Chapter 1 Introduction

1.1. Historical Background

Automatic speech recognition is the mapping from speech to underlying text. Research in speech recognition has been going on for more than 30 years. Speech recognition is a field that combines acoustic-phonetic modeling, signal processing, pattern recognition, and artificial intelligence techniques to achieve its goal. In this thesis, we primarily address the acoustic-phonetic modeling and pattern recognition aspects of the task. For continuous speech recognition, such as this, the acoustic unit we model is the phoneme.

Of the techniques used in speech recognition, pattern classification approaches have given the best results [2]. There have been three major pattern classification algorithms used in speech recognition, Dynamic Time Warping (DTW), Hidden Markov Models (HMM), and Artificial Neural Networks (ANN).

The first of these is Dynamic Time Warping (DTW) [3-5]. In this approach, speech is represented with templates. Reference patterns are obtained from collected speech data. The input speech is mapped to the reference patterns using a non-linear time alignment. The algorithm proceeds by searching the space to map from the time sequence of the input speech to that of the reference patterns so that the total distance is minimized. The lowest distance reference pattern label thus constitutes the recognized text. This technique is no longer common in speech recognition due to the introduction of a more successful statistical model, the HMM.

The second approach is HMMs. It is currently the most successful model in speech recognition [6]. HMMs were first introduced in the late 80's and were successful in modeling several important applications [1]. This method uses a soft decision mechanism to assign the data to the possible classes. Likelihoods are computed for each of the possible alternatives, and the most likely one is selected. More details are given about HMMs in Chapters 3.

The third model is Artificial Neural Networks (ANNs). It is inspired by the properties of biological neurons. ANNs are composed of a large number of interconnected elements analogous to neurons. These elements are tied together with weights. We learn which weights to use by presenting input/output training data to the networks. Training algorithms like back propagation are used to iteratively adjust the weights. Back Propagation is a gradient descent approach. The learned weights store the mapping that is used to classify new data. ANNs are a useful technique, especially for isolated word classification [7] [8]. For continuous recognition, they can be combined with HMMs to form hybrid classifiers [9]. Some of the limitations of artificial neural network are the training time and training data it requires.

1.2. Motivations

1.2.1. The Co-articulation Effect and Other Sources of Variability in Speech

Most of the difficulties related to the problem of automatic speech recognition come from the diverse variabilities involved with speech communication. In order to achieve good performance, any realistic speech recognizer must model most of these variables.

The first two issues that any continuous speech recognition system must face are the variability of the vocal tract configuration across phonemes and phoneme classes, and the fact that the statistical characteristics of speech vary every 20-25ms.

Phoneme pronunciation duration generally varies according to word duration. Phonemes are pronounced more quickly in longer words than in shorter words to adjust the speaking rate [1]. Also, some phonemes tend to be more stressed than others depending on their importance in understanding the meaning of the words. There are speaker dependent variabilities such as differences in speaking rate and differences in style. Speaker dependent variabilities also include those related to gender, age, and the dialect. Utterances recorded from younger speakers are different from those recorded from older speakers. Another source of variability is that introduced by background noise. Various phoneme models have been developed to address some of those variabilities.

In natural speech, there are no marked boundaries between acoustic data segments. Word and phoneme boundaries are non-existent. Even with expert labeling of the acoustic data, it is very difficult to establish a hard boundary between the phonemes and words that form an utterance.

It takes a fraction of a second to produce each phoneme. Speech is a stream of phonemes that sounds very smooth. This smoothness results from the coordination of the articulators' movements by the brain. The movements of these articulators - lips, tongue, jaw, velum, and larynx- are coordinated so that movements needed for adjacent phonemes are simultaneous and overlapping. This is known as co-articulation.

1.2.2. Modeling Co-articulation, Local and Global Variabilities in Acoustic

HMMs were initially used to model acoustic observations associated with individual phonemes without consideration of context (monophones). However, studies have shown that phonemes are highly influenced by co-articulation, as described above, and monophone models do not represent the context in which the phoneme is issued. To capture the co-articulation effect caused by the previous and next phoneme to a particular phoneme in a word or sentence, we use a context dependent phoneme model called triphone, which will be described in more detail in Chapter 4. Triphones were successfully introduced and remain the phoneme level model used in many current speech recognizers. Other models that have been proposed to account for the co-articulation effect and other sources of variabilities in phonemes include:

- Addition of feature derivatives [32].
- Trended HMM [34] [36].
- Stochastic trajectory models [40-42].

1.2.3. Issues With the Triphone Models

Triphone parameters are estimated from training data. One drawback of training is that it is not practical for all possible triphones to occur in a training speech corpus. There are triphones that have a very low frequency of occurrence, which are likely to be absent in any such corpus. Even if they do occur, this may be in such small amounts that parameter estimation for them will give very poor models, which will in turn decrease the performance of the recognizer. Added to this is the crucial need for well-balanced data in order to include variability of speakers, and to incorporate a broad coverage of co-articulation in the data. A large training data corpus consists of thousands of sentences,

and hours of recording. This results in costly material, transcription, and very long training time, often not possible.

1.3. Goal of the Research

As discussed in this chapter, triphones were introduced to solve problems related to co-articulation. Triphones are typically estimated through training, but the success of this training process is conditioned by requirements such as the need for a large amount of well-balanced training data.

In our work, we address these issues by learning ruled-based trajectory interpolations to build triphones directly from monophones. The goal is to find a means to generate models for all possible triphones while reducing training data cost, transcription cost and training time. A secondary goal of this study is to investigate the consistency of the way co-articulation affects the acoustic speech unit for real data, thereby making a bridge between the known theory of co-articulation and what is observed from real data.

1.4. Outline of Thesis

The remainder of this document is structured as follows. In Chapter 2 we describe the most common speech production model and the acoustic processing of speech. Chapter 3 covers the Hidden Markov Model. In this chapter, the general concept of Markov chain is first discussed, then the Hidden Markov Model for speech recognition is presented, including a summary of the training process, and the recognition algorithm. Chapter 4 is a brief overview of models that have been proposed to solve problems related to co-articulation and other sources of variability related to natural speech.

Models such as triphones, trended-HMMs, and stochastic trajectory models are presented and discussed. We end this section by emphasizing problems related to triphone estimation and decision tree-based clustering. Chapter 5 contains the outline of our new design method and details on its implementation. Experiments, results, and comparisons of the new design with existing methods are found in Chapter 6. The last chapter is devoted to a discussion of the various results obtained and prospective directions for future work.

Chapter 2 Speech Production Model and Acoustic Processing of Speech

The task of feature extraction for speech is one of finding a representation of speech that will yield good classification. It is a key element of the recognition process. Differences between phonemes are caused by differences in their production mechanism. Therefore, understanding the properties of human speech production processes and the human listening process is crucial for good feature extraction. This process exists across the linguistic, physiological, and acoustical levels, and knowledge about psychoacoustics, human anatomy, and physiology has contributed to the improvement of speech recognition performance through the years.

2.1. Speech Production Model

Each sound that is produced corresponds to a specific configuration of the vocal tract. A high level diagram of this chain of events, called the speech chain, [15] is provided in Figure 1. Speech is used to communicate information from a speaker to a listener. This process starts with a thought that the speaker wants to communicate to the listener [16]. The idea is arranged into a linguistic form by appropriately selecting words and phrases. These words are ordered according to the grammatical structure of the language. The brain sends, in the form of impulses along motor nerves, the required commands to the muscles that control the articulators. This produces the pressure changes in the surrounding air that propagates into the environment to produce the intended sound [17].

The detail of the process that takes place at the physiological level is pictured in Figure 2.

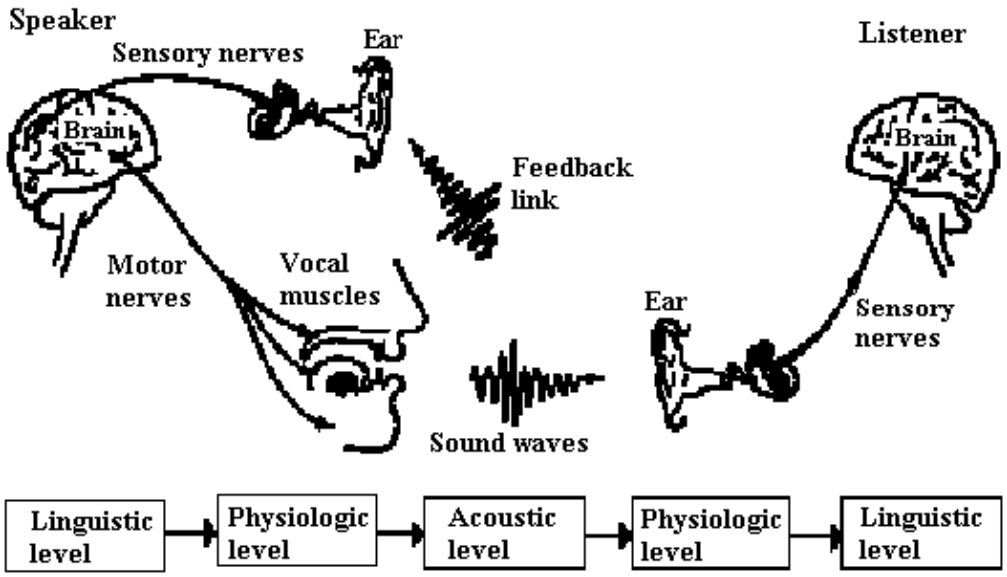

Figure 1: The speech chain (from Denes & Pinson 1992). Used by permission.

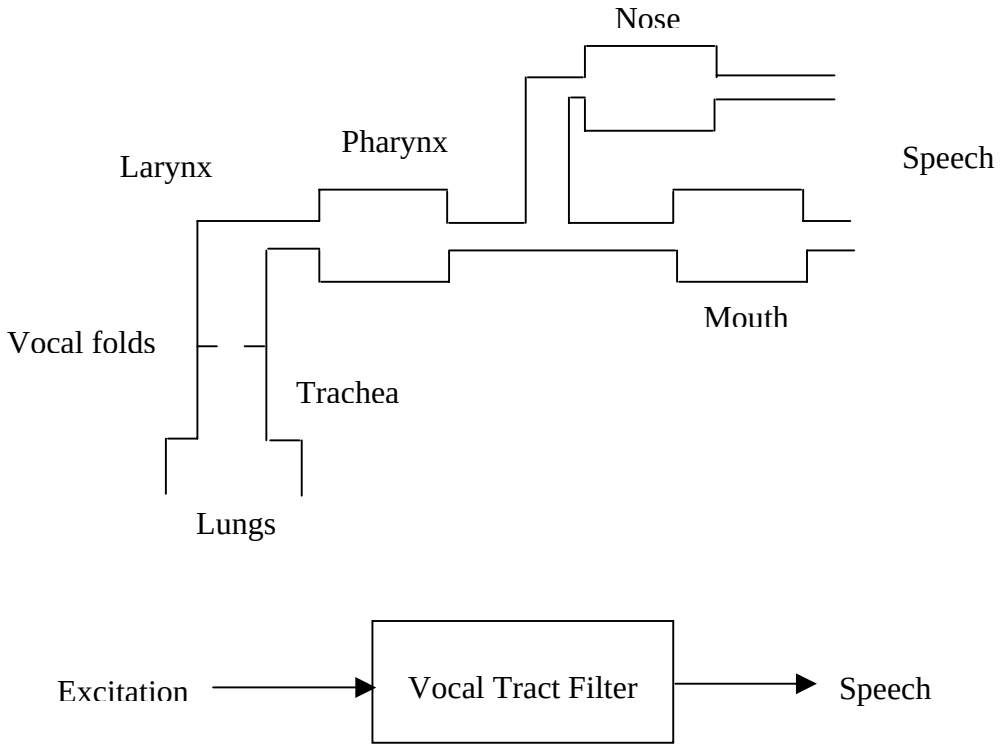

Figure 2: Block diagram of human speech production

The vocal tract is the path through the articulators that produces speech. The simplest model for speech production, derived from the above analysis of our actual human production process, models the vocal tract as an all-pole filter with transfer function

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-i}}, \tag{2.1}$$

where p is the number of poles and $a_0 = 1$. The a_i's are the filter coefficients called Linear Prediction Coefficients (LPCs). They are chosen to minimize the prediction error between the actual signal and the predicted signal.

Under this simplification, speech is represented as the response of this filter due to an excitation created by the vibration of the vocal folds. The filter coefficients extracted during this analysis are called LPC's.

