database [3] was designed to model commands to a computerized naval resource (such as ships) database and display system. Operational personnel provided sample sentences that were used to generate a set of patterns. Sentence lists generated from these patterns were filtered by humans to eliminate any unreasonable sentences. The sentences were read in a sound booth, direct digitized at 20 kHz, and downsampled to 16 kHz.

The database consists of two parts, each with three sections. The two parts are speaker dependent (same speakers for training and testing) (SD) and speaker independent (different speakers for training and testing) (SI). Each part contains three sections: training, development test, and evaluation test. Since repeated tests on the same data, in effect, train the system to the test data, the development test data was designated for algorithm development, and the evaluation test data was designated as an independent test set for formal evaluations. (In fact, the evaluation test data is not distributed to the sites until it is time to perform the evaluations.) Only development test results are given here, because all systems were tested on the same data, which gives more accurate system comparisons than comparisons that use different test data. The amounts of data used in these tests are shown in Table 1.

Table 1.			
Condition	*Number of Speakers*	*Number of Sentences*	*Approximate Time*
SD Train	12	600 per speaker	1/2 h
SI-72 Train	72	2880 total	3 h
SI-109 Train	109	3990 total	4 h
Test	12 (SD)	100 per speaker	—

(These amounts of data differ from the amounts shown in Ref. 3. Since the SD test set is used for SI testing, and most of the SD speakers were also used in the SI portion of the database, eight speakers had to be removed from the designated SI training set to create the SI-72 training set and 11 speakers had to be removed from the combined designated SI training and designated SI development test sets to create the SI-109 training set.) These amounts of training data may seem large, but they are minuscule compared to the amount available to humans. By the age of five, a typical child has heard thousands of hours of speech.

There are 991 words in the vocabulary. A perplexity-60 standard word-pair grammar—a list of allowable word pairs without probabilities—has been developed for recognition. The sentence patterns or a bigram (word pair with probabilities) grammar would have lower perplexities, but would have reduced the challenge to the CSR system developers.

6.2 *The Lincoln Robust CSR Algorithms*

The Lincoln Laboratory CSR system has progressed through a range of pdfs starting with single-Gaussian-per-state systems, Gaussian-mixture systems, and, finally, tied-mixture systems. For clarity, the intermediate systems [32] will not be described here, and the current system will be described as if it were descended directly from the whole-word IWR system. However, comparisons will be made to the intermediate systems whenever appropriate.

The system was extended to the CSR task by adding the following training features:

(1) separate mel-cepstral and delta mel-cepstral observation streams
(2) Gaussian tied mixtures (257 Gaussians/stream)
(3) tied variance per stream
(4) three-state linear triphone models
(5) function-word-dependent triphone dictionary
(6) triphones smoothed by diphones and monophones
(7) word-boundary context-dependent (WBCD) triphones
(8) unsupervised monophone bootstrapped training from a flat start

and the following recognition features:

(1) FSG language model
(2) missing triphone extrapolation
(3) Viterbi beam search
(4) word insertion penalty

In the IWR, the delta parameters were simply appended to the observation vector to make a higher-order observation vector. Due to the diagonal covariance matrix, the two parameter sets (mel-cepstrum and delta mel-cepstrum) were treated as if they were statistically independent in the nonmixture systems. However, when mixtures are used, the two sets are no longer independent because the mean vectors for each stream are paired in each mixture component. Thus the mel-cepstrum and the delta mel-cepstrum were separated into two separate observation streams feeding separate mixture pdfs [22, 53]. The mixture probabilities for the two streams are then multiplied to give the full pdf. We tried both combined and sepa-

rate observation streams in our TM system and obtained better recognition results with the separate streams because there were insufficient training data to train the correlations between the streams.

The 257 Gaussians per stream consist of one adaptive background Gaussian for the background model and 256 speech Gaussians. The tied variance is retained from the IWR system—any correlations between the parameters are modeled by the mixtures. The trained tied variance is used rather than the perceptually motivated fixed tied variance because it is more appropriate in the context of mixtures.

