BOSTON UNIVERSITY
COLLEGE OF ENGINEERING
DEPARTMENT OF BIOMEDICAL ENGINEERING
SENIOR PROJECT FINAL REPORT

An Investigation of Signal Processors for Recognition of Telephone Speech

Submitted in Partial Fulfillment for the Degree of Bachelor of Science

Date: April 28, 1995

Technical Advisor: Prof. Mari Ostendorf

Student: John P. Kaufhold _____

Biomedical Engineering Advisor: Prof. Laurel Carney _____

Course Advisor: Prof. Kenneth Lutchen _____

Abstract

Currently, in uncorrupted acoustic environments, a state-of-the-art, continuous speech, 5000-word vocabulary speech recognizer can achieve a recognition accuracy of 95%. In a telephone channel acoustic environment, the same recognition system can double in error rate, which is unacceptable for most applications. It is the goal of this project to quantitatively measure how well a speech recognition system recognizes natural number utterances (e.g. "four thousand lira") over a telephone line. The performance of recognizers will be quantified using word error, which is a measure of the difference between the actual transcription of a speech waveform and its recognized transcription. This project will compare two signal processing algorithms used in recognition of telephone speech. One standard, widely accepted signal processing algorithm, cepstral mean subtraction, motivated by computational efficiency and high performance, was evaluated. Another newer signal processing algorithm, RASTA, motivated primarily by auditory physiology was compared to cepstral mean subtraction in terms of word error. The two algorithms performed comparably in tests. CMS produced a 22% word error and RASTA produced a 25% word error on this difficult task. This project is significant from a biomedical point of view because telephone speech recognizers can be used to aid the handicapped. For example, the deaf could use a speech recognition system to aid them in their telephone communication with hearing people.

Contents

1 Glossary

Definitions of speech recognition terminology

1. Cepstra - A feature vector parameterization of speech waveform data motivated by high performance and relatively low computational cost of implementation
2. CMS - (Cepstral Mean Subtraction) an algorithm to subtract out steady-state spectral features in speech, like a phone channel
3. RASTA - (RelAtive SpecTrA) an algorithm to filter out steady-state spectral features in speech, like a phone channel
4. phoneme - A speech sound or sub-word unit, e.g. the phonemes corresponding to the transcription of the word ninety are the phonemes, n, ay, n, t, and iy
5. context - An adjacent phoneme, e.g. left context of ah in fox is f and right context is k
6. word error - A quantitative measure of the difference between the actual transcription of a waveform and its recognized transcription
7. corpus - A large collection of sampled acoustic speech data and their corresponding time-aligned word transcriptions
8. frame - The interval of time (20ms) one feature vector represents in the sequence of feature vectors representing an utterance; a chunk of speech

2 Mathematical Symbols

- $s(t)$: acoustic speech signal
- $S(w)$: frequency spectrum of speech
- $|S(w)|$: magnitude of frequency spectrum of speech
- $n(t)$: uncorrelated additive noise
- $N(w)$: uncorrelated additive noise spectrum
- $y(t)$: speech signal observed
- $Y(w)$: frequency spectrum of observed speech signal
- $C[n]$: feature vector representation of a frame of speech
- $\hat{C[n]}$: feature vector representation of a frame of speech
- $\bar{C}$: average feature vector representation of speech

3 Introduction and Background

Perhaps the most significant problem that limits speech recognition systems today is the effect of a variable acoustic environment. A noisy telephone channel is an example of a variable acoustic environment. Currently, in uncorrupted acoustic environments, a continuous speech speech recognizer can achieve a recognition accuracy of 95% on a 5000-word vocabulary task. However, in a telephone channel environment, the same recognition system suffers an increase in error rate that is unacceptable for most applications [10]. It is important to reduce the adverse effects of these adverse acoustic environments so that speech recognition applications over telephone lines and other types of noisy channels can be realized.

It is the goal of this project to quantitatively measure the accuracy of a speech recognition system to recognize natural number utterances such as "three thousand lira" recorded over a long-distance telephone line. This project will compare two signal processors used in preparing speech for recognition. The first signal processor, cepstral mean subtraction (CMS), is a standard, widely accepted signal processing algorithm for channel compensation motivated by computational efficiency and simplicity. CMS subtracts out steady state quantities in speech waveforms that do not contain information relevant to speech recognition. The second signal processor, RASTA, is newer and motivated primarily by auditory physiology. It also removes steady-state quantities in speech, but is potentially more useful for a time-varying channel, such as a phone channel.

This project is significant from a biomedical point of view because telephone speech recognizers can be used to aid the handicapped. Such a speech recognizer could be used by handicapped people who have limited use of their hands. In such an application, the speech recognition input to a computer could substitute for keyboard input. For the deaf, one can imagine using a speech recognition system to aid them in telephone communication with hearing people.

3.1 Biomedical Motivation for Speech Recognition

There are many biomedical motivations for developing speech recognizers that would recognize accurately over phone channels. Currently, the deaf community needs a TDD (telecommunication device for the deaf) to interact on the phone (see appendix for details). A home computer telephone interface for the deaf could consist of a speech recognition system and, optionally, a speech synthesizer. The incoming speech received over a telephone channel would be recognized by the recognition system and displayed on a monitor. If return speech were not possible, the receiver would enter a response with a keyboard and the speech synthesizer would synthesize the speech waveform to be sent back over the telephone line. Advances made in this area of speech recognition could also lead to the development of other aids for the handicapped. In the workplace, recognizers are implemented to help the handicapped reduce the number of keystrokes necessary to

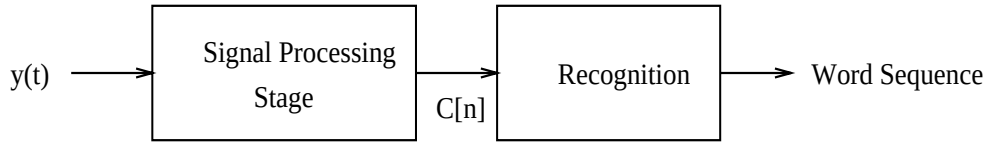


Figure 1: Basic steps in speech recognition.

complete a given task on a computer [1]. A difficulty with such systems is that those aids for the handicapped must be recalibrated when the user moves to a new location. In addition, the system might malfunction in the presence of acoustic noise from equipment. A speech recognizer that could compensate for a channel and reduce the adverse effects of additive noise on the recognizer performance would make these systems more portable, increasing the mobility and freedom of the human user.

3.2 Overview of Speech Recognition

Most automatic speech recognition systems operate in two steps (see Figure 1). In the first stage, the signal processing stage, a speech signal, $y(t)$ is recorded by a microphone transducer, sampled in time, and compressed using signal processing techniques. The signal processing stage converts the acoustic waveform into a sequence of feature vectors $C[n]$, which it passes into the second stage, the pattern recognition stage. The second stage, has apriori knowledge about the averages and distributions of feature vectors for different speech sounds, or phonemes. This signal behavior of the average and distribution of each phoneme is captured in a statistical model during the training phase of speech recognition. The pattern recognizer compares the observed feature vector information, $C[n]$ with these statistical models of different phonemes. Given these statistical speech models, the "closest match" to the feature vector representation of speech is the most probable word sequence determined by the recognizer [12]. The sequence of phonemic models that is statistically "closest" the feature vector sequence corresponds to a word sequence, which is chosen as the output word sequence.

3.2.1 Signal Processing Stage

This project will focus on the signal processing stage which serves two purposes. First, it compresses the data into feature vectors (see Figure 2) so that it can be characterized in fewer parameters. Second, it transforms the acoustic data into a feature space [12] that facilitates the discrimination of phonemes. For instance, the highest and second-highest energy frequency components in an acoustic waveform might be two features that form a basis for phoneme distinction. Two figures illustrate this concept. In Figure 3, the vector representations in the feature space distinguish one

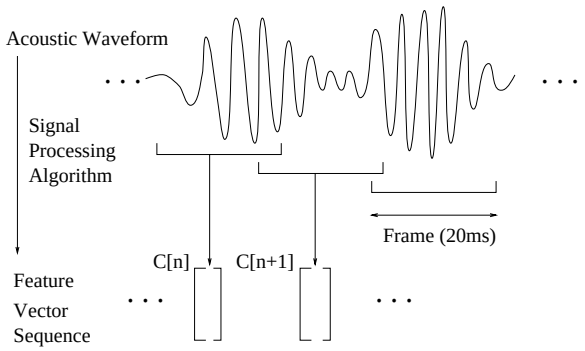


Figure 2: Illustrating the process of breaking a speech signal into feature vectors.

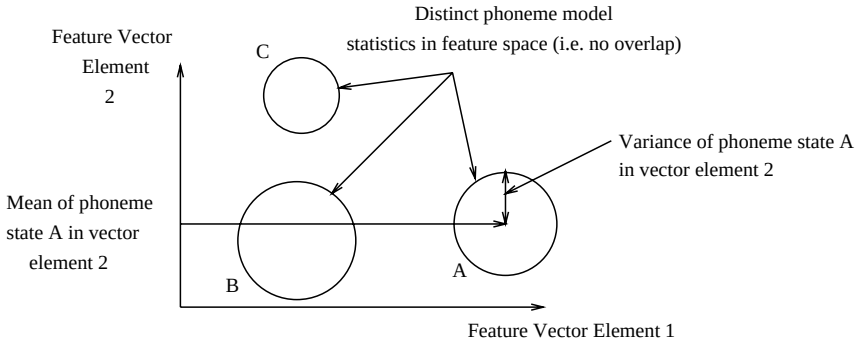


Figure 3: Illustrating the concept of a feature vector space in terms of feature vector elements.

phoneme state from another. In Figure 4, the vector representations in the feature space are not distiguishable based on their feature vector representation. This is because there are ambiguous regions of overlap. In the figure, an A could be classified as a B if the observation of A were in the overlap region and vice-versa. It is therefore desirable to parameterize the acoustic waveform into a feature vector space similar to that in Figure 3 in which phoneme features are distinct to each phoneme. If the location of a phoneme in the feature space distinguishes it from other phonemes, it will not be confused with those phonemes as often, which diminishes error rate. This project will focus on choosing a signal processing algorithm that best distinguishes phonemes recorded over telephone lines.

3.2.2 Modeling Phoneme Acoustic Features and Context

Once the acoustic waveform has been parameterized into a sequence of feature vectors, a statistical model of each phoneme is constructed to represent it in terms of a sequence of feature vectors.

Monophones and Triphones Monophone models are phonemic models of speech which are independent of context, or adjacent phonemes. The mathematical model of a monophone is only dependent on the current phoneme. Triphone models are phonemic models of speech whose statistics

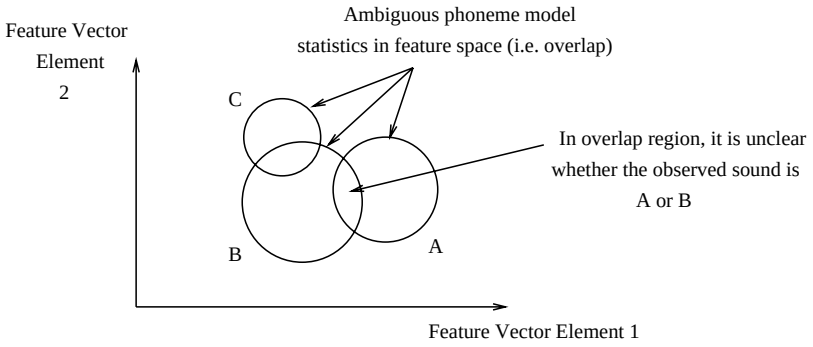


Figure 4: Illustrating an ambiguous feature space where some phonemes are confusable with others.