2.2. Production of Sounds

In the English language, sounds can be categorized based on the type of excitation, the manner of excitation, and the place of excitation. What makes substantial differences between sounds in the English language is the shape of the vocal tract. Depending on the manner and the place of articulation, the basic phoneme can be classified into classes. Classifications consist of Vowels, Fricatives, Nasals, Affricates, Stops, Semi-vowels, and Diphthongs.

2.2.1. Type of Excitation

There are two types of excitations: voiced and unvoiced. Voiced excitation is quasi-periodic and characterized by its fundamental frequency. The measured fundamental frequency is different from the perceived fundamental frequency known as

pitch. Conversely, unvoiced excitation exhibits no periodicity and is often modeled by white noise.

2.2.2. Manner of Articulation

The manner of articulation is determined by the path of the excitation signal as well as the diverse constrictions at various location of the vocal tract that modify the excitation signal. These differences in the vocal tract configuration result in different speech sounds or groups of sounds. For vowels, the periodic excitation passes through an unrestricted vocal tract (e.g., /a/, /u/). For fricatives, the excitation is random and passes through a constriction (e.g., /s/, /f/). Plosives are made by a brusque release of a restricted airflow due to an increased air pressure (e.g., /b/, /p/). Nasals result from an excitation passing through the nasal cavity (e.g., /m/, /n/).

2.2.3. Place of Articulation

The manner of articulation and the type of excitation divide English language into general groups of phonemes. The place of articulation allows finer differentiation between phonemes of the same group. For vowels, we can distinguish front vowels, central vowels and back vowels, depending on whether the constriction occurs in the front, central or back part of the oral tract. For example, for the vowel in the word “beet”, the tongue touches the roof of the mouth just behind the teeth whereas for the vowel in the word “boot”, the constriction is produced by the back of the tongue near the velum. There is also additional information about the place of articulation which helps us establish the discrimination of individual sounds. Table 1 contains the classification of non-vowel speech according to the place and manner of articulation.

Table 1: Classification of nonvowel sounds according to the place and manner of articulation

Place of articulation / Manner of articulation	Bilabial	Labiodental	Interdental	Alveolar	Palatal	Velar	Glottal
Nasal stop	/m/			/n/		/G/	
Oral stop	/p/, /b/			/t/, /d/		/k/, /g/	/Q/
Fricative		/f/, /v/	/T/, /D/	/s/, /z/			/h/
Glide	/w/				/y/		
Liquids				/l/, /r/			

2.2.4. Vowels

Vowels are voiced speech sounds produced with a constant vocal-tract shape. They vary a lot in duration, ranging from 40-400msec [16]. The position of the tongue helps to further classify vowel sounds in sub-categories. In American English, depending on the place of the constriction, we can classify vowels into three categories: front vowels, central vowels and back vowels. The front vowels are: /i/ (as in “heed”), /I/ (as in “hid”), /e/ (“hayed”), /E/ (“head”), /@/ (“had”). The central vowels category is composed of: /R/ (“heard”), /x/ (“ago”), /A/ (“mud”). Finally, the back vowels consist of: /u/ (“who’d”), /U/ (“hood”), /o/ (“hoed”), /c/ (“hawed”), /a/ (“hod”).

2.2.5. Diphthongs

Diphthongs are voiced dynamic phonemes containing two vowel sounds. They represent the smooth transition of articulators from the position required to produce the first vowel to that required for the next vowel. We can easily determine if a sound is vowel or diphthong by producing the sound. If the vocal tract does not maintain a constant shape and the two target sounds of the vocal tract are vowel then the sound is a diphthong. They are four diphthongs in American English, which are: /Y/ (“pie”), /W/ (“out”), /O/ (“toy”), /yu/ (“you”).

2.2.6. Semivowels

Semivowels are composed of two groups of sounds: glides and liquids. Glides are sounds that are associated with one target position. They consist of transitions toward and then away from a target, maintaining the target’s position for less time than vowels. In American English we find the following glides: /r/ (“ran”), /y/ (“yam”). Liquids have similar spectral characteristics to those of vowel sounds but result from a more constricted vocal tract. The liquids found in American English are: /w/ (“wet”), /l/ (“lawn”)

2.2.7. Fricatives

Fricatives are consonant sounds where the vocal tract is excited by a turbulent airflow, produced when the airflow passes a constriction in the vocal tract. In American English, we distinguish both voiced and unvoiced fricatives. The voiced fricatives consist of /v/ (“vine”), /D/ (“then”), /z/ (“zebra”), /Z/ (“measure”), and the unvoiced fricatives are /f/ (“fine”), /T/ (“thing”), /s/ (“cease”), /S/ (“mesh”).

2.2.8. Stops

Stop consonants, also called plosives, are produced in two phases. In the first phase, air pressure is built up behind a complete constriction at some point in the vocal tract. In the second phase, there is a sudden release of this air which produce the plosive sound. Stops are transient sounds, generally short in duration. In American English, we distinguish voiced plosives, which are /b/ (“be”), /d/ (“day”), /g/ (“go”) from unvoiced plosives, which are /p/ (“pea”), /t/ (“tea”), /k/ (“key”).

2.2.9. Affricates

Affricates are dynamic consonant sounds that result from the combination of two sounds: the transition from a plosive to a fricative. Two affricates are found in American English: the unvoiced affricate /C/ (“change”) and the voiced affricate /J/ (“jam”). The voiced affricate /j/ is formed by the transition from the voiced stop /d/ to the voiced fricative /Z/. The unvoiced affricate /C/ results from the production of the unvoiced stop /t/ followed by the voiced fricative /S/.

2.2.10. Nasals

Nasals are voiced consonant sounds produced as the airflow passes through an open nasal cavity while the oral cavity is closed. Nasals are lower in energy than most vowels due to the closure of the oral cavity and the limited ability of the nasal cavity to radiate sound. American English includes three nasals: /m/ (“more”), /n/ (“noon”), /G/ (“sing”).

2.3. Acoustic Processing of Speech

Our goal is to extract features that are important with respect to the goal of minimizing recognition error. For this goal, the shape of the spectral envelope should be

captured by the feature set. The excitation signal is speaker dependent and therefore not relevant to this task, in the case of a stress language like English. Several techniques can be used to extract vocal tract coefficients from the speech signal. These include filter bank analysis, Linear Predictive Coding (LPC), and cepstral analysis. The complex cepstrum of a signal $s(n)$ is:

$$FT^{-1}\{\log(FT\{s(n)\})\}. \tag{2.2}$$

The most commonly used feature is the Mel frequency cepstral coefficient (MFCC). MFCCs are extensions of the cepstrals which are used to better represent human auditory models. There are several possible methods to extract MFCCs. We will follow the description given by Figure 3. It is the method that results in good performance [18].

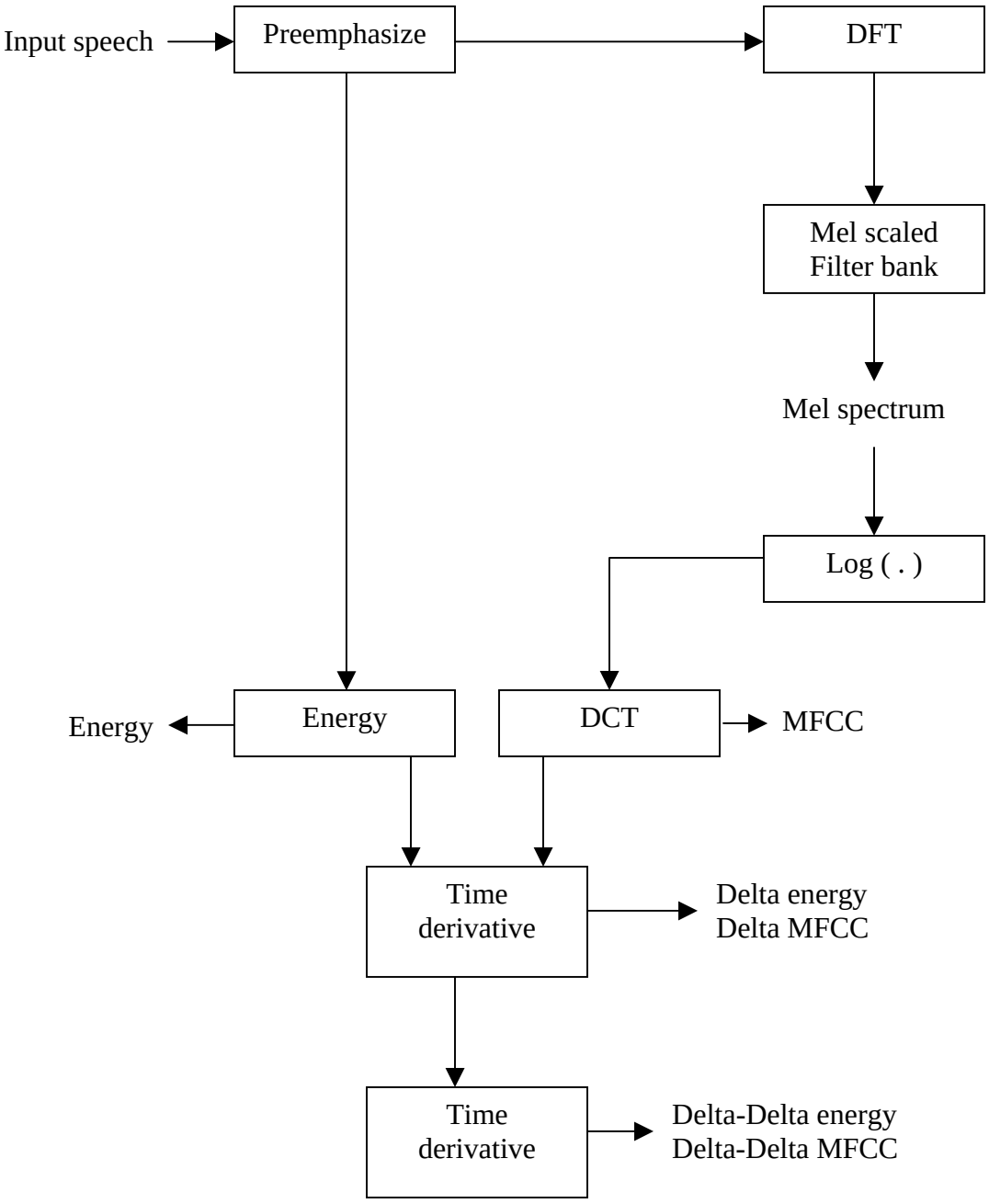

Figure 3: Block diagram of the MFCC feature extraction

2.3.1. MFCC Feature Analysis

The first step of feature extraction is to preemphasize the speech signal in order to compensate for the spectral tilt of the higher frequencies with respect to the lower

frequencies. This is done by filtering the speech signal with the first-order FIR filter $H(z) = 1 - kz^{-1}$ ($0 < k < 1$, generally we use $k = 0.97$). The signal is then divided into quasi-stationary segments using windowing (the characteristics of the speech signal are assumed to be invariant during the time interval of the window). A Hamming window is generally used for this purpose. A short-time discrete Fourier transform (short-time DFT) is performed on each windowed signal. The impulse response of the Hamming window is given by

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad n = 0, ..., N-1. \tag{2.3}$$

The short-time DFT of each window is then processed by Mel filter banks. Filter banks are banks of bandpass filters, usually non-uniformly spaced along the frequency axis used to measure the energy in various frequency bands. Frequencies below 1 kHz are processed with more filter-banks than frequencies above 1 kHz. For the Mel scale, which we use in speech recognition to correspond to the human perceptual scale, the filter-bank is uniformly scaled for frequencies below 1 kHz and logarithmically scaled for frequencies above 1 kHz as illustrated in Figure 4. An inverse discrete Fourier transform (IDFT) of the logarithm square of the filter banks output gives the MFCCs.