Triphones were chosen as the phonetic models. The acoustic realization of a phone depends upon many contextual factors, the strongest of which is the adjacent phones (*coarticulation*). Thus BBN proposed the triphone, or left- and right-phone context-sensitive phone model [54]:

```
monophone dictionary
    six:          s I k s
triphone dictionary
    six:          #-s-I s-I-k  I-k-s k-s-#
```

where "#" represents a word-boundary context. BBN also showed that the triphone-model parameter estimates could be improved by smoothing with left-diphone (left-phone context-sensitive) models, right-diphone models, and monophone (context-insensitive) models [54]:

$$M = \sum_{i \in \text{contexts}} \lambda_i M_i$$

$$\sum_i \lambda_i = 1, \quad \lambda_i \geq 0. \qquad (17)$$

The smoothing weights control the trade-off of highly specific models trained from a small amount of data and more general models trained from larger amounts of data. BBN chose fixed weights as a manually derived function of the number of training tokens for each model, but *deleted interpolation* restructures the weight estimation as an HMM problem and estimates the weights from the data by using the forward-backward algorithm [24]. We currently use a scheme similar to the BBN scheme, but are exploring deleted interpolation.

Because function words (such as articles and prepositions) are so frequent and generally unstressed, they are pronounced differently from other words. However, since so many tokens are available, specific models can be trained for them. Thus we also make the triphones word dependent for the function words [48].

The initial systems used word-boundary context-free (WBCF) triphones at the word boundaries—i.e., the word-boundary contexts were independent of anything on the other side of the boundary. But since phone coarticulation also extends across word boundaries, we implemented word-boundary context-dependent (WBCD) triphones [55]. (Two other DARPA-supported sites simultaneously and independently developed WBCD phone modeling [47, 51].) WBCD triphones simply include the word boundary and the phone on the other side of the word boundary in the contexts used to generate the triphones:

```
dictionary:        word 1    ...ab
                   word 2     cd...

word 1-word 2:  WBCF:    ...a-b-# #-c-d...
                WBCD:    ...a-b-#c b#-c-d...
```

(The word-boundary triphones are distinct from the corresponding word-internal triphones.) Training is easy—just train the observed word-internal and word-boundary triphones. Generating the recognition network is difficult because there will be many unobserved WBCD triphones required to generate the full network. We use a backoff scheme for the word-boundary triphones: if a desired boundary triphone has not been trained, substitute the corresponding WBCF triphone. (Extrapolation—to be discussed shortly—is used to estimate unobserved WBCF or word-internal triphones.) The simpler WBCF systems model each word as a linear sequence of triphones. The WBCD systems model each three-phone or longer word as an entry fan, a linear midsection, and an exit fan, as shown in Fig. 5. (Two-phone words connect the

fans and one-phone words require a full cross-bar, also shown in Fig. 5.)

The TM systems are trained by the following unsupervised bootstrapping procedure:

(1) Initialize the Gaussians by using an EM algorithm (equivalent to forward-backward training of a one-state Gaussian-mixture HMM).
(2) Set all mixture weights equal (flat start).
(3) Train monophones with forward-backward algorithm.
(4) Initialize the triphone models with parameters from the corresponding monophone.
(5) Train the triphones by using the forward-backward algorithm with smoothing.

This procedure requires only orthographically (i.e., only the text) transcribed data—no time registration is required. The monophone training stage has so many copies of each monophone in so many contexts that it is able to "find" the phones in the training text and train good monophone models. By initializing the triphone models with these monophone models, the monophone bootstrap guarantees that the detailed triphone models are trained on the proper data (i.e., the model converges to a good local maximum). In one test, the error rate doubled when we omitted the monophone bootstrap phase of training [32]. The flat start was also found to be superior to the Viterbi start because the linear alignment assumed by the Viterbi start is not accurate for continuous speech.