Table 1: Context-independent (Monophone) versus context-dependent (Triphone) models.

Sample Word	Context-independent (monopone models)	Context-dependent (triphone models)
PAM	(p)–(ae)–(m)	(sp–p–ae)–(p–ae–m)–(ae–m–sp)
PAT	(p)–(ae)–(t)	(sp–p–ae)–(p–ae–t)–(ae–t–sp)
	Models of phoneme ae are the same.	Models of phoneme ae are different because of different phonemes following ae.

are dependent on context. Triphone models depend on phonemes occuring to the immediate left, to the immediate right, or both left and right of the current phoneme. An example of the difference between context-dependent triphones and context-independent monophones is summarized in Table 1. The context-dependent triphones depend on both the left and right context of the current (middle) phoneme. The monophones are defined by the current phoneme.

Statistical and Linguistic Motivation for Context Dependence The pronunciation of a phoneme is colored by the preceding and succeeding phonemes, which causes a feature vector parameterization of a phoneme to have large variances from the mean feature vector representation of that phoneme at transitions from phoneme to phoneme. In this project, phonemes are each broken down into three states, the beginning, middle and end. This context-dependence causes overlap of monophone model parameter distributions in primarily beginning and end states. To keep overlap of model parameters as small as possible, a triphone model for each distinct left and right context can be constructed. Each new triphone model will have a tighter variance than its associated context-independent monophone model as a whole, making the context-dependent triphone models better discriminators.

Table 2: A typical word and its phonemic transcription.

Word-level transcription	Phoneme-level transcription
Ninety	(n) (ay) (n) (t) (iy)

3.2.3 Grammar and Pronunciation Network

- Grammar : A grammar dictates the possibility of all recognized word sequences. A word-pair grammar dictates what words may immediately follow other words in a recognized transcription. For instance, in a grammar recognizing numbers, a possible constraint could be, "thousand" or "million" may immediately follow "hundred", but "hundred" may never immediately follow "thousand."
- Pronunciation Network : A pronunciation network is a dictionary of phonemic transcriptions corresponding to a word. The pronunciation network determines what sequence of phonemes can make up a word. A pronunciation network contains all the words of a corpus and their corresponding phonemic transcriptions. Table 2 presents a typical word and its phonemic transcription.

3.2.4 Training the Recognizer

The goal of training is to create a statistical model of each phoneme based on a large number of observations of a phoneme and its corresponding feature vector parameterization. All of the training data corresponding to a certain phoneme are used to estimate the statistical parameters of each phoneme's mathematical model.

The recognizer is trained to identify a phoneme by first computing the mean feature vector associated with all occurrences of that phoneme and then computing the variance, or spread, of that phoneme in space associated with the feature vector parameterization produced by the signal processing stage. This training determines the distribution of phoneme over a region in the feature vector-parameterization space.

Iterative training can be performed to get better and better estimates of the statistical distribution of each phoneme. The statistics can usually be represented as a sum of Gaussian distributions, or mixtures. Increasing the number of mixtures used to represent statistics of a phoneme help to better estimate the true distribution of the phoneme because a phoneme's statistical distributions in the feature space may not be Gaussian, but rather have humps which can be better approximated by a sum of Gaussians.

Table 3: Comparing performance of state-of-the-art recognizers on uncorrupted and telephone speech.

Research Site	Type of Recognition	Word Error
SRI	Uncorrupted Speech	10.3%
SRI	Long-Distance Telephone Speech	22.5%

3.2.5 Recognizing a Word Sequence

The recognized transcription of a waveform is a "guess" at what word sequence corresponds to a given acoustic waveform. The computer uses statistics to "guess" what the underlying word-level information is by first breaking down the word level information to a level that can be compared to the acoustic information.

An overview of speech recognition sequence matching is shown in Figure 5. First, the possible word sequences given by the grammar are constructed in computer memory. An example word sequence consisting of three words, Word 1, Word 2, and Word 3 is shown as one possible word sequence. Second, each word sequence is broken down into a sequence of phonemes with the pronunciation network. In the diagram, Word 1 has 3 phonemes, Word 2 has 4 phonemes and Word 3 has 2 phonemes. The phonemes are further broken down into their mathematical models. For illustration purposes, each mathematical model in the diagram has three states. The states represent the breakdown of a phoneme into its beginning, middle and end in time. The acoustic waveform is parameterized into a sequence of feature vectors in the signal processing stage. It is the feature vector sequence that is compared to the state sequence in recognition. The different length arrows at the matching level show that matching can be good or bad in different places. In the diagram the feature vector sequence that is the closest to the overall state sequence statistically is chosen as the best match. This best match at the state level corresponds to a word sequence. That word sequence is the recognized transcription of the utterance.

3.3 Issues in Telephone Speech

3.3.1 Effects of Telephone Speech on Recognition Performance

Reported results for recognition of telephone speech demonstrate the harmful effect telephone channel corruption has on speech recognizers. The 1994 ARPA Spoken Language Program Benchmark Tests show that the word error can double in the presence of a long-distance phone channel. The results of these tests are summarized in table 3.

It can be seen from table 3 that the error rate more than doubles for telephone speech recognition. These results showing the degradation of performance on telephone-corrupted data are not exactly

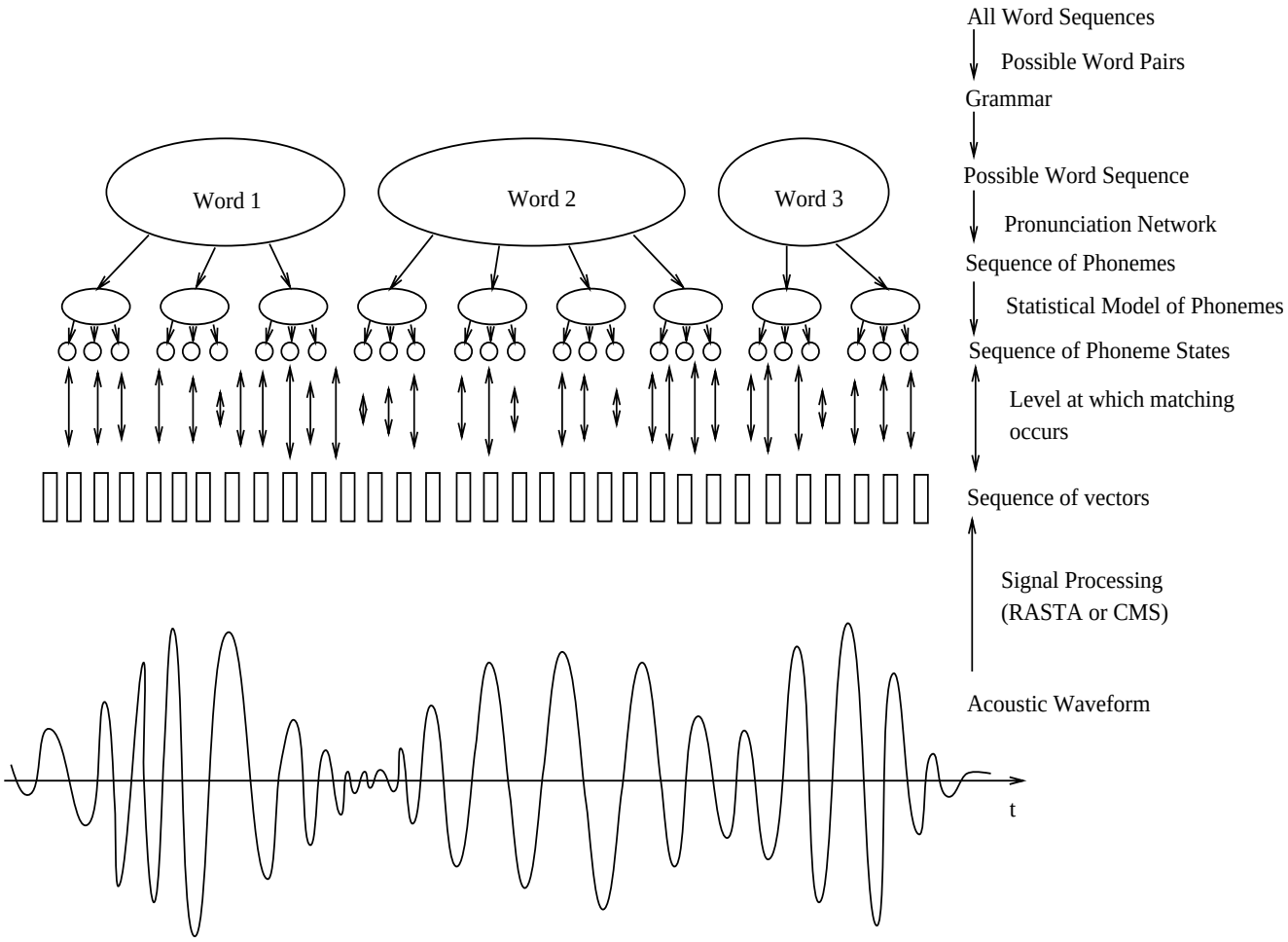


Figure 5: Recognition occurs at the feature vector level through multiple hierarchical constraints.

comparable to the results obtained in this project. The task summarized in table 3 was a larger vocabulary and the recognizers used more sophisticated channel compensation techniques as well as more sophisticated statistical language modeling techniques than those used in this project. The results in table 3 are for the same task except that there were errors due to human factors in the telephone speech. Such factors make the telephone task slightly more difficult, but are not completely responsible for the doubled word error. These results do strongly suggest that the task of recognizing long-distance telephone speech, such as the speech used in this project, is more difficult than recognizing uncorrupted speech.

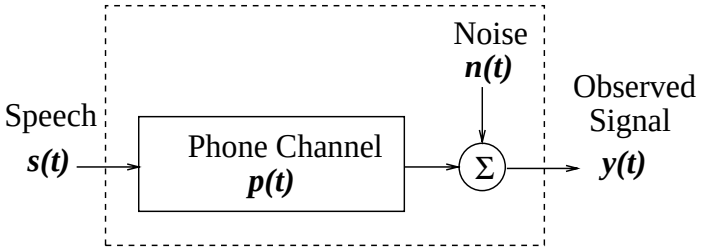


Figure 6: Block diagram model of telephone speech.

3.3.2 Mathematical Model of Channel Distortion and Additive Noise

The particular problem of interest in this project is improving speech recognition over a noisy telephone channel. In this project the phone channel is modeled as a linear, time-variant system with additive noise (see Figure 6). The total phone channel model in the diagram is boxed with a broken line. There are two different sources of corruption in the modeled telephone, additive noise and channel distortion. Mathematically, if a speaker utters $s(t)$,

$$y(t) = s(t) * p(t) + n(t) \tag{1}$$

where the observed speech is $y(t)$, the additive noise corresponds to $n(t)$, and the channel distortion corresponds to $p(t)$, or the linear system that the speech waveform passes through. Both of these effects degrade recognizer performance. For instance, even in uncorrupted speech, the microphone used to record a speaker's voice is a "channel." Every channel has its own input/output relationship, and if it is linear time-invariant, the observed speech signal is the convolution integral of the impulse response of the channel with the speech signal the speaker uttered. In telephone lines, the channel characteristics vary slowly in time, so a phone channel is not time-invariant [3]. For short utterances, some phone channels may be considered time-invariant. The variability of telecommunications channels makes recognition over the phone especially difficult for recognizers [10]. One reason that these effects make recognition difficult is that all telephone channels $p(t)$ have different characteristics and the noise, $n(t)$, is also unknown.