2.3.2. Energy Computation

Differences in energy among phonemes show that energy is a good feature to distinguish between phoneme sounds. The normalized log of the raw signal energy is usually used as the energy coefficient. The energy is computed as the logarithm of the signal energy. For a speech segment of length N, the energy is:

$$E = \log \sum_{n=1}^{N} s_n^2 \tag{2.4}$$

The normalized log energy is obtained by subtracting from E the maximum value $E_{\max}$ of energy in the utterance.

$$\overline{E} = E - E_{\max} \tag{2.5}$$

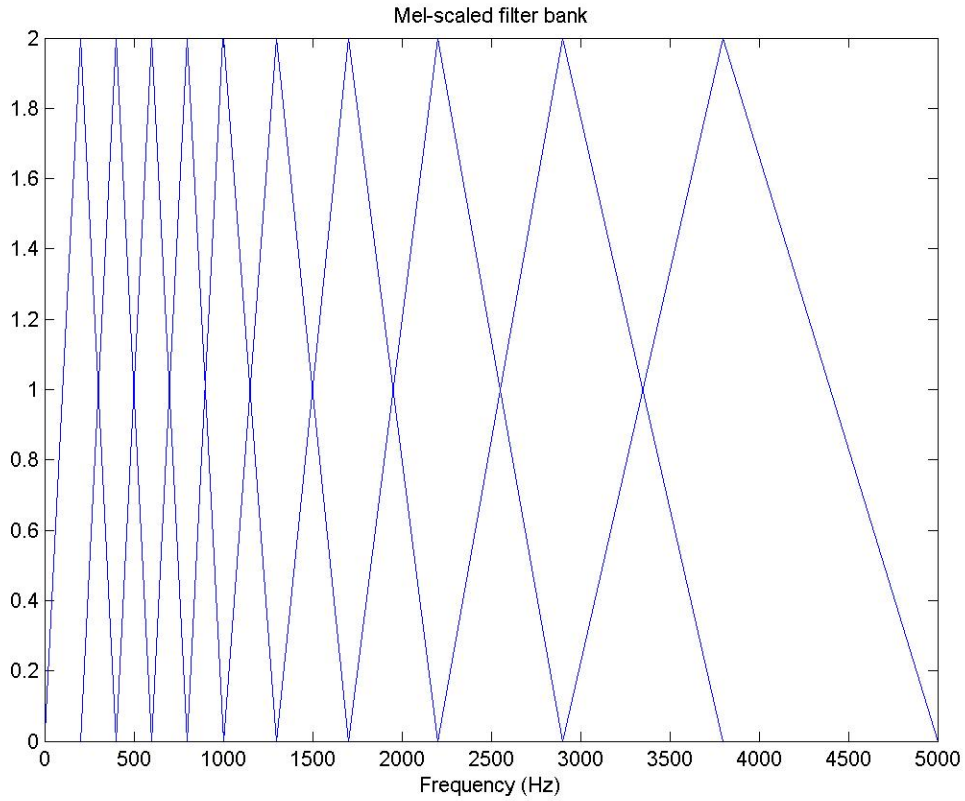

Figure 4: Mel-scaled filter bank

2.3.3. Delta and Delta-Delta Coefficients Computation

The feature vector included so far does not have any information about the time evolution of the spectrum. This information is often included in the feature set by the means of cepstral and energy derivatives. The first order derivatives of these coefficients are referred to as delta coefficients and the second order derivatives as delta-delta coefficients. The delta coefficients are computed using linear regression:

$$\Delta x(m) = \frac{\sum_{i=1}^{k} (i)[x(m+i) - x(m-i)]}{2\sum_{i=1}^{k} i^2}, \tag{2.6}$$

where $2k+1$ is the size of the regression window and x denotes the cepstrum.

The delta-delta coefficients, the second order derivatives are computed using the same linear regression applied to a window of delta coefficients.

Chapter 3 Hidden Markov Models

In this chapter, we describe the Hidden Markov Model (HMM) and its applications to speech recognition. We will first give a description of the general Markov process. By definition, a random process whose past has no influence on the future if its present is known is called a Markov process [19]. This concept can be applied to a real world process where the underlying random variable is either continuous or discrete. A continuous time process is said to be a Markov process if its probability density function (pdf) is completely determined by the value of the process for any infinitely small period of preceding time. Similarly, a discrete time process whose pdf is completely determined by the value of the process at the previous time is said to be a discrete Markov process, as shown in Equation (3.1). We generally denote a discrete time Markov process as $X(t)$

$$P(X_{t+1} \mid X_t, ..., X_1) = P(X_{t+1} \mid X_t). \tag{3.1}$$

This definition can be extended to a k-order Markov process. For a k-order Markov process, the value of the process at any given time is determined by its k-past values [19], i.e.

$$P(X_{t+1} \mid X_t, ..., X_1) = P(X_{t+1} \mid X_t, ...X_{t-k+1}). \tag{3.2}$$

The general Markov process is then a particular case of the k-order Markov process with $k = 1$.

The state of a process at any given time is defined as the value of the process. We can distinguish several types of Markov processes:

- Discrete-time, discrete-state
- Discrete-time, continuous state
- Continuous-time, discrete-state

- Continuous-time, continuous-state.

In the following section, we introduce the discrete-time, discrete-state Markov process, called a Markov chain, and then extend the concept to an HMM.

3.1. Markov Chain

A first order Markov chain is a stochastic process whose outcome is a sequence of T observations $O = o_1 o_2 \cdots o_T$ where each observation belongs to a finite set of N states $\{ s_1, s_2, \cdots, s_N \}$. If the outcome at time t is s_i, then we say that the system is in state s_i at that time. Any observation depends only upon the immediately preceding observation and not upon any other previous observation. For each pair of states $\{ s_i, s_j \}$, a_{ij} denotes the likelihood that the process goes to state s_j immediately after being in state s_i

$$a_{ij} = P(s_i \rightarrow s_j) = P(s_j \text{ at time } t+1 \mid s_i \text{ at time } t). \tag{3.3}$$

A Markov chain is entirely defined by the (A, π) where: $\pi = \{ \pi_1, \pi_2, \cdots, \pi_N \}$ is the initial state occupancy probability distribution, and A is the state transition matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}. \tag{3.4}$$

In many real word problems and in particular for speech, the only observable entity that we have is the acoustic signal. The states themselves are not observable entities, they are hidden. Because Markov chains cannot be used to model this problem, the model must be extended to include probabilistic functions as a mapping of the feature observations to the hidden state, called Hidden Markov Model [20]. The Hidden Markov Model is a doubly stochastic model, defining:

- A stochastic process that describes the state evolution, and
- A stochastic process for each state that describes the observation belonging to the state.

The rest of this chapter is devoted to the HMM and its application to speech recognition.

3.2. The Hidden Markov Model

As we have already stated above, the HMM is a doubly stochastic model with one random process describing the state evolution and a second describing the output observations inside each state. It can be defined as triple (A, B, π), where A is the transition probability distribution, $B = \{b_j(o_t)\}$ is the set of observation distributions inside states, that can be a set of matrices (discrete) or a set of distributions (continuous), and π is the initial state. In practice, we adopt the following simplified notation to indicate the HMM parameter set λ:

$$\lambda = (A, B, \pi). \tag{3.5}$$

3.3. HMM Training and Recognition

In order for our model to be useful, there are three problems that must be solved:

1. Given a model $\lambda = (A, B, \pi)$ and a sequence of observations $O = o_1 o_2 \cdots o_T$, we need to compute the probability of occurrence of that sequence.
2. Given an observation sequence O and a specific model λ, we need to identify the sequence of states $S = s_1 s_2 ... s_T$ that corresponds to the observation sequence.
3. Given training utterances, we need to find model parameters (A, B, π) to fit the data.

3.3.1. Forward and Backward Recursion

The first problem is one of computing a value for $P(O \mid \lambda)$. For a given HMM λ, an observation sequence $O = o_1 o_2 \cdots o_T$ and a state sequence $S = s_1 s_2 \cdots s_T$, the probability of the observation sequence O under the model λ can be written as

$$P(O \mid \lambda) = \sum_{all\ S} P(O, S \mid \lambda) . \tag{3.6}$$

The joint probability that O and S occur at the same time given the HMM λ can be obtained using the Bayes rule

$$P(O, S \mid \lambda) = P(O \mid S, \lambda)\ P(S \mid \lambda) . \tag{3.7}$$

The computation of the two terms in the right side of this equation is quite simple:

$$P(O \mid S, \lambda) = b_{s_1}(o_1) b_{s_2}(o_2) \cdots b_{s_T}(o_T) \tag{3.8}$$

$$P(S \mid \lambda) = \pi_{s_1} a_{s_1 s_2} a_{s_2 s_3} \cdots a_{s_{T-1} s_T} , \tag{3.9}$$

resulting in the probability $P(O \mid \lambda)$ as

$$P(O \mid \lambda) = \sum_{all\ S} \pi_{s_1} b_{s_1}(o_1) a_{s_1 s_2} b_{s_2}(o_2) ... a_{s_{T-1} s_T} b_{s_T}(o_T) . \tag{3.10}$$

Since the number of possible state sequences is N and there are T terms in each product, the calculation of $P(O \mid \lambda)$ using the formula above has a computational complexity in the $O(T\, N^T)$. Fortunately, more efficient techniques exist to solve this problem. In the following section we will present these techniques, called the forward and backward recursions, and explain how they are used to solve our problem.

3.3.1.1. Forward Recursion

The forward/backward algorithm is a dynamic programming approach to compute $P(O \mid \lambda)$ developed to improve computational complexity.

We define the forward variable as

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, s_t = i \mid \lambda). \tag{3.11}$$

This is the joint probability of having seen the partial observation $(o_1, \cdots, o_t)$ and being in state i at time t. The computation of the forward variable can be done recursively:

1. Initialization

The initial value of the forward variable is the joint probability of state i and the initial observation o_1.

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N \tag{3.12}$$

2. Induction

The induction step, which is the main step of the computation of the forward variable, can be derived as follows

$$\alpha_{t+1}(j) = P(o_1 o_2 \cdots o_t, o_{t+1}, s_{t+1} = j \mid \lambda). \tag{3.13}$$

Summing over all possible previous states i gives

$$\begin{aligned} \alpha_{t+1}(j) &= \sum_i P(o_1 o_2 \cdots o_t o_{t+1}, s_t = i, s_{t+1} = j \mid \lambda) \\ &= \sum_i P(o_1 \cdots o_t, s_t = i \mid \lambda) P(s_t = i, s_{t+1} = j \mid \lambda) P(o_{t+1} \mid s_t = i, \lambda) \\ &= \sum_i \alpha_\tau(i) a_{ij} b_j(o_{t+1}) \end{aligned} \tag{3.14}$$

so that we have

$$\alpha_{t+1}(j) = \left[\sum_i \alpha_t(i)\, a_{ij} \right] b_j(o_{t+1}) \quad 1 \le t \le T-1, \quad 1 \le j \le N, \tag{3.15}$$

which is the recursion formula.

3. Termination

The termination formula of the forward recursion is simply

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) . \tag{3.16}$$

3.3.1.2. Backward Recursion

The backward variable is defined as:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T \mid s_t = i, \lambda) . \tag{3.17}$$

This is the probability to have seen observations from time $t+1$ on, given both that we are in state i at time t and the model λ. As with the forward case, the 3 steps required to compute the backward variable are:

1. Initialization

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{3.18}$$

2. Recursion

$$\beta_t(i) = \sum_j a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad 1 \le i \le N, \quad t = T-1, \cdots 1 \tag{3.19}$$

3. Termination

$$P(O \mid \lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_j(1) \tag{3.20}$$

Using the forward and backward recursion we finally simplify the computation of the likelihood of the observed data under the model λ:

$$P(O \mid \lambda) = \sum_i P(s_t = i, O \mid \lambda) = \sum_i \alpha_t(i) \beta_t(i) . \tag{3.21}$$

From the backward and forward probability calculation, we notice that any of the states can be a valid starting or ending state. In some cases, we can assume that the state sequence starts at state 1 and ends at state N. This simplification does not hold for the general HMM case but turns out to be useful in the case of continuous speech

recognition. This constraint is set by the addition of entry and exit states to the models. These are non-emitting states (states that have no observations associated with them at anytime) that will be used to tie several HMMs together in the case of sentence-level recognition. The transition probabilities out of the entry state are the initial state probabilities, and the transition probability out of the exit state is zero.

3.3.2. The Viterbi Algorithm

In the second problem, we try to uncover the hidden part of the model: find the best state sequence that corresponds to the given observation sequence. The solution to this problem is not unique, depending on the criterion that is to be optimized. Generally we use the Viterbi algorithm [21], to solve for the globally optimal path that is the state sequence with the maximum likelihood $P(O,S \mid \lambda)$. It is a dynamic recursion, which is based on the quantity:

$$\phi_t(j) = \max_{o_1 o_2 \cdots o_t, s_1 \cdots s_{t-1}} P(s_1 \cdots s_{t-1} s_t = i \mid \lambda). \tag{3.22}$$

The steps that make the Viterbi algorithm are similar to those of the forward and backward procedure, except that the maximum is kept at each step, rather than a sum of all possible sequences.

1. Initialization

$$\phi_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N \tag{3.23}$$

2. Recursion

$$\phi_t(j) = \max_i [\phi_{t-1}(j) a_{ij}] b_j(o_t) \quad 1 \le j \le N \quad 2 \le t \le T \tag{3.24}$$

3. Termination

$$P_{\max}(S,O \mid \lambda) = \max_i \phi_t(i) \tag{3.25}$$

The value of the state that gives the maximum value is kept at each step, and at the end of the recursion, the optimal state path is found by backtracking. We can also keep track of the N-best partial state paths instead of the single best partial state path, so that a lattice of likely state sequences is kept.