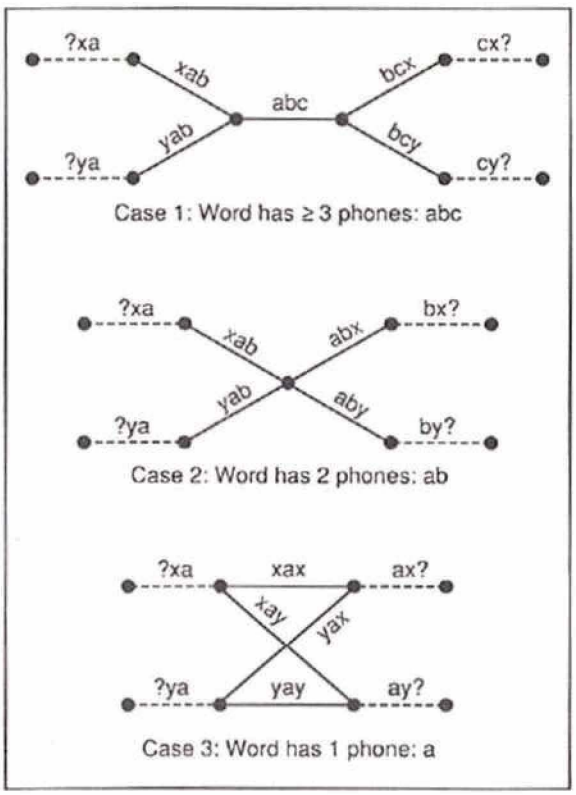


Fig. 5—Recognition network topologies for word-boundary context-dependent (WBCD) word models.

The current recognition system generates a recognition network from the triphone models, the dictionary, and an FSG or a null grammar (any word can follow any other word). When a word-internal or WBCF triphone is required, but not found in the trained phone models, a model is *extrapolated*. Extrapolation uses a weighted average of trained triphones of the same monophone class to estimate the parameters of an unobserved triphone. The network is searched by a time-synchronous Viterbi beam search. The beam search [56] is a pruned-search algorithm that computes a threshold as some constant (<1) times the probability of the highest-probability state. Any states with probabilities below the threshold are pruned. Since several small words can sometimes match a large word, we added a word insertion penalty that controls the trade-off between word insertion errors and word deletion errors. The word sequence of best path from the starting state of the network to exiting state is the recognized output.

6.3 *Performance of the Lincoln CSR Systems*

The Lincoln CSR systems have progressed through many different versions in their development. The following table shows a few key results of systems tested on the RM database. The systems are trained in three ways: SD (600 sentences per speaker), SI-72 (72 speakers, 2880 total sentences), and SI-109 (109 speakers, 3990 total sentences). The test results are obtained on the entire SD development test section: 12 speakers, 1200 total sentences,

10,242 total words. The official word-pair grammar is used in all tests. The word error rate is defined as follows:

$$word\_error\_rate = \frac{substitutions + insertions + deletions}{correct\_number\_of\_words} \quad (17)$$

The word error rates for the systems are shown in Table 2.

Table 2. Training Conditions

System	*SD*	*SI-72*	*SI-109*
Non-TM WBCF	—	12.9%	10.1%
Non-TM WBCD	3.0%	—	—
TM WBCD	1.7%	7.2%*	5.8%*

* includes a second differential observation stream

The non-TM SD systems use one Gaussian per state and the non-TM SI systems use variable-order Gaussian-mixture pdfs. The SD systems provide a factor of two or three better performance than do the SI systems. This ratio is reasonable since the SD systems need only model one speaker while the SI systems must model many speakers. The SI-109 systems perform better than the corresponding SI-72 systems, which indicates that we can probably do better if we have still more training data. (More data would probably also improve our SD system performance.) The TM performance numbers are commensurate with the best of the DARPA systems. However, the best of these systems has a 12% sentence error rate, which is clearly unacceptable in many environments.

Conclusions

Great strides in speech recognition have been made by using hidden Markov models. These models provide a flexible but rigorous stochastic framework in which to build our systems. In addition, the framework provides a computationally efficient training algorithm for estimating the parameters of our models. We were able to exploit this framework to obtain more than an order-of-magnitude improvement over our baseline system in robust isolated-word recognition, and similar improvements were obtained in speaker-dependent and speaker-independent continuous-speech recognition. However, HMMs do not model certain aspects of speech—such as suprasegmental (long span) phenomena—well. Part of the continuing challenge of this research is to adapt HMMs or find new frameworks that improve our ability to model and recognize human speech.