3.3.3 Motivation for Cepstral Mean Subtraction to Remove Channel Distortion

Although a standard procedure for noise subtraction and channel compensation exists for $n(t)$ known and $p(t)$ known, a standard algorithm does not exist for $n(t)$ unknown and $p(t)$ unknown. In this case, they must each be estimated during the utterance, which is difficult because $s(t)$, $p(t)$, and $n(t)$ must be separated. The most effective method of channel compensation to date has been cepstral mean subtraction (CMS). In CMS, the average value of each cepstral feature vector element

in the observation is computed over utterance length and then used to normalize the channel that the observation is passed through. The mathematical motivation behind CMS is derived below.

If $n(t) = 0$ in Figure 6, the only phenomenon corrupting the speech, $s(t)$, is the channel distortion, $p(t)$. Taking the Fourier Transform of equation 1, we have

$$Y(\omega) = S(\omega)P(\omega) \tag{2}$$

$$\log|Y(\omega)| = \log|S(\omega)P(\omega)| \tag{3}$$

$$C_y = \mathrm{IFT}(\log|S(\omega)P(\omega)|) \tag{4}$$

$$C_y = \mathrm{IFT}(\log|S(\omega)|) + \mathrm{IFT}(\log|P(\omega)|) \tag{5}$$

(where IFT is the Inverse Fourier Transform and $|x|$ denotes the magnitude of x)

$$C_y[n] = C_s[n] + C_p[n] \tag{6}$$

where $C_y[n]$, $C_s[n]$, and $C_p[n]$ are the cepstral coefficient feature vector representations of the windowed $y(t)$, $s(t)$, and $p(t)$, respectively (see Figure 2). The channel estimate is the average value of the sequence of cepstral feature vectors over the utterance. For an utterance that contains a sequence of L cepstral feature vectors,

$$\text{Channel Estimate} = \bar{C_p} = \frac{1}{L}\sum_{i=1}^{L} C_y[i] \tag{7}$$

where $\bar{C_p}$ is the estimated cepstral feature vector representation of the channel, or the contribution of the linear system, $p(t)$, in the cepstral domain if it is time-invariant. The observed speech can then be normalized by subtracting the channel estimate from each cepstral feature vector in the utterance, effectively removing the effect of channel distortion from the utterance

$$\hat{C_s}[n] = C_y[n] - \bar{C_p}, \tag{8}$$

where $\hat{C_s}[n]$ is the normalized sequence of cepstral feature vectors. The sequence $\hat{C_s}[n]$ is an estimate of the feature vector representation of $s(t)$ with no channel distortion. The advantage of this algorithm for channel compensation is its effectiveness versus computational cost. Compared to signal processing algorithms that do not incorporate this mean subtraction, error rates decrease with the implementation of a mean subtraction. There is an increase in performance for utterances observed in a channel as well as for utterances observed in additive noise. However, the performance increase is greater for channel distortion than for that of additive noise. This algorithm is less useful for slowly-varying channels such as real phone channels. This is because the channel estimate is one constant feature vector in time, and can not vary as the channel varies in time. Computation of an average channel estimation over an utterance converges in approximately 50 words, although it is still useful for fewer words [11]. For the recognition application in this particular project, the

Figure 7: Figure showing the convergence of a channel estimation from Lerner [11].

utterances are on the average smaller than fifty words. These utterances contain approximately 6 words per utterance and the channel estimation may not converge completely during the utterance (see Figure 7). Thus, this technique may not be as effective for this task.

3.3.4 Motivation for RASTA

Time-filtering of feature vectors of the observation with RelAtive SpecTrA (RASTA) might better compensate for the channel if it is not time-invariant. The RASTA method estimates the channel based on its recent history and performs estimation with a causal filter rather than with a mean-subtraction (which requires a non-causal filter). For every frame of the observed waveform in time, there is a feature vector associated with that frame. RASTA is a method that filters the sequence of feature vectors. The RASTA technique band-pass filters each element of the feature vector with a fifth-order filter.

$$M(z) = \frac{z^4(0.2 + 0.1z^{-1} - 0.1z^{-3} - 0.2z^{-4})}{1 - 0.94z^{-1}} \tag{9}$$

The high-pass portion of the filter alleviates the effect of the channel. This filtering of time-trajectories of features of the cepstral feature vectors more closely matches the way a human ear processes speech [7]. Likewise, it can more closely "follow" a channel characteristic as it varies in time because it is not a static estimate over an utterance. This ability to deal with time-invariance compared to CMS is illustrated diagramatically in Figure 8. For utterances recorded over long-distance telephone lines with a signal-to-noise ratio of greater than 20dB in an isolated-digit, 11-word vocabulary recognition task, an increase in performance from 40.3% without RASTA preprocessing to 80.1% with RASTA was reported [5]. One drawback of RASTA is that experimental results reported by Singer and Paliwal show that RASTA processing hurts performance for context-independent monophone models [16]. RASTA filtering in this project was performed on PLP feature vectors, which is a feature vector parameterization similar to cepstra that is motivated by auditory physiology (see appendix). In this paper, RASTA PLP signal processing is referred to simply as RASTA.

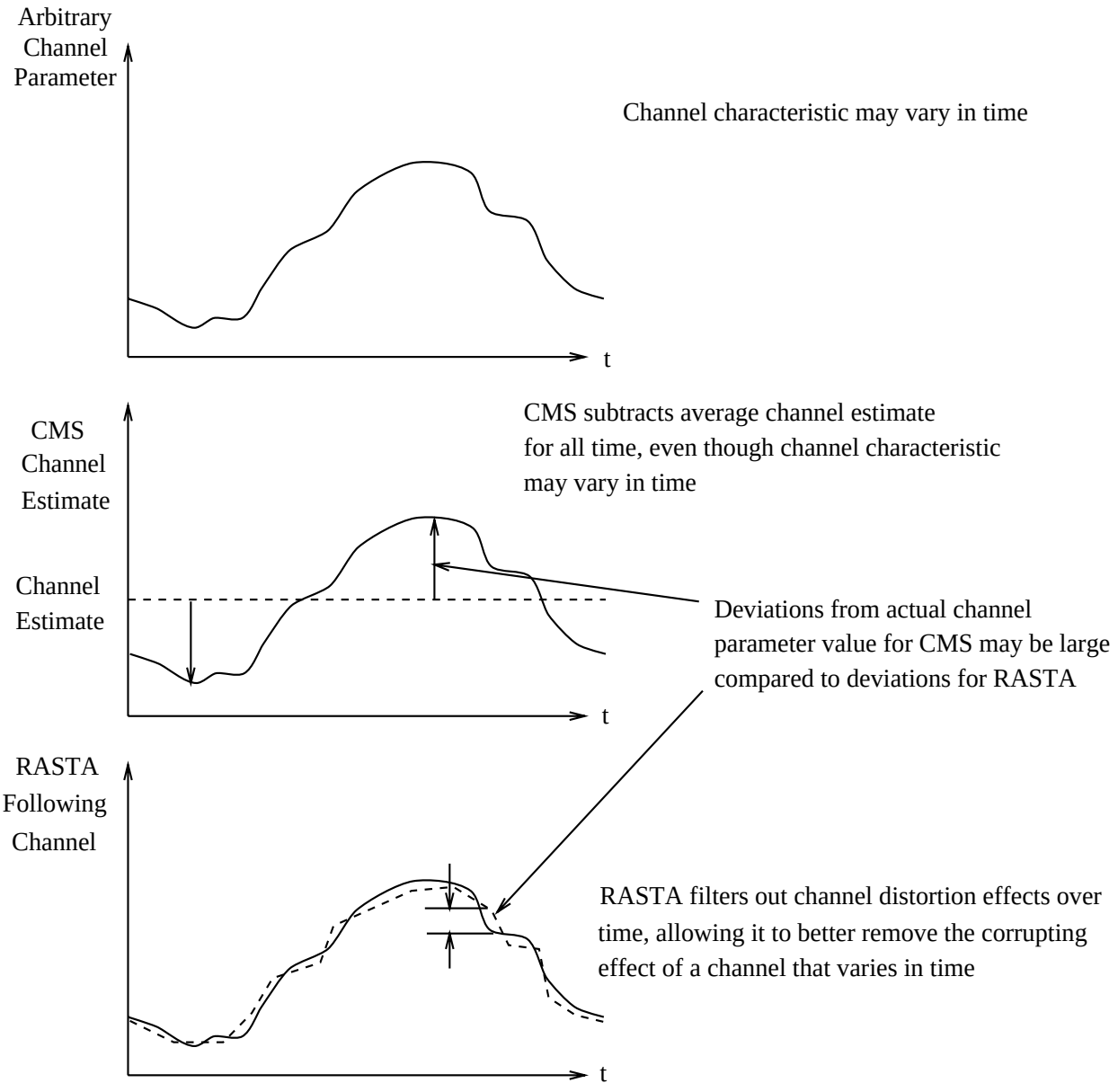


Figure 8: The advantage of RASTA in a time-variant channel.

3.4 Objectives and Project Summary

3.4.1 Objectives

It is the goal of this project to quantify recognition performance of telephone speech for a real task, such as recognizing natural numbers. To answer this question, the natural numbers section of the Macrophone corpus recorded over long distance telephone lines was used to train and test each recognition system. First, the word error of context-dependent (triphone) models versus context-independent (monophone) models was measured using the standard, widely accepted CMS signal processor. The CMS performance was then compared to results obtained in the same way for RASTA performance to determine which signal processor performed better on long-distance telephone data. It has been reported [16] that the RASTA signal processor hurts performance with monophone models. That was a finding that this project also aimed to confirm quantitatively.

3.4.2 Project Summary

A histogram of the frequency of utterances versus their SNR was first constructed to determine the global noise level in long-distance telephone speech. It was found that more than 95% of the SNR distribution in this telephone speech data falls in the range 20-80dB, which is not typically "noisy" speech. This observation about telephone additive noise suggests the following:

- The global SNR over the entire utterance is not as important as the local SNR's in the neighborhood of corrupted frames (i.e. the noise is intermittent and should be detected and dealt with adaptively).
- Telephone speech is not severely degraded in terms of global SNR. Thus, noise compensation techniques on a global level will probably not yield as large gains in recognition performance as channel compensation techniques.

This discussion motivated initial experimentation to compensate for the phone channel before attempting to compensate for additive noise effects because the global additive noise was not found to lower the SNR significantly for telephone speech. Two channel compensation algorithms, Cepstral Mean Subtraction and RASTA were implemented as channel-compensating signal processors in the recognition system. The two algorithms were both evaluated based on their measured word errors (see appendix). The word errors for both signal processors are summarized in 4. It can be seen from the table that the lowest error is achieved with the CMS channel compensation algorithm.

Singer and Paliwal's [16] finding that RASTA hurts performance for monophones was confirmed. Appreciably lower word error rates were observed for CMS than for RASTA.

Table 4: Comparing word error of RASTA and CMS for best recognition results.

	RASTA triphones	CMS triphones
Word error	26%	22%

4 Specific Methods

4.1 Experimental Paradigm

In this section, the speech data and software used to analyze and recognize the data will be discussed. The **Macrophone natural numbers corpus** chosen for experimentation is introduced first. The National Institute of Science and Technology **(NIST) SNR computation software** as well as the scope of the **Hidden Markov Model Toolkit**, the recognition software, is summarized. Finally, the evaluation criteria, the **word error** is defined. It will be used to measure performance for every experimental result.