3.3.3. The Baum-Welch Re-estimation

The third problem is that of the optimization of the model parameters. We use the observation sequences taken from training data to adjust the model parameters. To create the best models for our task, we learn model parameters to match the observed training data such that the observation probability $P(O)$ will be maximized over the entire training data and models. There is no globally optimal method of estimating the model parameters, which include both the transition probabilities and the observation distributions (the third parameter π_i of the model, the initial state occupancy probability, is not needed in the case of continuous speech recognition [1]). Statistical estimation techniques such as Maximum Likelihood (ML) yield locally maximized estimates. In the particular case of speech recognition, the set of parameters of the model is estimated using an iterative procedure known as Baum-Welch re-estimation [22] method that is a form of the Expectation Maximization (EM) [23] method. EM is a technique that is used to estimate model parameters specifically in the case of missing data.

The re-estimation formulas for a discrete observation HMM are given below,

$$a_{ij}^{'} = \frac{Number\ of\ transitions\ \ from\ state\ i\ to\ \ j}{Number\ of\ \ transitions\ out\ of\ \ state\ i} \tag{3.26}$$

$$b_j^{'}(o_t) = \frac{Number\ of\ \ times\ observing\ o_t\ \ from\ state\ \ j}{Number\ of\ \ times\ in\ state\ \ j}, \tag{3.27}$$

where the right side of these equations is evaluated using the values of a_{ij} and $b_j(o_t)$ obtained from the previous iteration. We then introduce the one-state occupancy probability

$$\gamma_i(t) = P(s_t = i \mid O, \lambda) \tag{3.28}$$

and the two-state occupancy probability

$$\varepsilon_{ij}(t) = P(s_t = i, s_{t+1} = j \mid O, \lambda). \tag{3.29}$$

Recalling that $P(s_t = i, O \mid \lambda) = \alpha_t(i)\beta_t(i)$, it follows that

$$\gamma_i(t) = \frac{P(s_t = i, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} \tag{3.30}$$

and

$$\varepsilon_{ij}(t) = \frac{P(s_t = i, s_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O \mid \lambda)}. \tag{3.31}$$

Replacing these two expressions in the re-estimation formulas of the state transition probability and the output distribution gives

$$a_{ij}^{'} = \frac{\sum_{t=1}^{T-1} \varepsilon_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{3.32}$$

$$b_j^{'}(o_t) = \frac{\sum_{t=1}^{T} \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)}, \tag{3.33}$$

where the notation $s_j \rightarrow o_t$ means the observation at time t is emitted by the state j.

In the case of continuous observation HMMs, the form of the distributions $B = \{b_j(o_t)\}$ determines the re-estimation formulas. For the most common case, Gaussian,

we have parameters defined by means and variances, with re-estimation formulas given by

$$\mu_j^{'} = \frac{\sum_{t=1}^{T} \gamma_j(t) o_t}{\sum_{t=1}^{T} \gamma_j(t)} \tag{3.34}$$

$$\Sigma_j^{'} = \frac{\sum_{t=1}^{T} \gamma_j(t)(o_t - \mu_j)(o_t - \mu_j)^{'}}{\sum_{t=1}^{T} \gamma_j(t)}, \tag{3.35}$$

where $\gamma_j(t)$ is the probability of being in state j at time t.

Chapter 4 Co-articulation Effects in Speech and Models for Co-articulation

4.1. The Choice of the Speech Unit to Be Modeled

When building a speech recognizer, the choice of which unit of speech to model is an important factor. If word units are chosen, then the number of models will increase as a function of task vocabulary. For any new word, new examples have to be recorded and a new model trained, necessitating a closed-vocabulary system. For these reasons, sub-word units such as phonemes are more efficient for medium or large vocabulary applications. The basic unit used to describe speech at the linguistical and acoustical level is typically the phoneme. Since any new word can be formed by concatenation of phonemes, this allows system flexibility as vocabulary extension without making new templates.

American English has about 42 phonemes [16], which can be grouped into vowels, semivowels, diphthongs, fricatives, stops, affricates, and nasals. As previously mentioned, there are many sources of variability that determine the complexity of the resulting acoustic waveform. Co-articulation is one of the most significant of such factors, and is an important phenomenon for deciding about the correct model for a phoneme [24]. In this chapter, we will first look at how co-articulation occurs in speech and then present the different approaches that have been introduced to deal with this problem as well as their weaknesses. We will use that knowledge to develop a new method later in this document.

4.2. Co-articulation Effects in Speech

It takes only a fraction of second to produce each phoneme, but the overall articulatory motion is very smooth. This smoothness results from the coordination and timing of the articulators' movements by the brain to form the proper vocal tract shape. The movements of these articulators – lips, tongue, jaw, velum, and larynx – are coordinated so that movements needed for adjacent phonemes are simultaneous and overlap each other in time, thereby causing sound patterns to be in transition most of time. Consecutive phonemes are articulated together to facilitate pronunciation during natural speech.

Time is required for our articulators to effectively reach target position as well as to make the transition between these targets [16]. When an articulator movement, called a gesture, for one phoneme is not in conflict with that of a following phoneme, the articulator may move toward a state appropriate for the following phoneme. This is referred to as left-to-right anticipation [16] [25]. Similarly, because of the target for a phoneme on the right, the movement of articulators of the prior phoneme on the left are modified. As a consequence, movements of some articulators start earlier [27] [28], so that each articulator may move toward its next required state as soon as the last phoneme that needs it has finished. These interactions do not generally affect the way the sound is perceived by our auditory system. For example, lip rounding for a vowel usually commences during preceding nonlabial consonants; the formant lowering that the rounding imposes does not cause these consonants to be perceived differently [25] [29]. It is of interest to know when and how co-articulation occurs. Not all articulators co-articulate in all contexts. The phoneme ends as one or more articulators move toward

positions for the next phoneme, causing acoustic changes in the speech signal. A phoneme's "articulation period" exceeds its "acoustic period" [26]. In fact, the gesture for a phoneme starts during a previous phoneme and finishes during the next one. The time of largest acoustic changes between phonemes, usually identified as the phoneme boundaries, are associated with changes in manner of articulation, which often involve vocal tract constriction. Most of the time, the motion of articulators directly involved in a phoneme contraction specifies the boundaries of its phone, while other articulators are free to co-articulate. Phonemes with labial constriction allow the tongue to co-articulate, and lingual phonemes permit labial co-articulation, e.g., the lip-rounding feature of a labial spreads to adjacent lingual consonants [25]. Lingual phonemes are phonemes that are pronounced with the aid of the tongue like /t/, /d/. Labial phonemes are phonemes that are pronounced with the aid of the lips like /b/, /m/. The velum is lowered in advance to the pronunciation of nasal consonants, causing the spread of nasalization to adjacent phonemes. In vowel co-articulation, the tongue is moved toward targets of the previous or the next phonemes.

Depending on the phoneme sequences, articulators move from positions for one phoneme to the next. The speech signal during this transition is affected by context. This is obvious in the formant transitions before and after oral-tract closure for stops and nasals. The amount of co-articulation depends on the speaking rate and style. Undershooting of co-articulation (caused by the spread of articulator motion among adjacent phonemes which occur when moving from one phoneme to the next) in moving toward the phoneme target occurs most often when one speaks rapidly [30] [31].

In actual speech, steady state position and formant frequency targets for many phonemes are rarely reached completely. In fact, many recent linguistic models emphasize the importance of dynamic articulatory gestures and suggest that transitions, not steady-state targets, may be the units of speech production [30]. Co-articulation effects beyond the immediate surrounding phonemes only help the speaker to reduce effort, and this level of co-articulation is not required for fluent speech production. Acoustic variability, such as changes in duration or formant frequencies across different sounds for the same phoneme, can be separated into inherent variability and effects of context. For the same speaker, the acoustic variations produced in identical phonetic contexts are ranged from 5-10 ms in phone duration and 50-100 Hz in F1-F3. Variations in different contexts beyond these amounts are mainly due to co-articulation. Incorporating co-articulation into our phonetic model is very important.

4.3. Models for Co-articulation

Due to factors such as co-articulation, each phoneme will have a variety of acoustic manifestations or realizations. For our case, where only co-articulation effect is taken into account, the differences among acoustic realizations of the same phoneme are caused primarily by the preceding and following phonemes. These phonemes are given the name left and right context, respectively. In order to address this problem, information about the surrounding phonemes should be incorporated into the phoneme model. Monophones do not preserve this information since all acoustic realizations of each phoneme are used together to estimate a model for the phoneme. Several techniques have been proposed to deal with this problem. Early attempts included capturing feature movements in the parameter space by adding feature derivatives. To implement this,

dynamic coefficients such as deltas and delta-deltas (defined in Chapter 2) can be added to the feature vector [32]. Another attempt to capture dependency is the use of triphone models, where right and left context information is preserved in the model. Besides these techniques, other types of model that address this issue include stochastic trajectory models (STM), and trended HMMs. These models, because they model time variation of articulatory motions at local level (variation inside states, called local variation), or include information about the neighboring phonemes, have made some recognition improvements compared to monophones. We will next discuss the main idea of each of these models and understand why they are successful.

4.3.1. Triphone Model

The triphone model is a context-dependent model that includes contextual information at the cost of training many more base units. It uses both left and right context information to capture dependency of the observations, which is important to represent continuity and co-articulation effects in the speech [33]. In this framework, the phoneme model is then conditioned by the right and left phonemes. For example, the word "elephant," will give the following monophone and triphone labeling:

- Monophone models:

< eh l ah f ah n t >

- Triphone models:

< eh+l eh-l+ah l-ah+f ah-f+ah f-ah+n ah-n+t n-t >,

where "+" is used to separate the phoneme from the right context and "-" to separate the phoneme from it left context.

The observation distributions are still state conditioned, independent and identically distributed (IID), as in the HMM (Chapter 3). Experimental results on large training data sets have shown significant improvement of triphone-based recognition systems over that of monophones. The problem with this model is its huge demand in training data to allow all possible triphones to be learned.

4.3.2. Biphone Models

Biphones, like triphones, are context dependent models, where only one of the two contexts is considered. Biphone models with only left context are called left biphones and biphone models with only right context are called right biphones. In the word "elephant" in our previous example, we have the following left and right biphones labeling:

- Left biphone models:

< eh eh-l l-ah ah-f f-ah ah-n n-t >

- Right biphone models:

< eh+l l+ah ah+f f+ah ah+n n+t t >.

4.3.3. Trended HMM

The trended HMM models the parameter's evolution inside each state by a regression function in time as opposed to HMM where the value of the parameter inside each state is fixed. The trended HMM is a discrete time Markov process with its states associated with a distinct time series. The observation is partitioned into two components. The first is a deterministic function of time G_t given by a mathematical function of the parameter of the state. The second is a random component R_t that follows a zero-mean IID Gaussian distribution. This later is called the residual.

The observation time-series model becomes

$$O_t = G_t + R_t, \tag{4.1}$$

and the parameter variation obeys a Markov chain [43]. With the observations modeled by a time-series with deterministic trends, this model leads to a local or state conditioned non-stationarity. The overall model consists of an HMM with trend-plus-residual, which has been called trended HMM for simplicity [34] [35]. The model still relies on a Markov chain for it global non-stationarity. The parameters of the model are $\mathrm{A}, \Theta, \Sigma$, and the observation distribution at any time t is given by:

$$\mathrm{O}_t = G_t(\Theta_i) + R_t(\Sigma_i), \tag{4.2}$$

where A is the state transition probability matrix of the homogeneous Markov chain,

Θ_i is the state dependent parameter used in the deterministic trend function,

Σ_i is the covariance matrice of R_t,

R_t is a zero-mean Gaussian IID residual,

i is the state given by the Markov chain evolution, and

$R_t(\Sigma_i)$ is assumed to have the following zero-mean multivariate Gaussian distribution:

$$\frac{1}{(2\pi)^{D/2}\left|\Sigma_i\right|^{1/2}} \exp\left\{\tfrac{1}{2} R_t^T \Sigma_i^{-1} R_t\right\}, \tag{4.3}$$

where D is the dimension of the observation space.

The computation of the density function of a sequence of observations is derived from the preceding Gaussian distribution. In this model, the state-conditioned observation distributions have the same variance, but different means. The mean of the observations inside a given state is model by a trended function, which assigns a value to the mean

according to the time the observation vector is uttered. As a consequence the observation sequence is no longer state conditioned stationary.