Our performance is still inadequate on our current 1000-word vocabulary task. However, one can trade off recognition difficulty for performance and perform tasks successfully with the currently available technology. For instance, we have a task-stressed SD CSR demonstration with a perplexity 7, 28-word language that has a sentence error rate below 0.1%, and commercial HMM IWRs have been viable for several years. We and other researchers are continuing to work on the large-vocabulary automatic speech recognition problem.

For Further Reading

Several noteworthy tutorials on the use of HMMs in speech recognition are available: Rabiner and Juang [57], Poritz [15], and the tutorial on the forward-backward algorithm by Levinson [14]. Much of the early HMM CSR development took place at IBM, which is described in Ref. 24. Most of the sites working in the field (including us) publish results in the ICASSP proceedings. In addition, the DARPA recognition program publishes a proceedings [58–61]. O'Shaughnessy's book [29] is a modern reference for speech production, speech perception, and basic speech processing.

Acknowledgments

The author wishes to acknowledge the contributions of Yeunung Chen, Richard Lippmann, and Edward Martin to the development of our robust IWR system, and thank the other DARPA speech recognition sites for sharing their knowledge with us. The author also wishes to

thank DARPA program managers Allen Sears (for 1985 through 1988) and Charles Wayne (for 1988 through the present) for their continued support.