4.1.1 The Macrophone Natural Numbers Corpus

The Macrophone Speech Corpus [17] is a collection of speech waveforms collected over long-distance telephone lines. The acoustic speech waveforms were sampled at 8kHz. This is a standard speech corpus available through the Linguistics Data Consortium. The Macrophone Corpus will be used to evaluate experiments. In particular, speaker-independent, continuous-speech, natural number recognition experiments will be performed with the Natural Numbers section of the corpus. The Natural Numbers of the Macrophone Corpus consists of over fifteen thousand utterances describing quantities (e.g. "thirty-six lira" or "ten ounces"). Each utterance consists of approximately 6 words, with at most 18 words per utterance. The corpus vocabulary is 530 words. The task is recognition of real long-distance telephone speech on a 500 word vocabulary.

4.1.2 NIST SNR Computation Software

The NIST standard SNR computation software was used to measure the SNR in all utterances. The speech was first segmented into non-speech (noise) and speech. The power in the speech and the power in the noise was computed by the software. The SNR was computed in dB based on average speech and average noise power.

Table 5: Possible types of speech recognition errors.

Transcription Errors	Word 1	Word 2	Word 3	Word 4
None	I'm	making	**the**	bread
Deletion	I'm	making	-	bread
Substitution	I'm	making	the	**bed**
Insertion	I'm	making	the **uh**	bread

4.1.3 Hidden Markov Model Toolkit (HTK)

The Hidden Markov Model Toolkit is a toolkit for building continuous density Hidden Markov Model (HMM) based pattern recognizers. It is primarily intended for building sub-word based (phoneme-based) continuous speech recognizers and can be used in a wide range of pattern classification problems. The toolkit includes signal processing functions, speech model training and testing tools, language modeling support and scoring software [8].

4.1.4 Quantification of Evaluation Criteria: Word Error

The word error is given by

$$\text{Percent Word Error} = (D - S - I)/N \tag{10}$$

where N is the total number of words in the reference transcriptions. In word error, D is the total number of word deletion errors, S is the total number of substitution errors, and I is the total number of insertion errors (see Table 5). The word error will be used as the evaluation criteria [8]. The word error, therefore, is a metric to evaluate the performance of a recognition system. The word error is zero for perfect word recognition and has no upper bound because a recognition system can insert an arbitrary number of words, driving I to a value greater than N. The word error is a widely accepted metric for quantifying recognition performance because it measures all types of word-level errors that can occur in recognition, namely deletions, substitutions and deletions.

4.2 Recognition System Design

In this section, the methods will be described in the sequence in which they were each categorically implemented. A flowchart of the steps involved in training is shown in Figure 9.

Data Preparation First, the Macrophone natural numbers speech waveforms were formatted for use with the SNR computation and signal processing software. The Macrophone natural numbers speech waveforms were then characterized in terms of their individual SNR, computed with

the **NIST SNR** computation software. The speech acoustic waveforms were then each converted into a sequence of numerical **feature vector parameterizations**.

It is then necessary to edit the transcriptions into **speech labels** that the recognition and scoring software can use. Using the Macrophone natural numbers transcriptions of speech, a word sequence constraint called a **grammar** is imposed on the recognizer. Because the recognition in this project is based on details of individual speech phonemes, word level transcriptions are not sufficient to train the speech phoneme statistical models. The Macrophone natural numbers word transcriptions were broken down to phoneme-level transcriptions using a **pronunciation dictionary** received from BBN (Bolt Beranek and Newman).

Model Topology and Training Once the speech waveforms are processed into feature vector parameterizations, and the words are broken down into phonemes, a **Hidden Markov Model topology** must be chosen for the phoneme set used in the pronunciation network. Given the phoneme set, statistics of how each phoneme is distributed across speakers and utterances are computed because not every phoneme always sounds exactly like every other observation of that phoneme. The mathematical model parameters are then estimated based on the statistics of many observations of a certain phoneme and its associated feature vector parameterization. This **training** phase is improved by modifying the phoneme's statistical model structure. Modifications to the distribution of a phoneme may improve performance in some cases where statistics are hard to estimate because the sample size is too small. If the number of observations of a phoneme is small, a variance associated with that phoneme may vanish. This causes mathematical problems in recognition. A **variance floor** may be introduced to help diminsh this effect. Grouping of models with similar statistical properties also helps to make the training phase more robust where there may not be enough observations of a phoneme to get a good estimate of its statistics. This is called **tying and clustering**.

Testing Recognizer Performance Once the phoneme model parameters have been estimated, the pattern recognizer can begin recognizing unknown speech. A sequence of models can be compared to the vector parameterization of an unknown speech waveform. This comparison can be "scored" to determine what sequence of models, and thus sequence of words, best matches the feature vector parameterization of that speech waveform. This is a **test** of how well the feature vector parameterization represents the salient "features" in the speech waveform. The closest statistical match between the feature vector sequence corresponding to the unknown speech and the word sequence given by a sequence of phoneme models is output as the recognized transcription. This recognized transcription can then be compared to the real transcription of the unknown speech waveform to calculate a quantitative measure of the performance of the recognizer.

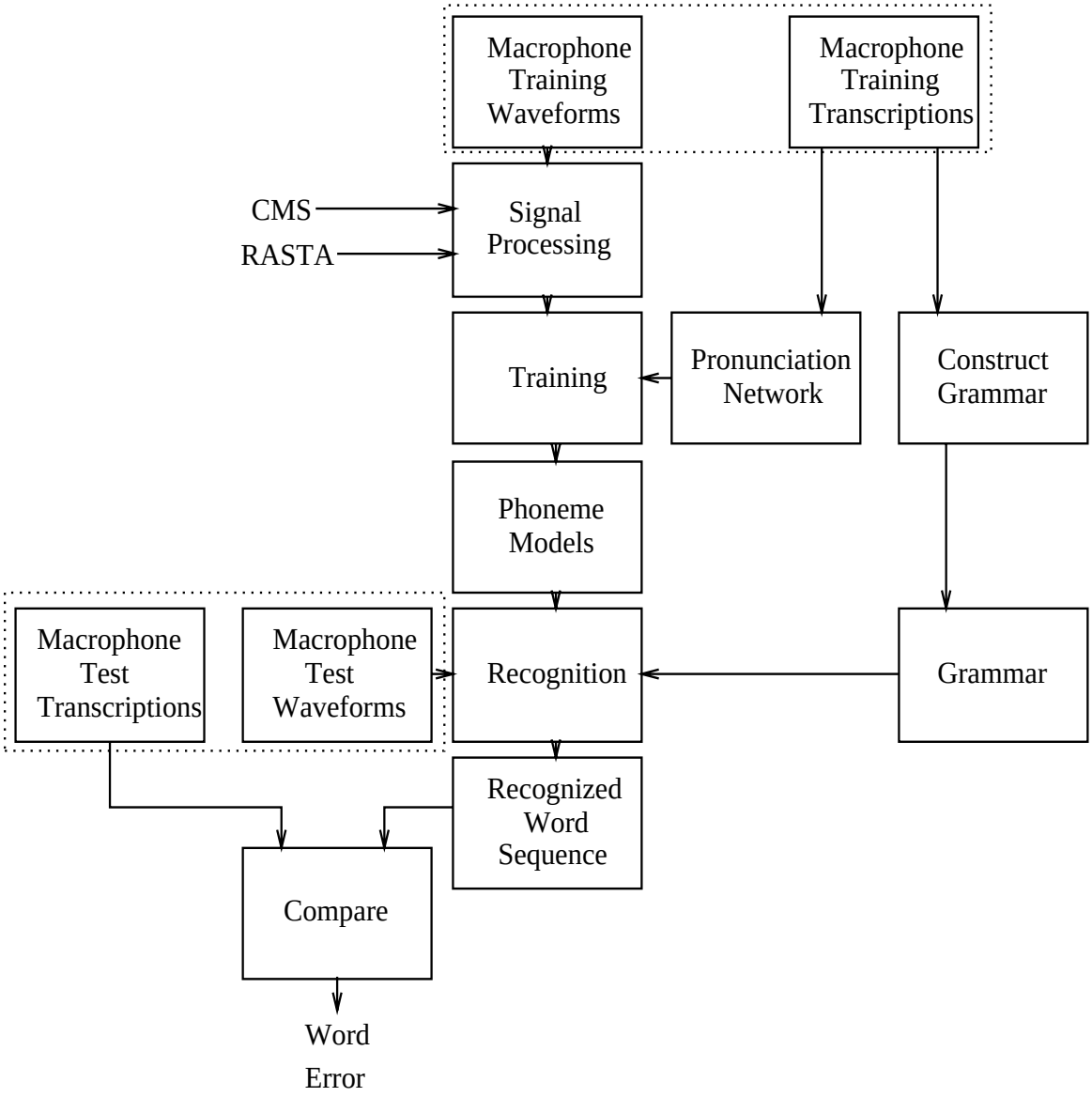


Figure 9: Illustrating steps required in this speech recognition paradigm.

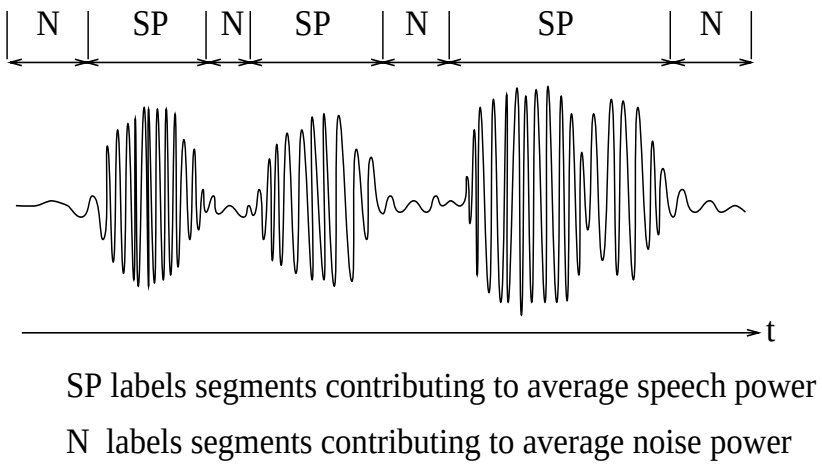


Figure 10: Marking noise segments and speech segments to compute SNR.

4.2.1 Computation of SNR with NIST software

The SNR of each utterance is calculated by measuring the average power in speech and the average power in noise and computing the ratio of average speech to noise power. To perform the average power in speech computation, the speech segments were detected

using a simple speech detector (described in the appendix). Given the marked speech segments, the average power across those segments was measured. The non-speech segments were labelled noise, and the power in those segments was measured similarly (see Figure 10). The SNR is defined as:

$$\mathrm{SNR} = 10 \log \frac{\mathrm{Power}_{\mathrm{SpeechSegments}}}{\mathrm{Power}_{\mathrm{NoiseSegments}}} \tag{11}$$

A histogram of the frequency of utterances versus the SNR was constructed using the list of SNR's computed with the software.

4.2.2 Feature Vector Parameterization of Acoustic Data

The purpose of parameterizing the sampled speech data into feature vectors is to compress the data and extract the information relevant to speech recognition from the acoustic waveform. The information extracted from the acoustic waveform is usually a numerical representation of certain features in speech, such as frequency content, or power. The sampled acoustic speech waveform is chunked into 20ms lengths, called frames, which overlapped each following frame by 10ms (see Figure2). The data was first parameterized as CMS feature vectors and then as RASTA feature vectors (the particular coding steps are archived in the appendix). The feature vector parameterization was performed using, for CMS, HTK libraries, and for RASTA, Nelson Morgan's mrasta code. The RASTA feature vectors are actually RASTA PLP feature vectors. PLP is a type of feature vector parameterization that aims to transform the observation into distinguishable speech

features in a way similar to cepstra. Throughout the text, RASTA PLP feature vectors are referred to simply as RASTA feature vectors. The training procedure described below was performed for both feature vector parameterizations.