Some of the most common trend functions are polynomial regression functions [36]. An example of such model is

$$O_t = \sum_{m=0}^{M} B_i(m) t^m \quad + \quad R_t(\Sigma_i). \tag{4.4}$$

The standard HMM is itself a special case of the trended HMM, with constant trend function. Experiment fitting of the trended HMM to actual speech data has shown that the data fitting performance of the trended HMM is superior to that of the standard HMM.

4.3.4. Stochastic Trajectory Modeling

One problem with the HMM is that the model rely onlies on independent observations, which does not preserve the trajectory of consecutive observations. The stochastic trajectory model (STM) approaches this problem by modeling parameter evolution using a sequence of observation vectors. Phoneme based speech models for STM [37] [38] [39] use clusters of trajectories to represent acoustic observations of a phoneme. Trajectories are represented by a random sequence of states and the mathematical model for a trajectory is a mixture of probability distributions [40] [41]. The state itself is associated with a multivariate Gaussian density function. Since a cluster of trajectories only contains trajectories of fixed duration, time rescaling is used to accommodate duration variation in the observed trajectories to those of the modeled trajectories.

Given X_n as a sequence of n vectors, the joint density function of X_n, the phoneme s, the trajectory T_k, and the duration d of the is computed as

$$p(X_n, T_k, d, s) = p(X_n \mid T_k, d, s)\Pr(T_k \mid d, s)\Pr(d \mid s)\Pr(s) \tag{4.5}$$

$$p(X_n, T_k, d, s) = p(X_n \mid T_k, d, s)\Pr(T_k \mid d, s)\Pr(d \mid s)\Pr(s), \tag{4.6}$$

where $\Pr(s)$ is the initial state occupancy likelihood,

$\Pr(d \mid s)$ is the duration probability, such as Γ- distribution [42] [37],

$\Pr(T_k \mid d, s)$ is independent of the duration, and

$\Pr(X_n \mid T_k, d, s)$ is the product of n IID Gaussian distribution of the n point that composed the trajectory T_k.

Stochastic trajectory modeling has also been successful in improving recognition accuracy over standard HMMs [42].

Chapter 5 New Design Method

5.1. Problem Formulation

The most successful speech recognition systems have been based on HMM modeling of acoustic units or sub-units [1]. HMMs were initially used to model acoustic observations associated with each phoneme (monophones). But, studies have shown that a phoneme is highly influenced by its immediate surrounding phonemes. This co-articulation effect, discussed in Chapter 4, is lost in monophone modeling. To capture the co-articulation effect caused by the acoustic contexts, several methods have also been proposed. As already stated, the method we consider here is triphone. Triphone parameters are typically estimated through training and for the training process to be successful, large amounts of well-balanced data are needed. This results in costly material, transcription, and very long training time. Different parameter tying approaches have been used to solve this problem such as:

- Data-driven clustering, and
- Decision tree-based clustering.

5.1.1. Tying of Acoustically Similar State

A tied state system [10][11] is one in which similar states share the same set of parameters. When re-estimation occurs, all of the data that would have been used to estimate each of the individual untied states is used to estimate the single tied state [12]. This allows robust parameter estimation of the model. Whole models can also be tied, by allowing similar models to share the same set of parameters. We will discuss the two most common ways of tying state.

5.1.2. Data-Driven Clustering

In typical data-driven clustering [33], all states are first placed in singleton clusters (each cluster contains exactly one state). Let us assume we have n different states; the first step results in n individual clusters. Then $n-1$ clusters are formed by merging the closest pair of clusters. We repeat this process until the size of the largest cluster reaches a predefined threshold. The similarity between clusters is measured by a distance metric.

5.1.3. Decision Tree-Based Clustering

The data-driven clustering method does not deal with unseen triphones. Unseen triphones are triphones for which there is no example in the traning data. Decision tree-based[13] [14] clustering is the procedure that is used to address this issue. Instead of using distance metrics for similarity measure, this technique consists of a binary tree in which a yes/no phonetic question is attached to each node. Contrary to data-driven clustering, which is a top-down design approach, in this method, all the states are put into the root node of the tree and go through each question until they reach the leaf nodes. The leaf-nodes constitute the final clusters. This technique can also be used to cluster whole models.

5.1.4. New Approach to Triphone Creation

In our work, we address these issues by learning rule-based trajectory interpolations to build triphones directly from monophones. This provides both a means to generate models for all possible triphones and a method for dealing with training data cost, transcription cost and training time. The goal of this study is not merely to design triphones that will yield to the best performance but also to investigate consistency in the

way co articulation affects acoustic speech units using real data, thereby making a bridge between the known theory of co-articulation and what is observed from real data. Considering the co-articulation effect at the physiological level, we ask ourselves if the process that generates co-articulation is purely random and unpredictable or if instead it is possible to learn how individual phonemes actually combine to form triphones. In our experiments, we will try to answer this question.

5.2. Triphone Creation Method

As stated earlier, the fundamental goal in triphone creation is to incorporate co-articulation effects due to the left and right phonetic contexts in the phoneme model. Triphones are directly created from monophones using rule-based interpolation. This rule based interpolation represents the rules for combining individual phonemes to yield triphones. The principal goal of this design method is to be able to determine these rules using our knowledge of the problem so they will generalize across tasks. Once we generate these rules, they will be used to create triphones for speech recognition systems directly from the monophones. This new design procedure can be divided basically into three major steps as follows:

- Step 1: Analysis of real triphones and monophones
- Step 2: Rule based interpolation learning
- Step 3: Triphone creation using rule based interpolation.

5.2.1. Analysis of Real Triphones and Monophones Data

The set of models that we will use to derive this interpolation approach consists of trained triphones and monophones. The trained triphones constitute our reference triphone models, as opposed to interpolated triphones we will learn without training. The

rule will be learned from these trained triphones and applied to a different data set than that used to train the triphones.

The parameter set for these includes mean vectors, covariance matrices, and transition matrices of the monophones as well as those of the triphones. Each phoneme is represented by a three state HMM. For simplicity, we use only one Gaussian to model each state.

The first parameters we consider are the transition probability matrices. The normal practice is to tie all transition matrices of triphones of a given base monophone, and we have done that here. This allows parameter sharing and good estimation of transition probabilities. Because of this, we do not have sufficient information to learn how transition matrices of three monophones combine to form the transition matrix of a triphone, or reason to believe that this should have impact on triphone model characteristics. Thus we assume that the state transition probabilities of the phoneme do not undergo major changes from one realization of the phoneme (triphone) to another. The other parameters that we consider are the state parameters: the mean and covariances of each state. To study the change in the parameters among triphones of the same phoneme, we plotted the mean vector and the covariance vector of the models. As shown in Figures 5-8, the parameters of the center state of the triphones are almost identical to those of the corresponding monophone. However, the model parameters in the first and third states are affected by a considerable change due to the context of the reference triphones. This confirms the fact that co-articulation effects are greater at the extremities of the phoneme (transitions between phonemes) than at the center of the phoneme (target

sound). The change observed clearly depends on the nature of the phonemes that precedes and follows the triphone.

This result motivates our choice of the method we used. In our new design method, we model the parameters of the triphone as mathematical functions of the parameter set of the three monophones, which define it. The detailed implementation of this method is given in the next section.

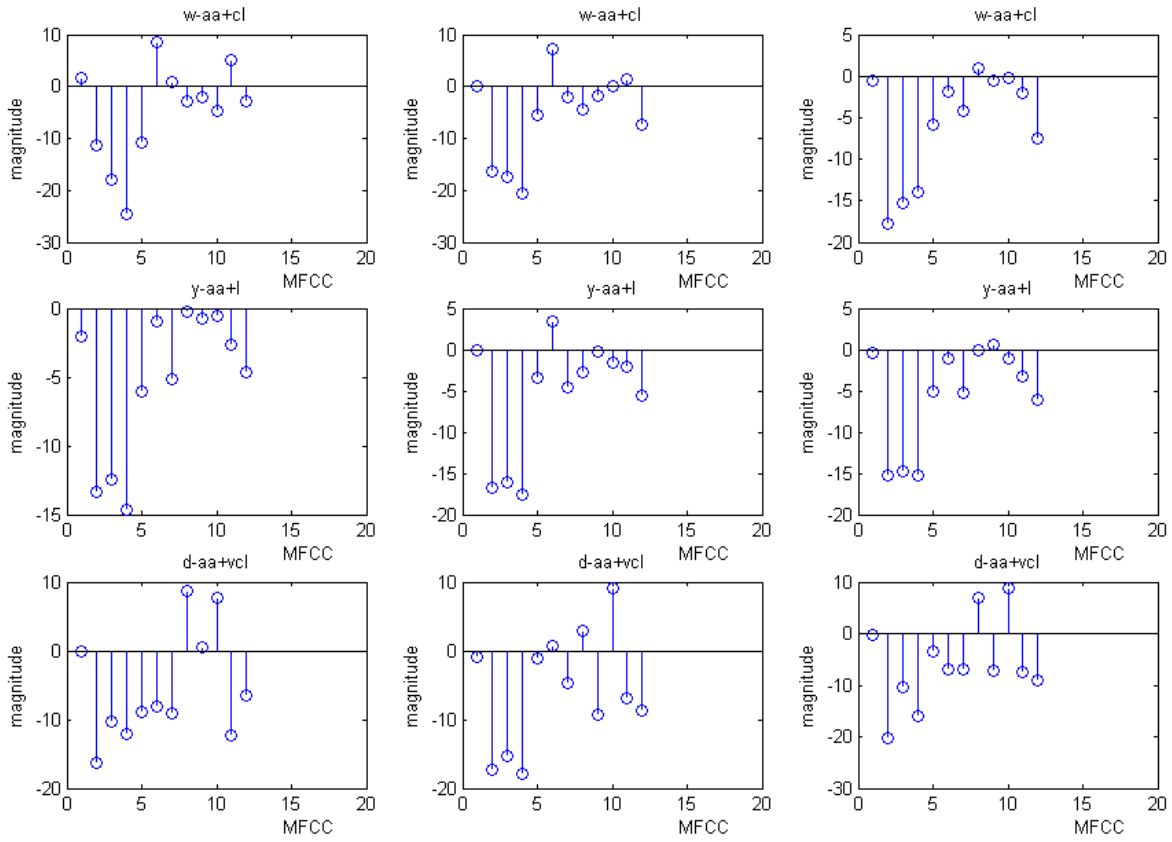

Figure 5: Plot of mean of triphones of /aa/, begining, center and end states shown

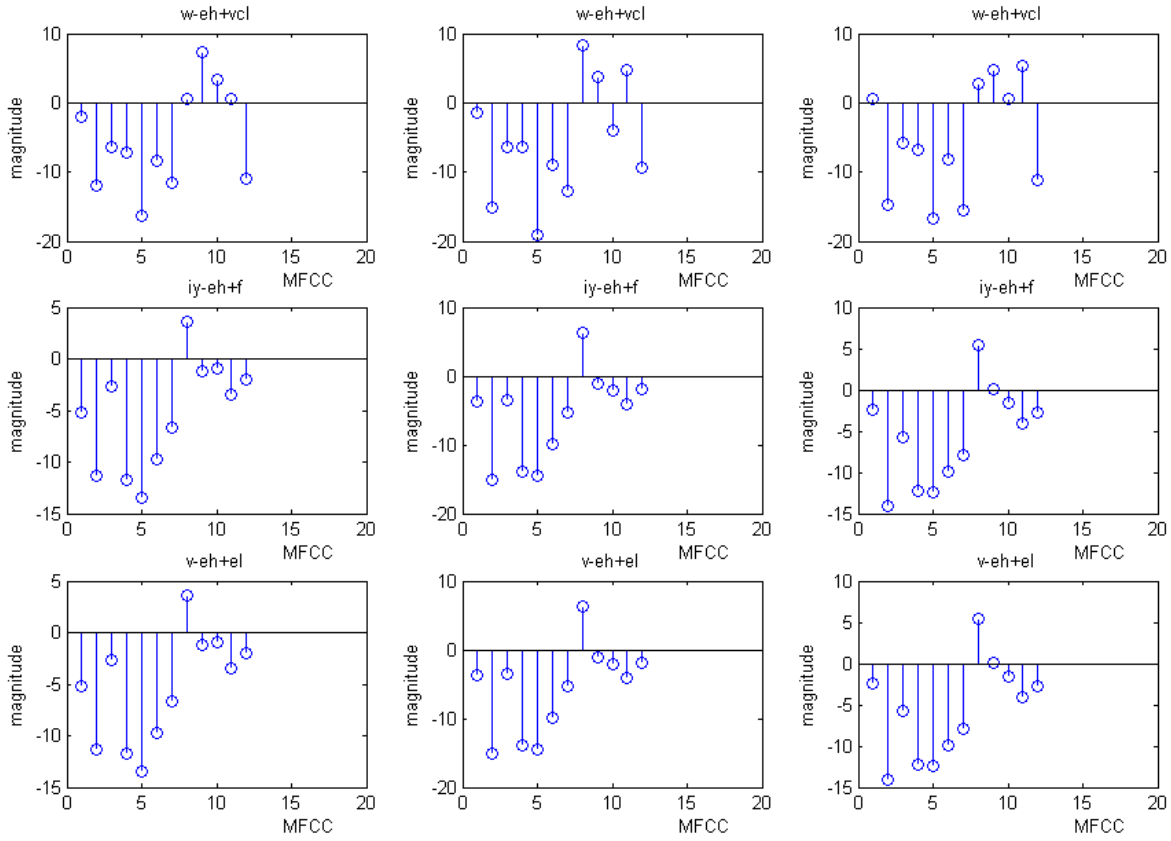

Figure 6: Plot of mean of triphones of /eh/, begining, center and end states shown