References

1. P.B. Denes, "Automatic Speech Recognition: Old and New Ideas," in *Speech Recognition*, ed. D.R. Reddy (Academic Press, New York, 1975), p. 73.
2. H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans Acoust. Speech Signal Process.* **ASSP-26**, 43 (1978).
3. P. Price, W. Fischer, J. Bernstein, and D. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 651.
4. G.R. Doddington and T.B. Schalk, "Speech Recognition: Turning Theory into Practice," *IEEE Spectrum* **18**, 26 (1981).
5. P.K. Rajasekaran, G.R. Doddington, and J.W. Picone, "Recognition of Speech under Stress and in Noise," *Proc. ICASSP 86* **1**, *Tokyo, 7–11 Apr. 1986*, p. 733.
6. A.A. Markov, "An Example of Statistical Investigation in the Text of 'Eugene Onyegin' Illustrating Coupling of 'Tests' in Chains," *Proc. Acad. Sci. St. Petersburg VI Ser.* **7**, 153 (1913).
7. L.E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process," *Inequalities* **III**, 1 (1972).
8. L.E. Baum and J.A. Eagon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology," *Bull. Amer. Math. Stats.* **37**, 360 (1967).
9. L.E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *Ann. Math. Stat.* **37**, 1554 (1966).
10. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Ann. Math. Stat.* **41**, 164 (1970).
11. J.K. Baker, "The Dragon System—An Overview," *IEEE Trans. Acous. Speech Signal Process.* **ASSP-23**, 24 (1975).
12. J.M. Baker, personal communication, Jan. 1990.
13. F. Jelinek, "Continuous Speech Recognition by Statistical Methods," *Proc. IEEE* **64**, 532 (1976).
14. S.E. Levinson, L.R. Rabiner, and M.M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *Bell Sys. Tech. J.* **62**, 1035 (1983).
15. A.B. Poritz, "Hidden Markov Models: A Guided Tour," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 7.
16. R. Bakis, "Continuous Speech Recognition via Centisecond Acoustic States," *J. Acoust. Soc. Am.* **59** *Supp. 1* (1976).
17. D.B. Paul, "Training of HMM Recognizers by Simulated Annealing," *Proc. ICASSP 85* **1**, *Tampa, FL, 26–29 Mar. 1985*, p. 13.
18. L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition," *Proc. ICASSP 86* **1**, *Tokyo, 7–11 Apr. 1986*, p. 49.
19. L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "A New Algorithm for the Estimation of Hidden Markov Model Parameters," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 493.
20. J. Makhoul, S. Roucos, and H. Gish, "Vector Quantization in Speech Coding," *Proc. IEEE* **73**, 1551 (1985).
21. J.R. Bellagarda and D.H. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 13.
22. X.D. Huang, H.W. Hon, and K.F. Lee, "Large Vocabulary Speaker-Independent Continuous Speech Recognition with Semi-Continuous Hidden Markov Models," *Eurospeech 89, Paris, Sept. 1989*.
23. X.D. Huang and M.A. Jack, "Semi-Continuous Hidden Markov Models for Speech Recognition," *Computer Speech and Lang.* **3**, 239 (1989).
24. L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-5**, 179 (Mar. 1983).
25. E. Lombard, "Le Signe de l'Elevation de la Voix," *Ann. Maladière Oreille, Larynx, Nez, Pharynx* **37**, 101 (1911).
26. D. Pisoni, R.H. Bernacki, H.C. Nusbaum, and M. Yuchtman, "Some Acoustic-Phonetic Correlates of Speech Produced in Noise," *Proc. ICASSP 85* **4**, *Tampa, FL, 26–29 Mar. 1985*, p. 1581.
27. B.J. Stanton, L.H. Jamieson, and G.D. Allen, "Acoustic-Phonetic Analysis of Loud and Lombard Speech in Simulated Cockpit Conditions," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 331.
28. S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously-Spoken Sentence," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-28**, 354 (1980).
29. D. O'Shaughnessy, *Speech Communication, Human and Machine* (Addison-Wesley, New York, 1988).
30. R.P. Lippmann, E.A. Martin, and D.B. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," *Proc. ICASSP 87* **2**, *Dallas, 6–9 Apr. 1987*, p. 705; *Proc. DARPA Speech Recognition Workshop, San Diego, 24–26 Mar. 1987*, p. 96.
31. D.B. Paul, "A Speaker-Stress Resistant HMM Isolated Word Recognizer," *Proc. ICASSP 87* **2**, *Dallas, 6–9 Apr. 1987*, p. 713; *Proc. DARPA Speech Recognition Workshop, San Diego, 24–26 Mar. 1987*, p. 85.
32. D.B. Paul and E.A. Martin, "Speaker Stress-Resistant Continuous Speech Recognition," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 283.
33. S. Furui, "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-34**, 52 (1986).
34. R.P. Lippmann, M.A. Mack, and D.B. Paul, "Multi-Style Training for Robust Speech Recognition under Stress," *J. Acoust. Soc. Am. Supp. 1* **79**, 595 (1986).
35. A.E. Cook and M.J. Russell, "Improved Duration Modeling in Hidden Markov Models Using Series-Parallel Configurations of States," *Proc. Institute Acoust. Conf. on Speech and Hearing, 1986*.
36. J.D. Ferguson, "Variable Duration Models for Speech," in *Proc. of the Symp. on the Applications of Hidden Markov Models to Text and Speech*, ed. J. D. Ferguson (IDA-CRD, Princeton, NJ, 1980), p. 143.
37. M. Russell and R. Moore, "Explicit Modeling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Proc. ICASSP 85* **1**, *Tampa, FL, 26–29 Mar. 1985*, p. 5.
38. Y. Chen, "Cepstral Domain Talker Stress Compensation for Robust Speech Recognition," *Technical Report 753*, Lincoln Laboratory (10 Nov. 1986).

DTIC #AD-A-176068.