4.2.3 Grammar

A grammar that constrains possible word sequences must be used to make nonsense word sequences unallowable. The grammar used in this project was a word-pair grammar, which is a listing of the words that can possibly follow other words. If a certain word is not allowed to follow another word by the grammar, that word-pair will never be recognized.

The typical format for a natural number utterance is a number or sequence of numbers, generally followed by a street name or type of unit. In order to take advantage of this simplicity, we use a word-pair grammar. In this project, the main focus was improving signal processing so the simple grammar was a better choice for studying signal processing. The word-pair grammar was developed using the 15016 utterance training data transcriptions. That grammar was expanded by grouping digits and digit modifiers. For instance, if the word "hundred" was found as a possible follower, the words "thousand" and "million" would also be added as possible word-followers. This was done because in the training although "five hundred" may be observed, "five thousand" might not be observed, but it is not because that is an unallowable word sequence. It is only because it did not occur, and it would be incorrect to rule it out.

4.2.4 Speech Labels

The speech labels corresponding to acoustic speech waveform data in the Macrophone Corpus were word-level transcriptions. In order to make the word-level transcriptions useable in the recognition paradigm the word-level transcriptions had to be converted to phonemic transcriptions. Using a monophone pronunciation network received from a company that researches speech recognition, BBN (Bolt Beranek and Newman, Cambridge), the word-level transcriptions were converted to phoneme-level transcriptions via word lookup in that pronunciation network. The monpohone pronunciation network was converted into a triphone pronunciation network as well so that words could also be broken down into triphones. A suitable set of monophones and triphones were chosen based on the phonemes occuring in the corpus (see appendix for list of phones). An example of a phonemic transcription (both monophone and triphone) for the word, "ninety" is presented in Table 6. A noise label was added to the Macrophone transcriptions where the speech was corrupted by additive noise (e.g. mouth noise caused by receiver interference). All noisy portions of utterances were used to train a noise model, "dumping" that corrupted acoustic data into a model that was not used in recognition, effectively removing it.

For the monophone and triphone experiments, one silence model, a begin utterance and end

Table 6: A typical word and its monophone and triphone transcription.

Word-level transcription	Monophone transcription	Triphone transcription
Ninety	(n) (ay) (n) (t) (iy)	(sp-n-ay) (n-ay-n) (ay-n-t) (n-t-iy) (t-iy-sp)

of utterance model was used to model the silence between the beginning of the utterance and the beginning of speech as well as the silence between the end of speech and the end of the utterance.

For the triphone experiments, an additional inter-word silence model was also used. This model was concatenated at the end of every pronunciation in the pronunciation network to model the brief silence between words.

4.2.5 Hidden Markov Model Topology Choice and Initialization

A 3-state HMM (Hidden Markov Model) was chosen. A 3-state topology separates each phonemic mathematical model into 3 temporal states. The states must be initialized with some initial condition, or "guess", at model parameters. In this project, each state distribution was initialized as a single Gaussian with zero mean and unity diagonal covariance. (see appendix for HTK and HMM details concerning structure and initialization). The initial triphone models were also 3-state, diagonal covariance HMM's. The triphone models were numerically initialized with the values of the 1 mixture monophone models after 4 iterations of reestimation.

4.2.6 Training Method

The goal of training is to create a statistical model of each phoneme based on a large number of observations of a phoneme and its corresponding feature vector parameterization. All of the training data corresponding to a certain phoneme were used to estimate the statistical parameters of each phoneme's mathematical model.

The phoneme models were trained using the Baum-Welch reestimation technique (see math appendix for formulae). The reestimation algorithm in the HTK tool, HERest was used to train all models. Models were trained by performing 4 iterations of HERest on all of the training data [8].

The number of mixtures is the number of Gaussians used to model a state. The more Gaussians that are used, the more accurate the sum of those Gaussians can represent the states (see Figure 11). After four iterations of HERest, the number of Gaussian mixtures used to model a state was increased by one.

To increase the number of mixtures by one, the n^{th} mixture weight is halved, and that new n^{th} mixture is copied to the $n + 1^{th}$ position. After copying the mixture, the n^{th} and $n + 1^{th}$ mixture are perturbed by adding 0.2 standard deviations from the mean feature vector to the n^{th} mixture

Figure 11: Illustrating advantage of estimation using more mixtures.

and by subtracting 0.2 standard deviations from the mean feature vector to the $n + 1^{th}$ mixture. The upmixing procedure was implemented with the HTK tool, HHEd (see appendix for specific edit commands). The number of Gaussian mixtures was incrementally increased in this way until five fully trained mixtures were used to model a state.

4.2.7 Tying and Clustering Triphones

Separating the monophones into triphones distributes the speech phoneme observations over many more models, making the amount of data available to train each model smaller. Not all triphone combinations will occur in the training data. Thus, only observed triphones can be modeled and trained for in any given set of training data. In this project, 1405 triphones occured in the training data. The conversion of monophones to triphones reduces the data available to train each model on because what data was used in the monophone case to train each model is now distributed over 1405 models. Some of the models would be trained on only one or two occurrences if a certain triphone only occured in one word that was very infrequently uttered throughout the corpus. To deal with this reduction in training data per model, the model parameters were clustered and tied. Tying means that a set of parameters can be linked to more than one model. (see Figure 12) (from Figure 17, User Manual [8]) shows an example of tying in a 3-state hidden Markov Model. In Figure 12, the center states are tied to the Gaussian distribution, C, and the adjacent distributions for each triphone model, L1 through L4 and R1 through R4 are not tied. In the example, only the middle state, C, is tied for illustration purposes. The tying procedure used in this experimentation was more general and made each state of every occuring triphone a potential tie point. To determine what state distributions were tied, states were clustered. In the multivariate space defined by the feature vector distributions of triphone models, similar phonemes will be clustered, or "near"

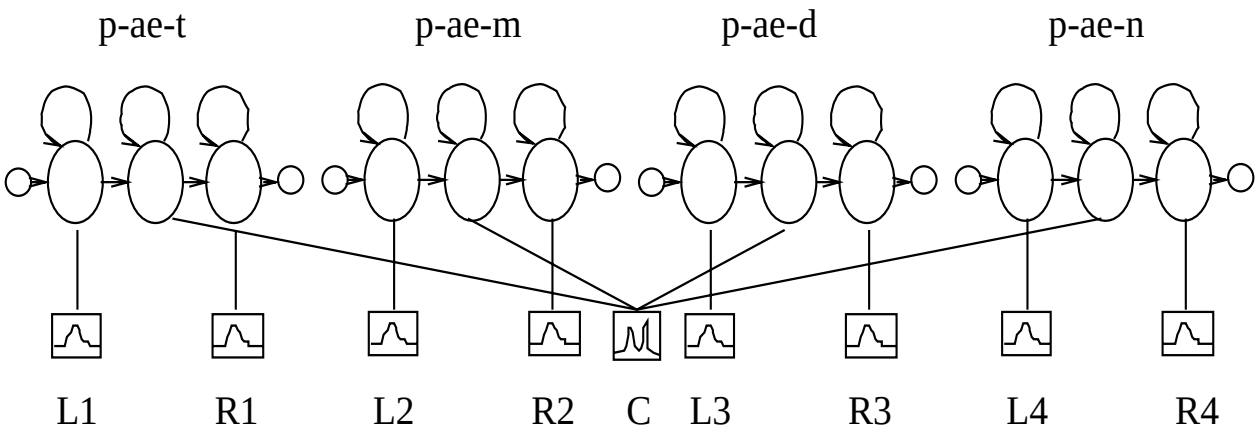


Figure 12: Illustrating tied states, adapted from the HTK User Manual, figure 17, [8].

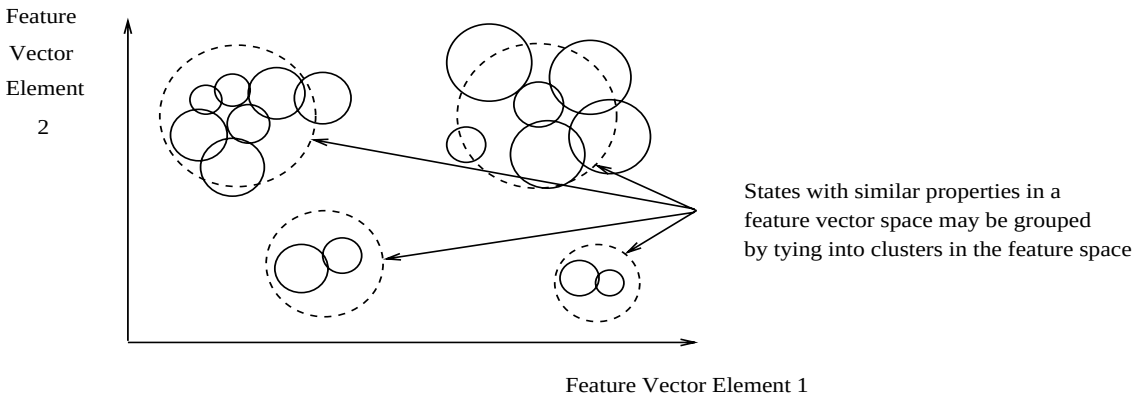


Figure 13: Diagram illustrating the concept of clustering.

each other. The 2-dimensional feature vector space in Figure 13 illustrates the concept of state clustering. The smaller circles within the large broken circles are the state parameters and the large circles are the state clusters. In this experimentation, there were more than 2 dimensions, but the concept can be expanded to any number of dimensions. The state distances are defined as a weighted distance between Gaussian distribution means. The distances between states are calculated, which determines how "far" one state is from another. Arbitrarily, "near" was defined as a multivariate normalized distance of 0.5 standard deviations, such that the average triphone model estimates that fell within 0.5 standard deviations of the cluster were included. Future experimentation with this distance could increase performance, and is a potential route for future experimentation. Clustering states ties their parameters in a general sense, which means after tying, all states in a cluster contribute to training the parameter feature vector associated with that cluster. This partially alleviates the problems associated with sparse training data because statistically similar models are grouped into one model that exemplifies the statistical properties of the whole group.

4.2.8 Variance Floors

A vanishing variance causes mathematical problems in recognition. The covariance matrix, a matrix of parameter variances, must be inverted to compute the score for any utterance. If any variance in the covariance matrix is zero, the matrix will be singular, and the recognition program will flag the problem with an error. In the four and five mixture stages of statistically clustered triphone training, variance floors were implemented (see appendix for numerical minimum variance feature vector). Clustering does not completely alleviate the problems associated with scarce training data. Once clusters are determined, the number of mixtures can be gradually increased. This worsens the problem of scarce training data because more parameters must be estimated from a scarce pool of data. To accommodate this possibility, the variances can be floored to some minimum value to ensure that variances will not vanish, even when the number of observations is very small. To pick the minimum variance feature vector, a subset of the estimated cluster model parameters were examined. A minimum variance was chosen such that it was less than 90% of the variance values observed in the subset. That variance feature vector was then reduced by an order of magnitude to ensure that it would be smaller than variances not observed in the subset. This variance floor choice was arbitrary and may be a route for future experimentation.

4.2.9 Testing the Trained Models

In testing the question is: given the trained recognizer and a waveform corresponding to some unknown word sequence, what sequence of words will most closely match that waveform's feature vector parameterization. To measure the performance of a recognizer, the question is: how close is that recognized word sequence to the actual transcription of that speech waveform?