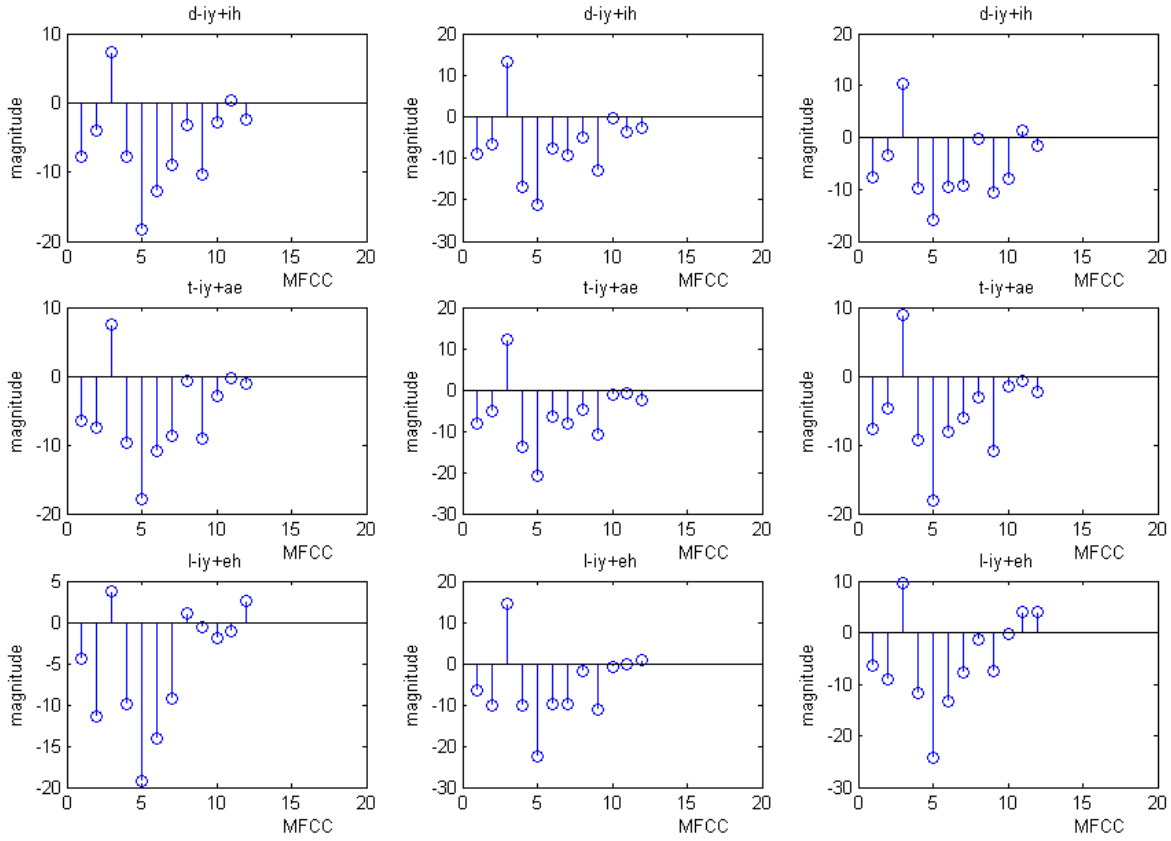

Figure 7: Plot of mean of triphones of /iy/, begining, center and end states shown

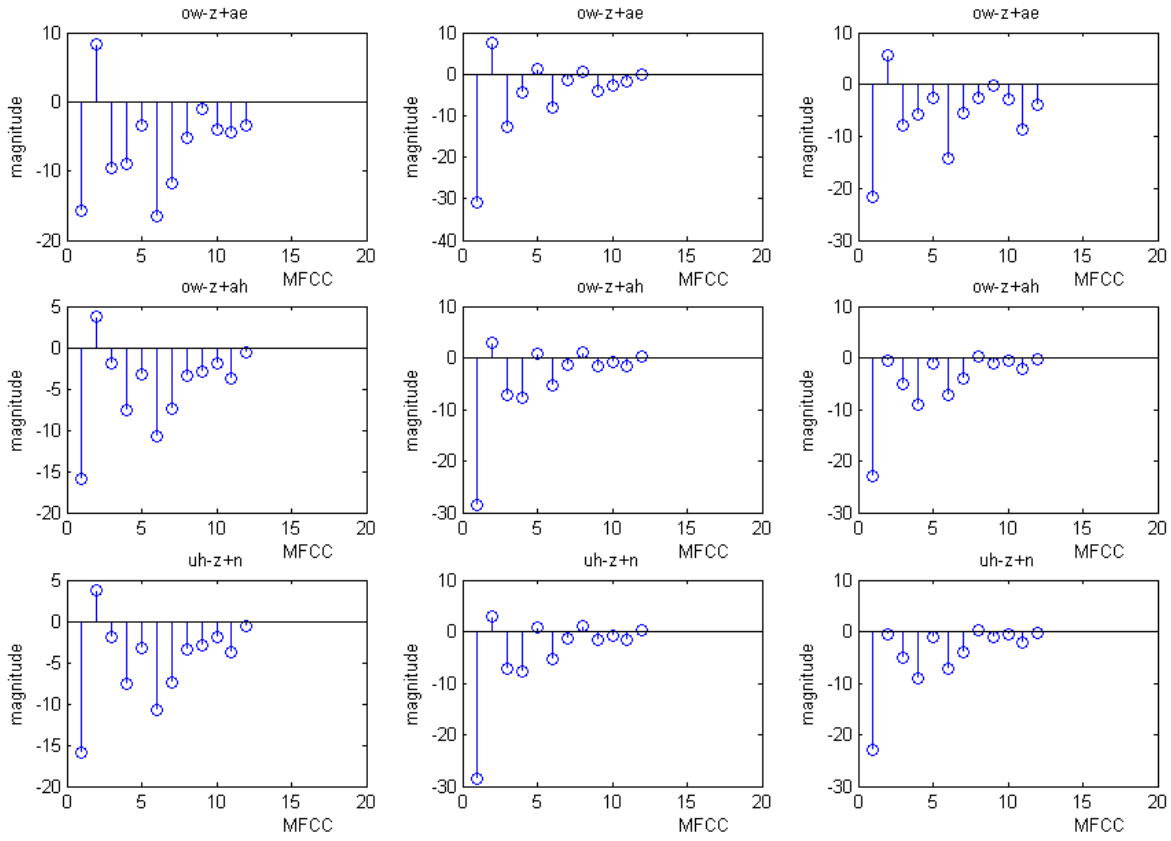

Figure 8: Plot of mean of triphones of /z/, begining, center and end states shown

5.2.2. Interpolation Based Triphone Methods

We pose our problem as follows: for each phoneme, we desire to learn how its corresponding triphones are built. In other words, for each phoneme p, we want a function F_p such that any of its triphones can be written as:

$$(l - p + r) = F_p(l, p, r) \tag{5.1}$$

where l is the left context monophone, r is the right context monophone, and $l - p + r$ is the triphone for p in that specific context.

In this approach, we assume that there are consistencies in the way monophones combine to form triphones. In reality, we want to investigate if there is such consistency. If the process that combines monophones into triphones is completely random, our design will be unsuccessful. There is reason to believe, however, based on when and how co-articulation occurs (Chapter 4), that this is not the case [16] [25-28].

In this method, and in all our methods, we only consider the means of the models, and we assume the mean to be of dimension d.

From our analysis in Section 5.2.1 we decide that the center state of the phoneme will not vary, and the first and third states are the ones that are affected by co-articulation. Moreover, for the linguistic reasons, discussed in Chapter 4, we consider that this effect comes from the two adjacent target states (center states), and the states between them. This allows us to expend Equation (5.1) as follows:

$$(l - p + r)_1 \quad = \quad h\,(l_2, l_3\,, p_1, p_2) \tag{5.3}$$

$$(l - p + r)_2 \quad = \quad p_2 \tag{5.4}$$

$$(l - p + r)_3 \quad = \quad g\,(p_2, p_3\,, r_1, r_2)\ , \tag{5.5}$$

where we consider a 3 emitting states left to right HMM topology:

p_i is the mean of the i^{th} state of the phoneme p,

$(l-p+r)_i$ is the mean of the i^{th} state of the triphone p,

h is the interpolation rule for the left context of the interpolated triphone, and

g is the interpolation rule for the right context of the interpolated triphone.

Equation (5.3) means that state one of the triphone corresponds to a transition between target sound l_2 and p_2, where aspects of this transition are already present in l_3 and p_1. And because the co-articulation effect is more important at the extremities, we will use only l_3 and p_1. Therefore we can further simplify our model to:

$$(l-p+r)_1 = h(l_3, p_1) \tag{5.6}$$

$$(l-p+r)_2 = p_2 \tag{5.7}$$

$$(l-p+r)_3 = g(p_3, r_1). \tag{5.8}$$

We will introduce four different methods to build interpolated triphones. These methods range from low complexity to high complexity. The first method is just an averaging of the models parameters, and no coefficients are learned to build the interpolated triphones. The second method also is a low complexity method, in which only 4 coefficients are learned to build the interpolated triphones. The third method is a medium complexity method, in which 52 coefficients are derived to generate the interpolated triphones. The last one of these methods is a high complexity method in which 676 coefficients are generated to build the interpolated triphones.

5.2.2.1. Design Method 1: Averaging

In our first method, we form the new mean of the leftmost state of the interpolated triphone, $(l-p+r)_1$, by averaging the mean of the rightmost state of the left context l_3 and that of the leftmost state of the monophone p_1. The center state of the triphone is obtained as in Equation (5.7), so that the center state of the monophone is used as the center state of the interpolated triphone. To obtain the mean of the rightmost state of the monophone, $(l-p+r)_3$, we average the mean of the rightmost state of the monophone p_3 with the leftmost state of the right context r_1.

The three equations that comprise this method are summarized below:

$$(l-p+r)_1 = 0.5 \cdot l_3 + 0.5 \cdot p_1 \tag{5.9}$$

$$(l-p+r)_2 = p_2 \tag{5.10}$$

$$(l-p+r)_3 = 0.5 \cdot p_3 + 0.5 \cdot r_1 . \tag{5.11}$$

It is clear that in this method, we are setting the interpolation coefficient to an arbitrary value 0.5 rather than learning the coefficient. The total number of parameters that we have for this interpolation approach is zero. The underlying hypothesis for this method is that each of the context monophones contribute the same amount toward the formation of the state of the interpolated triphone. Later on, in the results section, we will see that this hypothesis does not hold.

5.2.2.2. Design Method 2: Interpolation with 4 Coefficients per learned triphone

In the first method, we used an arbitrary value of 0.5 for the interpolation coefficient to build the interpolated triphone from the monophones. Here, we learn the weights that best fit, using our trained triphones and the monophones. To do this, we model the mean of the leftmost state of the triphone $(l-p+r)_1$ as a weighted sum of the

means of the rightmost state of the left context l_3 and that of the leftmost state of the monophone p_1 (Equation (5.12)). Similarly, the rightmost state of the interpolated triphone $(l-p+r)_3$ is modeled as a weighted sum of the mean of the rightmost state of the monophone p_3 and the mean of the leftmost state of the right context r_1 (Equation (5.14)). The center state of the interpolated triphone is again obtained by direct replacement. The set of equations that describe this process is given below

$$(l-p+r)_1 = \alpha_l \cdot l_3 + \beta_l \cdot p_1 \quad (5.12)$$

$$(l-p+r)_2 = p_2 \quad (5.13)$$

$$(l-p+r)_3 = \alpha_r \cdot p_3 + \beta_r \cdot r_1 \quad (5.14)$$

where α_l and β_l are the coefficients used to build the leftmost triphone state, and α_r and β_r are the coefficients used to build the rightmost triphone state.