39. Y. Chen, "Cepstral Domain Stress Compensation for Robust Speech Recognition," *Proc. ICASSP 87* **2**, *Dallas, 6–9 Apr. 1987*, p. 717; *Proc. DARPA Speech Recognition Workshop, San Diego, 24–26 Mar. 1987*, p. 90.
40. E.A. Martin, R.P. Lippmann, and D.B. Paul, "Two-Stage Discriminant Analysis for Improved Isolated-Word Recognition," *Proc. ICASSP 87* **1**, *Dallas, 6–9 April 1987*, p. 709; *Proc. DARPA Speech Recognition Workshop, San Diego, 24–26 Mar. 1987*, p. 100.
41. E.A. Martin, "A Two-Stage Isolated-Word Recognition System Using Discriminant Analysis," *Technical Report 773*, Lincoln Laboratory (5 Aug. 1987), DTIC #AD-A-187425.
42. E.A. Martin, R.P. Lippmann, D.B. Paul, "Dynamic Adaptation of Hidden Markov Models for Robust Isolated-Word Speech Recognition," *Proc. ICASSP 88* **1**, *New York, 11–14 April 1988*, p. 52.
43. R.P. Lippmann and E.A. Martin, "Discriminant Clustering Using an HMM Isolated-Word Recognizer," *Proc. ICASSP 88* **1**, *New York, 11–14 Apr. 1988*, p. 48.
44. F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," *Proc. IEEE* **73**, 1616 (1985).
45. A. Nádas, "Estimation of Probabilities in the Language Model of the IBM Speech Recognition System," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32**, 504 (1984).
46. R. Schwartz, O. Kimball, F. Kubala, M.W. Feng, Y.L. Chow, C. Barry, and J. Makhoul, "Robust Smoothing Methods for Discrete Hidden Markov Models," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 548.
47. K.F. Lee, H.W. Hon, M.Y. Hwang, S. Mahajan, and R. Reddy, "The SPHINX Speech Recognition System," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 445.
48. K.F. Lee, *Automatic Speech Recognition: The Development of the SPHINX System* (Kluwer Academic Publishers, Norwell, MA, 1989).
49. V. Zue, J. Glass, M. Philips, and S. Seneff, "Acoustic Segmentation and Phonetic Classification in the SUMMIT System," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 389.
50. V. Zue, J. Glass, M. Philips, and S. Seneff, "The MIT SUMMIT Speech Recognition System: A Progress Report," *Proc. Feb. 1989 DARPA Speech and Natural Language Workshop, Philadelphia, 21–23 Feb. 1989*, p. 179.
51. M. Weintraub, H. Murveit, M. Cohen, P. Price, J. Bernstein, G. Baldwin, and D. Bell, "Linguistic Constraints in Hidden Markov Model Based Speech Recognition," *Proc. ICASSP 89* **2**, *Glasgow, Scotland, 23–26 May 1989*, p. 699.
52. C.H. Lee, B.H. Juang, F.K. Soong, and L.R. Rabiner, "Word Recognition Using Whole Word and Subword Models," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 683.
53. V.N. Gupta, M. Lennig, and P. Mermelstein, "Integration of Acoustic Information in a Large Vocabulary Word Recognizer," *Proc. ICASSP 87* **2**, *Dallas, 6–9 Apr. 1987*, p. 697.
54. R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech," *Proc. ICASSP 85* **3**, *Tampa, FL, 26–29 Mar. 1985*, p. 1205.
55. D.B. Paul, "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89* **1**, *Glasgow, Scotland, 23–26 May 1989*, p. 449.
56. B.T. Lowerre, "The HARPY Speech Recognition System," Ph.D. thesis, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, 1976.
57. L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-3**, 4 (1986).
58. *Proc. 1986 DARPA Speech Recognition Workshop, Palo Alto, CA, 19–20 Feb. 1986, Science Applications International Corporation Report SAIC-86/1546.*
59. *Proc. 1987 DARPA Speech Recognition Workshop, San Diego, 24–26 Mar. 1987, Science Applications International Corporation Report SAIC-87/164.*
60. *Proc. Feb. 1989 DARPA Speech and Natural Language Workshop, Philadelphia, 21–23 Feb. 1989.*
61. *Proc. Oct. 1989 DARPA Speech and Natural Language Workshop, Cape Cod, MA, 15–18 Oct. 1989.*

DOUGLAS B. PAUL is a staff member in the Speech Systems Technology Group, where his research is in speech bandwidth compression and automatic speech recognition. He received his bachelor's degree from The Johns Hopkins University and his Ph.D degree from MIT, both in electrical engineering. Doug has been at Lincoln Laboratory since 1976. He is a member of Phi Beta Kappa, Tau Beta Pi, and Eta Kappa Nu.