To choose which word sequence most closely matches the waveform, the possible word sequences are first broken down into possible phoneme sequences. The possible recognized phoneme sequences and their corresponding models are "scored" against the sequence of feature vectors that represent the utterance. The statistical phoneme models are compared to the observed speech feature vectors and the sequence of models that minimizes the statistical distance to the sequence of speech feature vectors is the recognized phoneme sequence. The word sequence corresponding to that phoneme sequence is the recognized transcription. The feature vector sequences of the independent data were recognized using the recursive Viterbi algorithm (see math appendix for formulae). The Viterbi algorithm was implemented using the HTK tool, HVite. The recognized word sequence corresponding to each feature vector parameterization was then scored against the transcribed word sequence with the HTK tool, HResults. The word error was used as the evaluation criteria. To perform the testing, independent Macrophone utterances not used to train the models were then parameterized in terms of feature vectors in the same way that the training data were parameterized

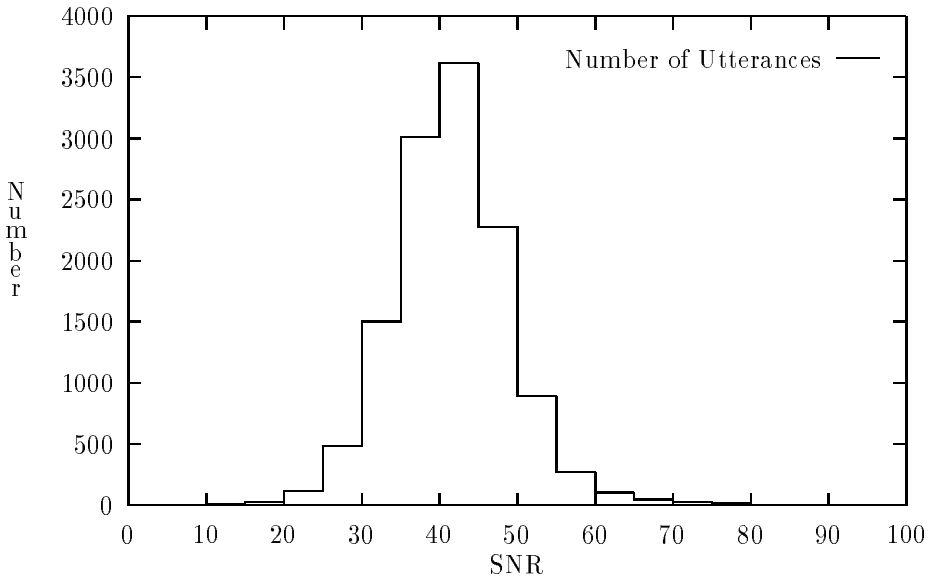


Figure 14: Histogram of SNR in Macrophone natural numbers telephone data.

[8]. These feature vector parameterizations of the independent data were recognized and compared to their reference transcriptions, calculating the word error.

5 Results

5.1 Histogram of Telephone SNR

The global SNR of each utterance was computed. The NIST standard SNR computation algorithm described in the appendix was used to measure the SNR of each utterance. It should be noted that this calculation was performed for more than ten thousand utterances, which took four days of pure computation on a SPARC-10. The number of utterances falling within certain SNR bounds is plotted against the SNR in Figure 5.1. More than 95% of the utterances fall in the range of 22dB or higher SNR. The utterances falling into the range of SNR less than 22dB were often corrupted by a 60 Hz hum or a static noise.

5.2 Presenting Results and Splitting Test Data

In presentation of results, tables comparing word error are used to summarize performance of a recognition system given certain experimental conditions. Such conditions tell whether the system used monophone models or triphone models, whether the system used CMS or RASTA processing or how many mixtures were used to estimate model distributions across phonemes. All end results are summarized for performance of the system based on independent data not used to train the

Table 7: Table of CMS monophone vs. triphone performance.

Signal Processor	Phoneme Type	Word Error
CMS	monophone	45%
CMS	triphone	22%

system, or Macrophone test data (see Figure 9). Using one set of test data is not always a reliable test because performance for another set of test data could produce higher or lower results. As a confidence check, the test data were split into two halves. The recognition performances for the first and second half were measured and averaged for each test below. In all measurements, the first and second half of test data produced performance indices within 2% of each other, which suggests that the measured performances were reliable. Thus, with a high confidence, the calculated performance values can be expected to deviate from true values by 2% or less.

5.3 CMS Monophone vs. Triphone Performance

This experiment was performed to quantify how much telephone speech recognition performance changed when the monophone models were changed to triphone models. This will be a measure of how much the incorporation of context-information changes recognition performance for the CMS algorithm.

The monophone CMS performance result was obtained by first training the monophone set with four iterations of parameter reestimation per mixture, where the number of mixtures was incremented from one to five. The trained CMS monophone models were then tested with independent data. The word error was calculated and is presented in table 7.

The triphone CMS performance result was obtained by initializing the triphone set with the monophone model parameters after four iterations of training on one mixture. Those triphone models were then trained with four iterations of reestimation, after which the model parameters were tied and clustered. The number of mixtures was incremented from one to five. The trained state-clustered CMS triphone models were then tested with independent data. The word error was calculated and is presented in table 7.

It should be noted that these experiments require powerful computer resources and require approximately two weeks of pure computation. On a SPARC 10 workstation, the monophone training required six days and the triphone training required a total of eight days. The testing phase required less time, approximately two days.

Table 8: Table of RASTA monophone performance vs. CMS monophone performance.

Signal Processor	Model Type	Word Error
CMS	monophone	70%
RASTA	monophone	82%

5.4 RASTA Monophones vs. CMS Monophones

This experiment was performed to quantify how much recognition performance differed with different signal processing algorithms using monophone models. This experiment was performed to test Singer and Paliwal's [16] finding that RASTA processing degrades recognition performance for monophones.

The monophone CMS performance result was obtained by training the monophone set with four iterations of parameter reestimation. The trained CMS monophone models were then tested with independent data. The word error was calculated and is presented in table 8.

The monophone RASTA performance result was obtained by training the monophone set with four iterations of parameter reestimation. The trained CMS monophone models were then tested with independent data. The word error was calculated and is presented in table 8.

The performance of RASTA monophones to CMS monophones is summarized in Table 8. This CMS result was based on only one mixture monophone models, not five mixture monophone models, which is why the CMS monophone result in table 8 is much lower than the CMS monophone result obtained in table 7 describing performance changes due to context information incorporation and a full training to five mixtures.

These results required additional training and testing of RASTA processed data. This process took approximately three days of pure computation time on a SPARC 10 workstation.

5.5 Overall RASTA vs. CMS Signal Processing

This experiment was performed to quantify how much telephone speech recognition performance changed when the signal processor was changed. This will be a measure of how well, comparatively, each signal processing algorithm performs in a telephone channel. It is not a measure of how good performance could be because this model development was carried out only to five mixtures and the grammar is a primitve model of language which could be improved. This will, however serve as a means to compare the two signal processors and help to decide which is the best choice for a similar task.

The triphone CMS performance result in table 9 reflects the performance of the fully-trained CMS triphone recognizer.

Table 9: Table of best recognizer performance for the CMS and RASTA signal processors.

Signal Processor	Model Type	Word Error
CMS	triphone	22%
RASTA	triphone	26%

The RASTA triphone performance result was obtained by initializing the RASTA triphone set with the RASTA monophone parameters after four iterations of training on one mixture. Those RASTA triphone models were then trained with four iterations of reestimation, after which the model parameters were tied and clustered. The number of mixtures was incremented from one to five. The trained state-clustered RASTA triphone models were then tested with independent data. The word error of the fully developed, RASTA to CMS triphones is summarized in Table 9.

In these results comparing CMS and RASTA, the word error should reflect the fidelity with which the signal processing stage captures the speech information-carrying features in telephone speech because the paradigm and experimental procedure for training and testing was the same for both signal processing algorithms.

These final results required additional training and testing of RASTA processed data. This process took approximately seven days of pure computation time on a SPARC 10 workstation.

6 Discussion and Future Experiment Recommendations

6.1 Open Questions

6.1.1 RASTA vs. CMS

The CMS and RASTA results obtained above in Table 9 show that the traditional CMS outperformed the RASTA signal processing algorithm by a small margin on this task. Both results are relatively good because the recognition strategy was primitive and the task was difficult. Two sources of difficulty in this task were human factors and intermittent noise. Throughout the corpus, speakers stop and repeat their words, make long pauses, and utter fragments. The intermittent noise was also frequent in the corpus used. Speakers made mouth noises with the phone receiver, there was background babble noise in some utterances, and in some utterances, though few, there was an audible hum or static interference that caused a low SNR.

The word error was similar for CMS compared to RASTA. It can be reasoned that the RASTA algorithm helped to alleviate the effect of the channel with by high-pass filtering approximately as well as CMS helped to alleviate the channel. The effectiveness of CMS over RASTA could be

attributed to one or both of two phenomena. It could be due to the fact that the RASTA advantage in a time-varying channel was not realized because the utterances were too short, on average, for channel characteristics to change appreciably. It could also be due to the hypothesis that the channel estimate presented in Lerner [11] need not converge completely to be effective as a channel compensation algorithm.

6.1.2 RASTA Monophones vs. CMS Monophones

The results for CMS monophones versus RASTA monophones summarized in Table 8. This result confirms the results reported by Singer and Paliwal that RASTA hurts performance for monophones. This could be due to the fact that RASTA processing is best-suited to triphone models. RASTA is a high-pass filtering which emphasizes transitions and filters out steady state information in speech. Since transitional information is associated with context changes, it makes sense that RASTA would be more helpful for triphones. Triphones are better-suited to RASTA because they depend on the sound occuring immediately before it and RASTA filters out a channel based on its recent history. The monophone models do not depend on the sound occuring just before them. Thus, monophones rely on gleaning information in the low-frequency middle state more than information near the beginning and ends of phonemes. RASTA may filter some of this steady sound out in the same way it filters out a channel. In that case, RASTA is actually filtering out speech information. This may be the phenomenon causing RASTA processing to hurt performance for monophones.

6.1.3 Potential Caveat

Although the process of recognition was the same for both signal processing algorithms, certain experimental parameters were chosen rather heuristically. Their estimates were "educated guesses" at what numerical values would work best (see caveat appendix). It could be that certain arbitrary values worked better for RASTA than for CMS. Picking those arbitrary values differently could increase or decrease performance for either CMS or RASTA, which could alter the end result. It is not clear how influential these parameters might be on performance

6.2 Recommended Future Directions

6.2.1 A Breakdown of Cepstra, PLP, RASTA, and Mean Subtraction

The RASTA and CMS performance can be further broken down experimentally. The RASTA channel-compensation was performed with PLP (see appendix) coefficients and the CMS channel compensation was performed with cepstra (see appendix). The performance of PLP implemented with a mean subtraction (PLP MS) and cepstra implemented with RASTA processing should be measured to determine if the increase/decrease in performance from CMS to RASTA results is due

primarily to the channel compensation (mean subtraction or RASTA processing) or the coefficient type (cepstra or PLP).

6.2.2 Exploration of Additive Noise Phenomena

Based on the histogram of SNR in Figure 5.1, an educated guess can be made as to which noise compensation strategies would yield gains in recognition performance. Although the global SNR level in this corpus of telephone speech was measured to be small, the temporally local SNR's in the neighborhood of corruption by additive noise are not necessarily small. This suggests that if noise compensation techniques were used to clean telephone speech, techniques that adaptively removed local noise in the neighborhood of corrupted frames would work better than techniques that removed global noise from an entire utterance.