For each phoneme, we need to learn α_l and β_l to define how the first state's mean is formed, and α_r and β_r to define how the third state's mean is formed, using the equations defined above.

This is a simple model, but if consistency exists in the way triphones are produced, this should be a good model to start with. The parameters are learned to minimize the MSE (Mean Square Error) between the designed interpolated triphones and the trained triphones. To summarize:

1. We use monophones models and our trained triphones to learn some rules,
2. We then generate triphones using the learned rules.

This method can be used to generate:

- All the triphones needed for recognition directly from monophones, or
- Only the unseen triphones.

Learning the rules

In practice, and to reduce the number of models to store in memory for each phoneme, we use the left biphones of the phonemes to learn α_l and β_l and its right biphones to learn α_r and β_r. This means that the references we have used are actually trained biphones. Originally, for each phoneme, we have to store about 2304 models permanently into memory in order to learn it interpolation mechanism. Each model by itself already requires a bit of storage space. By using trained biphones, we only have to store about 96 models permanently into memory. This clearly reduces the memory space required by a factor of almost 23 and in the same way improves the computational time.

Learning the leftmost state parameters α_l, β_l

With this modification, Equation (5.12) becomes:

$$(l-p)_1 = \alpha_l \cdot l_3 + \beta_l \cdot p_1 . \qquad (5.15)$$

Let us assume that we have N different left biphones of p in our reference biphone model set (trained biphones), denoted:

$$(l^1 - p), \quad (l^2 - p), \cdots, (l^N - p),$$

where $(l^i - p)$, $i = 1,....,N$, refers to one specific biphone model, the biphone model of phoneme p with left context = phoneme l^i.

Applying Equation (5.15) to the left biphones of p gives the following set of equations as model for the leftmost states of the interpolated triphone:

$$\begin{cases} (l^1 - p)_1 = & \alpha_l \cdot l_3^1 & + & \beta_l \cdot p_1 \\ (l^2 - p)_1 = & \alpha_l \cdot l_3^2 & + & \beta_l \cdot p_1 \\ & \vdots & & \\ (l^N - p)_1 = & \alpha_l \cdot l_3^N & + & \beta_l \cdot p_1. \end{cases} \tag{5.16}$$

We desire to minimize the mean square error (MSE) between the trained model state and the constructed interpolated model state (in this case for the leftmost state):

$$\min_x \sum_{i=1}^{N} \left\| (l^i - p)_1 \quad - \quad \alpha_l \cdot l_3^i \quad + \quad \beta_l \cdot p_1 \right\|. \tag{5.17}$$

With $\| x \| = \sum_{i=1}^{2} x^2(i)$ and $x = [\alpha_l, \beta_l]$, we have

$$\min_x \sum_{i=1}^{N} \sum_{j=1}^{d} \left[(l_3^i - p)_1(j) - (\alpha_l \cdot l_3^i)(j) + (\beta_l \cdot p_1)(j) \right]^2, \tag{5.18}$$

where the notation $p(j)$ denotes the j^{th} feature of model p.

This can be written as

$$\min_x f(x), \tag{5.19}$$

where $f(x)$ is quadratic.

This is a quadratic optimization for which there are methods to find the solution. We use Matlab functions “lsqlin.m” and “fsolve.m” that implement techniques for solving optimization problems to solve this equation.

Learning the rightmost state parameters α_r, β_r

These parameters are learned in a similar way to α_l, β_l.

The right biphones are denoted:

$$(p + r^1), \quad (p + r^2), \cdots, (p + r^N)$$

For this case, the equivalents of Equations (5.15) and (5.16) are:

$$(p+r)_3 \quad = \quad \alpha_r \cdot p_3 \quad + \quad \beta_r \cdot r_1 \tag{5.20}$$

$$\begin{cases} (p+r^1)_3 = & \alpha_r \cdot p_3 & + & \beta_r \cdot r_1^1 \\ (p+r^2)_3 = & \alpha_r \cdot p_3 & + & \beta_r \cdot r_1^2 \\ & \vdots & & \\ (p+r^N)_3 = & \alpha_r \cdot p_3 & + & \beta_r \cdot r_1^N . \end{cases} \tag{5.21}$$

The minimization MSE between the reference data and the built triphones is:

$$\min_x \sum_{i=1}^{N} \sum_{j=1}^{d} \left[(p+r_1^i)_1(j) - (\alpha_r \cdot p_3)(j) + (\beta_r \cdot r_1^i)(j) \right]^2 . \tag{5.22}$$

5.2.2.3. Design Method 3: Interpolation with 52 Coefficients per learned triphone

In this method, we increase the complexity of the interpolation model by learning one weight for each element of the mean vector, instead of learning one weight for each monophone, as in design method 2. This results in learning 26 coefficients representation for how monophones combine to form the leftmost state of the triphone and 26 more coefficients for how monophones combine to generate the rightmost state of the triphone. This increases the number of coefficients learned in this method to 52. The derived model is given by:

$$(l-p+r)_1 \quad = \quad \acute{\alpha}_l \cdot l_3 \quad + \quad \hat{\alpha}_l \cdot p_1 \tag{5.23}$$

$$(l-p+r)_2 \quad = \quad p_2 \tag{5.24}$$

$$(l-p+r)_3 \quad = \quad \acute{\alpha}_r \cdot p_3 \quad + \quad \hat{\alpha}_r \cdot r_1 , \tag{5.25}$$

where $\acute{\alpha}_l = diag([\alpha_l(1), \cdots, \alpha_l(d)])$ and $\hat{\alpha}_l = diag([\beta_l(1), \cdots, \beta_l(d)])$ are diagonal matrices that are used to build the leftmost state of the triphones, and $\acute{\alpha}_r = diag([\alpha_r(1), \cdots, \alpha_r(d)])$ and $\hat{\alpha}_r = diag([\beta_r(1), \cdots, \beta_r(d)])$ are diagonal matrices that are used to build the rightmost state of the triphones.

For each phoneme, we need to learn $\acute{\alpha}_1$ and $\hat{a}_1$ to define how the first state's mean is formed, and $\acute{\alpha}_r$ and $\hat{a}_r$ to define how the third state's mean is formed, using the equations defined above.

The derivation of the optimal interpolation coefficients is similar to that of design method 2. We use the trained biphones and the monophones to set up an equation system similar to Equations (5.16) and (5.21), which establishes the basis on how monophone states should combine to make triphone states. We then derive the optimal coefficients by minimizing the MSE between the trained and the interpolated triphones in the same way as in Equations (5.18) and (5.22).

5.2.2.4. Design Method 4: Interpolation with 676 Coefficients per learned triphone

We now increase the complexity of the model again, letting $\acute{\alpha}_1$, $\hat{a}_1$, $\acute{\alpha}_r$, and $\hat{a}_r$ defined in Equations (5.23), (5.24) and (5.25) be full matrices. We have

$$(l-p+r)_1 = \acute{\alpha}_1 \cdot l_3 + \hat{a}_1 \cdot p_1 \tag{5.26}$$

$$(l-p+r)_2 = p_2 \tag{5.27}$$

$$(l-p+r)_3 = \acute{\alpha}_r \cdot p_3 + \hat{a}_r \cdot r_1, \tag{5.28}$$

where $\acute{\alpha}_1$ and $\hat{a}_1$ are $d \times d$ matrices used to build the leftmost state of the triphones, and, $\acute{\alpha}_r$ and $\hat{a}_r$ are $d \times d$ matrices used to build the rightmost state of the triphones.

The derivation of the optimal interpolation coefficients is again similar to that of design method 2. We use the biphones and the monophones to set up an equation system similar to Equations (5.16) and (5.21), which establishes the basis on how monophone states combine to make interpolated triphone states. We then derive the optimal coefficients by

minimizing the MSE between the trained and the interpolated triphones in the same way as in Equations (5.18) and (5.22).

The size of each matrix is 13*13=169 in our case (d=13). We have in total four matrices to learn, so the total number of coefficients to be learned is 676.

In this chapter, based on our current stage of understanding of the reality of co-articulation, we propose through the use of interpolated triphones a technique to apply models of co-articulation to the problem of speech recognition. Our model addresses key issues of trained triphones, including the need for a huge amount of training data. We propose and present four different techniques to create interpolated triphones. In the first one of these methods we do not learn any coefficient but we rather use coefficients set up on arbitrary basis. Therefore, we expect this method to give poor performance. In the second method, we learn 4 interpolation coefficients and our expectation is that the performance achieved will be relatively low due to the low complexity of the method. In the third method, we expect the performance to be relatively good and noticeably better than the first two. Also, we expect this method to be consistent, that is, to generalize very well across different data sets. As for the last method, due to the number of parameters learned, our expectation is that it gives a good performance especially on the data set that is used to derive the interpolation coefficients and the data sets that are like that one. In addition, because we do not have enough data to learn 676 parameters, we also suspect that this method may not achieve the goal of capturing the mechanism underlying co-articulation. In order to evaluate the performance of these methods, we perform several tests that we present in the next chapter.

Chapter 6 Experiments and Results

The purpose of this Chapter is the implementation and performance evaluation of the new design methods. To achieve this, we run two sets of experiments for each of our four different approaches. The first experiment involves learning the interpolation mechanism for our new methods. In the second set of experiments we evaluate the performance of these new methods on new data. We use two different data sets in these experiments, Resource Management (RM) [44] and TIMIT [45]. RM is used to derive the interpolation coefficients and then these coefficients are used to build interpolated triphone recognition systems for RM (the same set used to learn them) and for the new TIMIT set to see how the rules generalized. In this section, after a brief description of the two speech data corpuses used, we will present the general setting of the system and then present the results obtained for each method.

6.1 The Resource Management (RM) and The TIMIT speech corpuses

6.1.1 Resource Management

The DARPA Resource Management (RM) [44] continuous speech corpora is a corpus of read speech designed to provide speech data for the development and evaluation of continuous speech recognition systems. It contains two main sections often referred to as RM1 and RM2. It includes utterances recorded from speakers representing a variety of American dialects. It was designed at BBN Laboratory, Inc. and RSI International.

6.1.2 TIMIT

TIMIT [45] is a corpus of read speech designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of speech recognition

systems. TIMIT contains recordings of speakers of 8 major dialects of American English. It includes time aligned orthographic, phonetic and word transcriptions for each utterance. It was co-designed by the Massachusetts Institute of Technology (MIT), SRI International (SRI) and Texas Instruments, Inc. (TI). TIMIT was prepared for CD-Rom production by the National Institute of Standards and Technology (NIST).

6.2. General Setting of the System

The phoneme set used in these experiments consists of 48 different phonemes of American English. For each of these phonemes, we build monophones, biphones, and triphones when necessary. For monophone models, each phoneme is modeled with a three state HMM. The number of mixtures is one per state and diagonal co-variances are used. We initialize the models with the global mean and variance of the data, then use Baum-Welch algorithm to perform embedded re-estimation of the model parameters.

The prototype for the biphones and triphones is the same as that of the monophones. We clone the monophones obtained previously to form an initial set of biphones by creating biphone models and initializing them with the monophone parameters. These biphones are then re-estimated using Baum-Welch embedded training. The final set of biphones that result from this procedure constitutes our reference biphones.

The steps that lead to the creation of trained triphones are exactly the same as those for trained biphones except that this time we clone the monophone into triphones.

The speech signal is first processed with 26 filter banks. The filter bank’s energies extracted are computed from a 25.6 ms Hamming window at a frame rate of 10 ms.

Twelve MFCCs plus energy are derived, making a feature vector of dimensionality equal to 13.

6.3. Design Method 1

6.3.1. Biphones and Triphones Creation

For each of the phonemes, we take the average of the mean of the monophone states involved into the creation of each state of the interpolated triphone according to equations (5.9) and (5.11). This generates the set of right and left biphones for this phoneme. The interpolated triphones are built by concatenating the first and the second states of the left biphones with the third state of the right biphones.

6.3.2. Results

After we have designed the interpolation coefficients, the next step is to evaluate the performance of our new design. We test our new design a first time with the RM database (the same set used to design the interpolation coefficients), and a second time with the TIMIT database. We then compare the performance of our design to monophone and trained triphones systems. Table 2 contains the experimental results for RM and Table 3 contains the experimental results for TIMIT. This method basically demonstrates poor performance for both RM and TIMIT task, the interpolated triphones performance is lower than those of monophones and trained triphones. These are expected results since we did not learn any coefficients for this technique.