Because certain noise phenomena are particular to telephone speech (e.g. acoustic interference with the mouthpiece of the phone), characterization of telephone noise phenomena could be explored. Although this analysis was not possible due to time restrictions, the correlation between error and labeled noise in the Macrophone Natural Numbers Corpus could be measured. If word error were correlated to a certain type of characterized noise, the type of noise causing most of the errors could be estimated. That estimation could then be used to detect that type of noise. If detected, the estimated properties of that noise could be subtracted from the signal in the temporal neighborhood of detection.

6.2.3 Exploration of More Detailed Models with RASTA

Since triphone models are more detailed models than monophone models, and large gains were observed for the development from monophone to triphone models for RASTA processing, it could be that RASTA works best with very detailed models. This could mean that although RASTA did not perform as well as CMS at the five mixture level, for more mixtures RASTA may have outperformed CMS.

7 Conclusions

CMS processing was found to slightly outperform RASTA processing for recognition of telephone speech. The word error for the best CMS and RASTA results is summarized in Table 9.

The histogram shows that more than 95% of the utterances in the Macrophone Natural Numbers corpus fell in an SNR range greater than 22dB as measured by the NIST standard. The frequency of utterances versus global SNR is plotted in Figure 5.1. This histogram shows that the Natural Numbers section of the Macrophone Corpus was shown to be relatively uncorrupted across utterances in terms of global SNR. This suggests that the channel distortion and intermittent noise are

greater sources of word error in speech recognition systems than the global additive noise. Thus, dealing with those phenomena would probably yield higher gains in recognition than attempts to subtract out global noise.

The word error for the monophone CMS and RASTA 1-mixture results is summarized in Table 8. The results obtained by Singer and Paliwal were confirmed by 70% CMS to 82% RASTA word error.

Using context-dependent triphone models over context-independent monophone models was shown to increase performance significantly. The results for CMS processing developed into 5 mixtures for both monophones and triphones is summarized in Table 7.

References

[1] J. Adam, "Technology Combats Disabilities," *IEEE Spectrum Magazine*, October, 1994, pp. 24-26.

[2] V. Digalakis, M. Weintraub, A. Sankar, H. Franco, L. Neumeyer, H. Murveit, "Continuous Speech Detection on ARPA's North American Business News Domain," *Proceedings of the ARPA Spoken Language Systems Technology Workshop,* January 1995, pp. 88-93.

[3] B. Hanson and T. Applebaum, "Subband or Cepstral Filtering for Recognition of Lombard and Channel-distorted Speech," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1993, pp. II79-80.

[4] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, April 1990, pp. 1738-1752.

[5] H. Hermansky and N. Morgan, "Towards Handling the Acoustic Environment in Spoken Language Processing," *International Conference on Spoken Language Processing*, 1992, pp. 85-88.

[6] H. Hermansky, N. Morgan and H. Hirsch, "Recognition of speech in additive and convolutional noise based on RASTA spectral processing," *Proc. of the Inter. Conf. on Acoust., Speech and Signal Proc.,* 1992, pp. II83-86.

[7] H. Hermansky and N. Morgan, "RASTA Processing of Speech," *IEEE Transactions on Speech and Audio*, October 1994.

[8] Entropic Research Laboratory, Inc., *HTK - Hidden Markov Model Toolkit v1.5,* Cambridge University Engineering Department, September 1993, pp. 18-69.

[9] C. R. Jankowski, "A Comparison of Signal Processing Stages for Automatic Speech Recognition," *Technical Report 1002*, Lincoln Laboratory, MIT, 1994, p. 3.

[10] B. H. Juang, "Speech Recognition in Adverse Environments," *IEEE Computer Speech and Language* (5), 1991, pp. 275-294.

[11] S. Lerner and B. Mazor, "Telephone Channel Normalization for Automatic Speech Recognition," *Proc. of the Inter. Conf. on Acoust., Speech and Signal Proc.,* 1992, pp. I261-264.

[12] L. R. Rabiner and R. W. Schafer, "Digital Processing of Speech Signals," pp. 3, 174, Prentice Hall, New Jersey: 1978.

[13] C. Segler, personal communication (unreported), March 8, 1995.

[14] H. Hermansky, N. Morgan, A. Bayya, and P. Cohen, "Compensation for the Effect of the Communication Channel in Auditory-like Analysis of Speech (RASTA-PLP)," *EUROSPEECH* 1991, pp. 1367-1370.

[15] Lee K-F, Hon H-W "Speaker Independent Phone Recognition Using Hidden Markov Models," *IEEE Trans ASSP*, Vol 37, No 11, 1989, pp. 1641-1648.

[16] Singer and Paliwal, "Effects of RASTA Type Processing for Speech Recognition with Speaker Rate Mismatches," to appear in *Proc. Eurospeech 1995.*

[17] K. Taussig and J. Bernstein, "Macrophone: An American English Telephone Speech Corpus," *Proc. ARPAP Human Language Technology Workshop,* March 1994, pp. 27-30.

A Hidden Markov Models

A.1 Model Topology

The speech recognition system described assumes that the speech signal is a sequence of one or more symbols. The symbols are encoded acoustically and it is the job of the recognizer to decode them. The speech waveform is first converted to a sequence of equally spaced (10ms) feature vectors. The parameterizations used in this project are CMS and RASTA. The role of the recognizer is to effect a mapping between sequences of speech feature vectors and the underlying symbol sequences.

In this experimentation, a Hidden Markov Model (HMM) is used to model speech. The probability of a transition from state to state in a HMM depends on the observation (the current feature vector sample), the transition probability and the conditional probability that the observation is classified as the next state [8].

The term model topology refers to the way the feature vector-parameterized data is fed into the pattern recognition stage. It summarizes the type of HMM used in pattern recognition. In this experimentation, a 5-mixture, left-to-right, no skips, 3 state Hidden Markov Model Topology was used for each feature vector parameterization. In general, more complex model topologies are slower and more accurate [8].

There was assumed to be little correlation between adjacent parameters in the feature vector representations of phonemes, which implies that all covariance elements (off-diagonal elements) would be near zero. Thus, a diagonal covariance matrix is sufficient to represent the feature vector parameterizations of phoneme variances [8].

A.2 Scoring HMM States and Scoring Word Sequences

Once a state is parameterized in terms of means, covariances, and weights, the observation can be compared to the description of that state. In this way, the acoustic feature vector observation is scored. The closer a feature vector parameterization of a waveform is to a parameterization of a phoneme, the more likely it is to be that phoneme. When scoring word sequences, a concatenation of states (influenced also by a grammar and dictionary, if used), the probability that a sequence of words occured is the product of the probabilities of each word given the observation [8].

A.3 Possible Parametric Caveats

One experimental parameter, the "minimum within cluster distance" was taken directly from a suggestion in the software manual. It was chosen to be 0.5. That parameter's effects on the number of triphone clusters could be experimentally tuned. It could be that this parameter choice worked better for RASTA than it did for CMS, creating 505 clusters in the RASTA space, and 415

clusters in the CMS space [8].

Another potential parametric caveat is that the minimum variance feature vectors chosen for CMS and RASTA caused RASTA to outperform CMS. The procedure was to examine a subset of model variances in the feature vector space. A variance that would be less than approximately 90% of the feature vector variances observed was chosen. This estimate of minimum variance was divided by ten to create the minimum variance feature vector. It could be that the RASTA minimum variance feature vector worked better for RASTA processing than the CMS minimum variance feature vector worked for CMS [8].

A.4 Hidden Markov Model Initialization

This project used the Hidden Markov Model Toolkit (HTK), a commercially available pattern recognizer (see stage two of Figure 1), to build and evaluate recognition systems. Each waveform and transcription in the training and evaluation set of the natural numbers section of the Macrophone Corpus was first converted to the HTK software format [8] using the coding steps listed above. The HTK bootstrap initialization model topology used was a left-to-right, no skips, 5-state (3-state model with 2 non-emitting nodes), single stream, diagonal covariance, 26 feature vector length model. The initialization model has the following form for the zero-mean HTK cepstra experimentation:

```
``model name''
<BeginHMM>
<NumStates> 5 <StreamInfo> 1 26 <VecSize> 26
<DIAGC> <NULLD> <MFCC_E_D_Z>
<State> 2
<Mean> 26
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 26
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3
<Mean> 26
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 26
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0
```

```
<State> 4
<Mean> 26
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 26
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0
<TransP> 5
0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 7.038784e-01 2.961216e-01 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 9.679869e-01 3.201302e-02 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00 8.053547e-01 1.946454e-01
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<EndHMM>
.
```

A.5 Minimum Variance Vectors

For CMS, the minimum variance vector was chosen to be:

```
~v "varFloor1"
<Variance> 26
1.000000e-02 1.000000e-02 1.000000e-03 1.000000e-03 1.000000e-03
1.000000e-04 1.000000e-04 1.000000e-04 5.000000e-05 5.000000e-06
5.000000e-07 5.000000e-07 5.000000e-07 1.000000e-02 1.000000e-02
1.000000e-03 1.000000e-03 1.000000e-03 1.000000e-04 1.000000e-04
1.000000e-04 5.000000e-05 5.000000e-06 5.000000e-07 5.000000e-07
5.000000e-07
```

For RASTA, the minimum variance vector was chosen to be:

```
~v "varFloor1"
<Variance> 26
1.000000e-04 1.000000e-04 1.000000e-05 1.000000e-05 1.000000e-05
1.000000e-05 1.000000e-06 1.000000e-06 5.000000e-07 5.000000e-07
5.000000e-07 5.000000e-07 5.000000e-07 1.000000e-04 1.000000e-02
1.000000e-03 1.000000e-03 1.000000e-03 1.000000e-04 1.000000e-04
1.000000e-04 5.000000e-05 5.000000e-06 5.000000e-07 5.000000e-07
5.000000e-08
```

B Commands to Parameterize Acoustic Data into HTK Format

B.1 CMS, and Coding Speech with HTK tool, HCode

The following steps converted the binary speech acoustic waveform into the HTK sequence of CMS feature vectors:

```
btosps -f 8000 -n 1 -c "Converted from headerless" binarywaveform espsfile
HCode -F ESPS -m -j -n 12 -p 12 -e -d -w 20 espsfile HTKCMSvectors
```

B.2 RASTA PLP, and Coding Speech with mrasta code

The following steps converted the binary speech acoustic waveform into the HTK sequence of RASTA PLP feature vectors:

```
mrasta -L -n 13 -i binarywaveform -o RASTAPLPbinaryvectors
setenv HCOERCE USER_D
btosps -n13 -tfloat -f100 -cRASTAPLP RASTAPLPvectors.bin espsvectors
HCopy -F ESPS espsvectors HTKRASTAPLPvectorsequence
```

C HTK tool formulae

C.1 Baum-Welch Reestimation Formulae

This section is a summary of the math involved in model training using the Embedded Reestimation algorithm [8].

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{P} \tag{12}$$

where α_j = joint probability of observing T speech feature vectors and being in state j at time t.

where β_j = conditional probability of observing T speech feature vectors and being in state j at time t.

Estimated mean for state j is:

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \tag{13}$$

Estimated covariance matrix for state j is:

$$\hat{\sigma_j} = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)}{\sum_{t=1}^{T} L_j(t)} \tag{14}$$

[8].

C.2 Viterbi Algorithm

This section is a summary of the math involved in recognition using the Viterbi algorithm.

$$\psi_j(t) = max_i \psi_i(t-1) + \log(a_{ij}) + \log(b_j(o_t)) \tag{15}$$

where ψ is the partial path of the state sequence [8].

D NIST Speech Detector

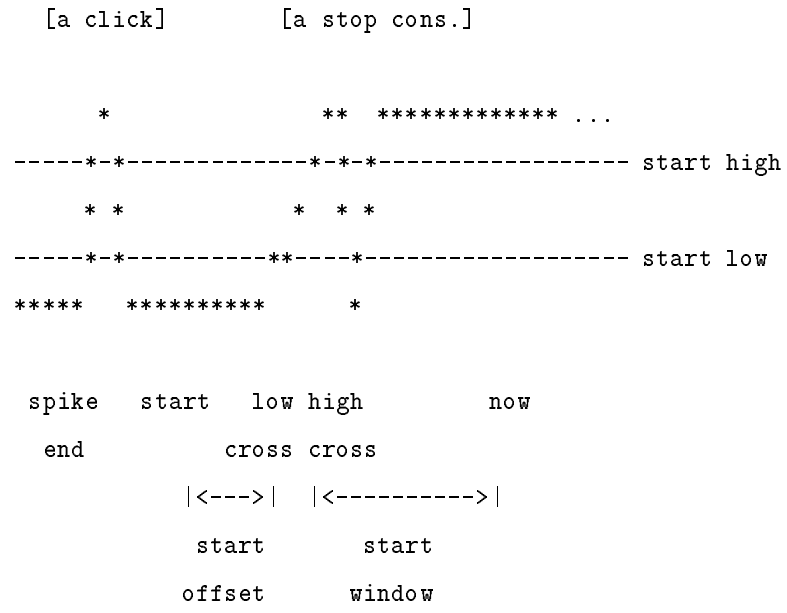
The NIST SPeech Quality Assurance (SPQA) Package

Version 2.3

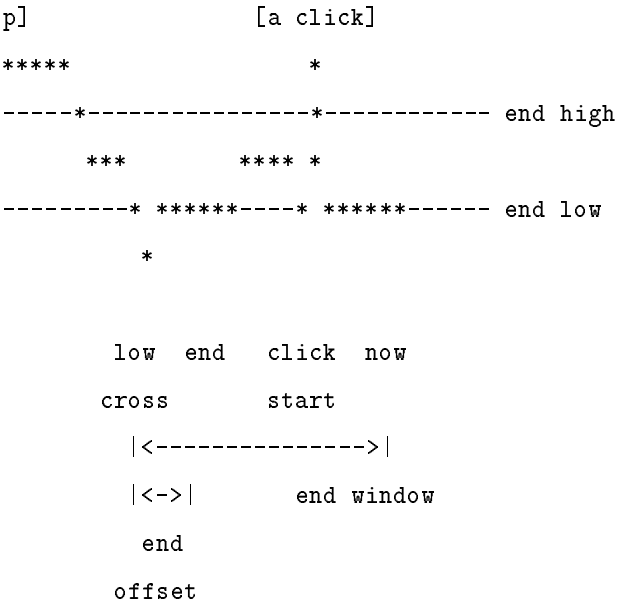
(taken from speech det.c – Speech Detection algorithm)

The speech detect is a fairly simple two event windowing algorithm, based on the value (usually in db) of the average energy (r0).

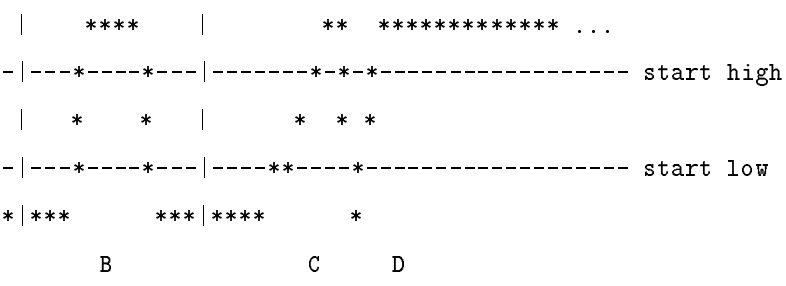
For start of speech, if r0 is greater than start high water mark for more than start window number of frames, then the beginning of speech was at start offset number of frames before when speech went above start low water mark.



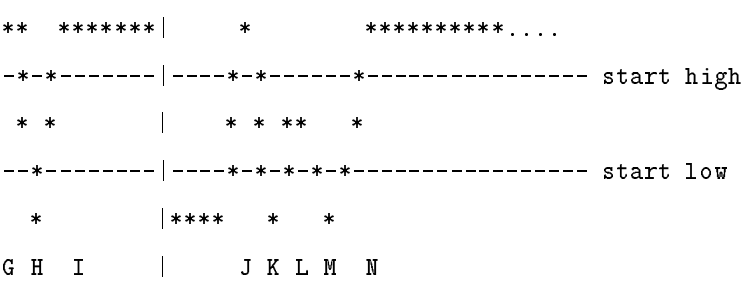
The end of speech is not quite symetric: If r0 goes under end low water mark, and then stays below end high water mark for end window number of frames, then end of speech is end offset frames after when speech went below end low water mark.



All this is complicated somewhat by the need to reject brief events (spikes and dropouts) in different ways depending on how close they are to other speech events. For instance:
(when starting)



A is short, and far away from other speech, so it is a spike and should be ignored. B is long so even though it is far from other speech, it is not a spike. C is short, but close to D, so it is part of the speech.
(when starting)



E is short and below start high, the dropout at F is shorter than max dropout, G is short but above short high, dropout at H is also shorter than max dropout, and from the beginning of G to into I is longer than start window. In this case, speech starts at the beginning of E. JKLMN is like EFGHI, but J is above start high, and L is not. Here also speech starts at the beginning of J.
(when stopping)

```
** ***    *****   *            *
-**------**------*-*-*-----------*------------ end high
        *      * * *       **** *
--------*---------**-* **********-* ********** end low
        *       *   *
        O       P Q             R
```

O is short, so it is part of the speech. P is a short dropout, followed by Q, a short spike. This is also part of speech. (Alternately Q is a short spike, but "close" to the speech, so it is part of the speech.) R is a spike far from speeh (on both sides), so it is not speech and should not be counted in the long end-of-speech timeout period.
(when stopping)

```
** ***            *    *****
-**------**---------*-*--*--------- end high
        *        * * *
--------*--------*-* *---------- end low
         ********   *
                  S   T
```

Also, S is a short spike, far from past speech, but it is followed closely by a second high point at T, so it is part of the speeech. This is like an end of speech event followed by a new beginning of speech event.
(when stopping)

```
** ***            **   **
-**------**---------*--*--* *-------- end high
        *        *  * *   *
--------*---------*--* *---*-------- end low
         ********    *     ****
                   U    V
```

Worse yet, neither U nor V are large enough to be counted as longer than a pulse spike, but together they are (beg of P to end of Q is long enough). They are speech (a short, unstressed word with a stop in the middle).

E Phoneme Lists

The list of monophones used was:

```
iy ih eh ae ix ax ah uw uh ao aa ey ay oy aw ow l r y w er m n nx ch jh dh
b  d dx g p t k z zh v f th s sh hh axr ei oh e h sil @noise
```

The list of triphones used was generated using the above monophone list and the pronunciation network using the HTK tool, HDEd:

```
HDEd -n triphonelist tri.ded monophoneprondict triphoneprondict
```

where triphonelist is the output list of triphones, the tri.ded option specifies a file containing only the letters TC (for triphone conversion), the monophoneprondict is the monophone input pronunciation network and the output triphone network is triphoneprondict. Before this can be done, the monophone dictionary must be edited to eliminate all $, % and ;.

F Examples of Grammar and Pronunciation Network

The network definition used for recognition with the HTK tool consisted of a word-pair language model and pronunciation network. The pronunciation network describes how the phoneme models are concatenated temporally to form words using the "glue" nodes in each phoneme HMM, 1 and 5. The pronunciation network ends at the pronunciation of the word, zero. The word pair language model begins after the definition of the pronunciation of zero. The network is too long to include as a part of this report, but the format used is illustrated with the following network definition used in the triphone experimentation. Vertical ellipsis are used to represent a repetition trend of definition type.

```
$a              = WD_BEGIN%a (( ax%% [sp%%] )) WD_END%a;
$a's            = WD_BEGIN%a's (( ey+z%% ey-z%% [sp%%] )) WD_END%a's;
$acre           = WD_BEGIN%acre (( ey+k%% ey-k+axr%% k-axr%% [sp%%] )) WD_END%acre;
.
.
.
$z              = WD_BEGIN%z (( z+iy%% z-iy%% [sp%%] )) WD_END%z;
$zero           = WD_BEGIN%zero((z+iy%% z-iy+r%% r-ow%% [sp%%]))WD_END%zero;

$TLOOP_BEGIN_FLLWRS = a| about| approximately . . . west| what| yes| zero;

$TLOOP_END_PREDS = a| acre| acres| again| alabama| ave| avenue| b|
```

```
barrel . . . yes| york| zero;

$a_FLLWRS = avenue| big| bout| broad| c| condominium| half| house|
hun| hundred| million| mobil| north| number| quarter| r| route|
rural| single| telephone| th| thousand| west| TLOOP_END;

$acre_FLLWRS = TLOOP_END;
.
.
.
$you're_FLLWRS = asking;

$zero_FLLWRS = blue| cobey| cumberland| decimal| doll| dot| eight|
fifteen| five| four| hilltop| lira|  nine| nineteen|
oak| one| point| sarah| seven| six| south| three| tons| two|
zero| TLOOP_END;
.
.
.
$wrd = TLOOP_BEGIN+TLOOP_BEGIN_FLLWRS |
TLOOP_END_PREDS-TLOOP_END |
a+a_FLLWRS |
acre+acre_FLLWRS |
.
.
.
you+you_FLLWRS |
zero+zero_FLLWRS;
( sil%% <<$wrd>> sil%% )
```

G Telephone Device for the Deaf (TDD)

A typical TDD costs between $230 and $600 depending on the range of application necessary. Currently, the deaf community can not receive federal funding for the purpose of purchasing a TDD. Conversations on TDD's typically last 25% longer than voice telephone communication. Relay stations are set up to facilitate hearing-deaf telephone communication. A hearing party can

dial a relay station interpreter and the interpreter dials the deaf party. The relay station interpreter then types into the TDD what is said by the hearing party and reads what is written by the deaf party. Besides the obvious impersonal nature of this type of communication, relay stations have set hours, and each relay station interpreter can only interpret one phone conversation at a time [13].

H Cepstra and PLP

H.1 Standard Cepstra (MFCC)

Cepstra is a means to parameterize speech data into feature vectors via four mathematical transformations of the sampled waveform in time. The method of cepstral computation used in this project is: 1) Take the magnitude of a 10 ms window of the Fourier Transform, 2) take the log of the magnitude spectrum (logspectrum), 3) warp the frequency axis according to a standard Mel-Scale, and 4) take the Inverse Fourier Transform of that logspectrum. Mathematically,

$$C_s = \text{IFT}(\log(frequency - warpedspectrum|S(\omega)|)) \tag{16}$$

(where IFT is the Inverse Fourier Transform and $|x|$ denotes the magnitude of x) where $C_s[n]$ is the cepstral feature vector representation of the windowed $s[n]$, whose spectrum is $|S(\omega)|$.

H.2 Perceptual Linear Prediction (PLP)

A standard method of modeling some of the characteristics of human auditory system is to use perceptual linear prediction (PLP) coefficients. In PLP, the frequency axis is warped. The warping reflects the Bark Scale rather than the Mel Scale. The magnitude spectrum is made to behave according to the intensity-loudness power law. This simulates the nonlinear relation between the intensity of phoneme and its perceived loudness. Also, in PLP the magnitude spectrum is filtered to simulate the sensitivity of hearing at the 40dB level. This filter slightly amplifies the high-frequency components of speech, or "pre-emphasizes" them. The resulting "auditory-like" spectrum is smoothed using linear prediction. The underlying principle of PLP analysis is to approximate the auditory spectrum of speech [4].