Table 2: Experimental results for method 1 (Averaging) with RM

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	14.00	62.59	61.97
Trained triphone	19.67	66.65	66.61
Interpolated triphone	12.33	58.14	57.20

Table 3: Experimental results for method 1 (Averaging) with TIMIT

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	16.13	41.25	41.25
Trained triphone	25.81	54.86	54.86
Interpolated triphone	14.80	39.13	39.13

6.4. Design Method 2

6.4.1. Interpolation Coefficients Learning

To learn the rules, we used the RM database. Only the part of the data that was reserved for training is used. For each phoneme, we extract the mean vector of the states of both monophone and trained biphones. We then relate the trained monophones and trained biphones according to Equations (5.16) and (5.21), in order to learn the interpolation coefficients that are unknown at this stage. Equations (5.18) and (5.22) are then used to derive the optimum coefficients. A diagram which summarizes this process

is given in Figure 9. The implementation of the algorithm used to learn the coefficients is written in Matlab and is given in the Appendix.

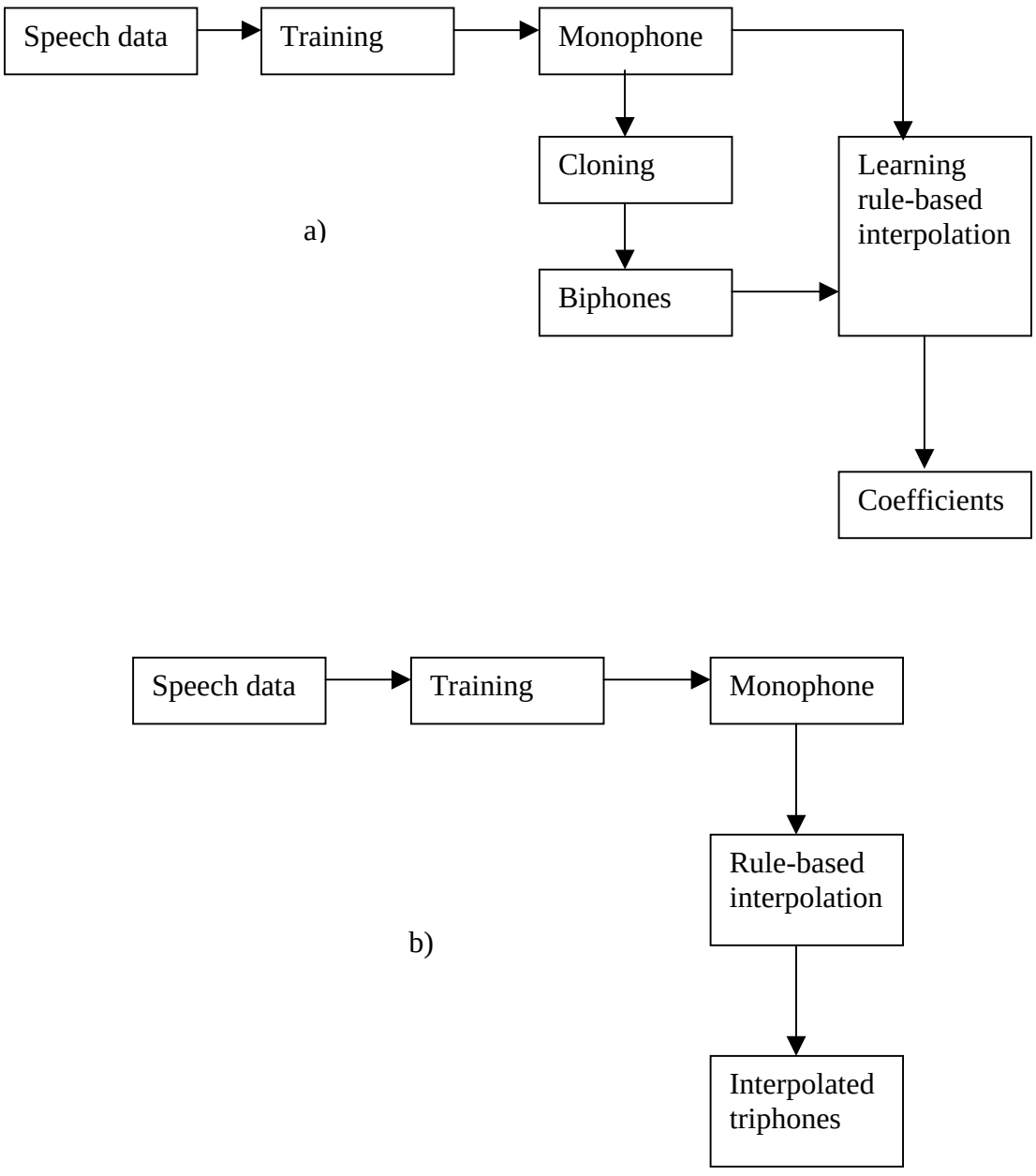

Figure 9: New design method –a) learning –b) triphone creation

6.4.2. Biphones and Triphones Creation

The biphones for the recognition system are generated by combining the means of the monophones according to Equations (5.16) and (5.21), where at this stage, the interpolation coefficients are known and the interpolated biphones parameters are the unknowns to be computed. The process is shown in Figure 9 a) above.

The interpolated triphones are created by concatenating the first and the second states of the left biphones with the third state of the right biphones.

6.4.3. Results

For this method, the performance is the same for interpolated triphones and monophones in the RM task, but in the TIMIT task, the interpolated triphones shows a little better performance than monophones. The performance of the trained triphones remain superior to those of the other models in both RM and TIMIT sentence recognition tasks. This result is also expected especially the fact that we can notice some improvements. Table 4 contains the experimental results for RM and Table 5 contains the experimental results for TIMIT.

% Word corr.: represent the percentage of correctly recognized words.

% Sent. corr.: represent the percentage of correctly recognized sentences.

% Word acc.: represent the difference between the percentage of correctly recognized words and the percentage of insertions.

Table 4: Experimental results for method 2 (4 coefficients) with RM

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	14.00	62.59	61.97
Trained triphone	19.67	66.65	66.61
Interpolated triphone	14.00	62.01	61.30

Table 5: Experimental results for method 2 (4 coefficients) with TIMIT

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	16.13	41.25	41.25
Trained triphone	25.81	54.86	54.86
Interpolated triphone	17.50	42.30	42.30

6.5. Design Method 3 Results

6.5.1. Interpolation Coefficients Learning

We follow the same steps as in design method 2 to learn the rules. We also use the RM database in this task. We first relate mean vectors of monophone and biphones of the same phoneme, and then derive the optimum coefficients. A diagram which summaries the process is given in Figure 9 a).

6.5.2. Biphones and Triphones Creation

The biphones for the recognition system are generated by combining the means of the monophones as in design method 2. The process is shown in Figure 9 b) above. The

interpolated triphones are created by concatenating the first and the second states of the left biphones with the third state of the right biphones.

6.5.3. Results

For this particular method, we clearly see that the interpolated triphones achieve better performance than monophones in both RM and TIMIT recognition tasks. We notice here again that the trained triphones achieve the best recognition performance, with the difference that the interpolated triphones performance is halfway between that of the monophones and the trained triphones, particularily for the TIMIT task. Table 6 contains the experimental results for RM and Table 7 contains the experimental results for TIMIT.

Table 6: Experimental results for method 3 (52 coefficients) with RM

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	14.00	62.59	61.97
Trained triphone	19.67	66.65	66.61
Interpolated triphone	15.93	64.83	63.33

Table 7: Experimental results for method 3 (52 coefficients) with TIMIT

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	16.13	41.25	41.25
Trained triphone	25.81	54.86	54.86
Interpolated triphone	22.58	46.30	46.30

6.6. Design Method 4 Results

6.6.1. Interpolation Coefficients Learning

We follow the same steps as in design method 2 to learn the rules. We also use the RM database in this task. We first relate mean vectors of monophone and biphones of the same phoneme, and then derive the optimum coefficients. A diagram which summaries the process is given in Figure 9 a).

6.6.2. Biphones and Triphones Creation

The biphones for the recognition system are generated by combining the means of the monophones as in design method 2. The process is shown in Figure 9 b) above. The interpolated triphones are created by concatenating the first and the second states of the left biphones with the third state of the right biphones.

6.4.3. Results

Contrary to what happens for design methods 2 and 3, design method 4 did not meet the performance goal that was expected from it. The results show that the improvement is very small compared to that of monophones. This is probably due to the large number of coefficients required. The data available to learn the rule-based interpolation coefficients is insufficient for this. Table 8 contains the experimental results for RM and Table 9 contains the experimental results for TIMIT.

Table 8: Experimental results for method 4 (676 coefficients) with RM

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	14.00	62.59	61.97
Trained triphone	19.67	66.65	66.61
Interpolated triphone	14.33	62.83	62.12

Table 9: Experimental results for method 4 (676 coefficients) with TIMIT

	% Sent. corr.	% Word corr.	% Word acc.
Monophone	16.13	41.25	41.25
Trained triphone	25.81	54.86	54.86
Interpolated triphone	15.12	39.65	39.65

Chapter 7 Conclusion

In This research we have given ourselves a complex and challenging goal which is to incorporate co-articulation into phoneme models. We present a model for co-articulation that is intended to be used to generate triphones directly from monophones. We use four different design approaches with various levels of complexity in terms of the number of interpolation coefficients used. Experiments conducted with RM and TIMIT speech corpuses show interesting results, especially for design method 3. The performance of method 3 could be explained by the moderate number of coefficients it uses, a simplicity which also facilitates its ease to generalize over different speech corpuses. Design method 4 in contrast, shows little improvement with regard to its large number of coefficients. Design method 3 is the most successfull of all four methods.

In general, the approach that we use shows a significant improvement compared to monophones but does not outperform trained triphones. These results show our model has learned something of how monophones combine to create triphones. Of course the method we design cannot represent this knowledge as well as trained triphones but the improvement over monophone clearly indicates its success. One possible explanation is that the specific interpolation model of co-articulation we use may not be able to fully model the phenomenon of co-articulation. Another reason is that we still do not have a complete understanding of how co-articulation works at the physiological level, and did not fully incorporate our understanding into our interpolation structure.

In the future, we intend to focus on searching for a better way to model co-articulation and to successfully (from the performance point of view relatively to trained triphones) use this model to build triphones directly from monophones. In addition, we

will classify the phonemes into classes on the basis of how similar they behave when combines to form triphones. This will allow us to define class-based interpolation method to improve recognition performance.

References

[1] Laurence Rabiner, *Tutorial on HMM and selected applications in speech recognition*, Proc. of IEEE 1989.

[2] Paul Van Alphen, *HMM-based continuous speech recognition*, PTT Research, Leidschendan, The Netherlands. PhD thesis, University of Amsterdam, 1992

[3] Rabiner L. R., *On creating reference templates for speaker independent recognition of isolated words*, IEEE Trans. on Ac. Sp. and Sig. Proc., vol. 26, pp. 34-42, Feb. 1978

[4] Bridle J. S., *Connected word recognition using whole word templates*, Proc. Of the Institute for Acoustics, Autumn conference, pp. 25-28 Nov. 1979.

[5] Myers et al., *Performance tradeoffs in dynamic time warping algorithms for isolated word recognition*, IEEE Trans. on Ac. Sp. and Sig. Proc. vol. 28, pp. 622-635, Dec 1980

[6] Yuxin Zhao et al., *An HMM based speaker-independent continuous speech recognition system with experiments on the TIMIT database*, ICASSP 1991.

[7] Hinton G. E. et al., *Boltzman Machines: Constraint satisfaction networks that learn*, Technical report, vol. CMU-CS –1984-119

[8] Morgan N. et al., *Continuous speech recognition using PLP analysis with multilayer perceptrons*, Proc. of ICASSP 91, pp. 49-52

[9] Piero Cosi, *Hybrid HMM-NN architectures for connected digit recognition*, IEEE-INNS-ENNS International Joint Conference on Neural Networks, vol. 5, Jul. 2000

[10] S. J. Young, *The general use of tying in phoneme-based HMM speech recognisers*, Proc. of ICASSP vol. 3 pp. 569-572, 1992

[11] M. Y. Hwang, *Shared-distribution hidden Markov models for speech recognition*, IEEE Trans. on Speech and Audio Processing, pp. 414-420, 1993.

[12] J. R. Bellegarda et al., *Tied mixtures continuous parameters modeling for speech recognition*, IEEE Trans. on Ac. Sp. and Sig. Proc. pp. 2033-2045, 1990

[13] M. Y. Hwang et al., *Predicting unseen triphones with senones*, Proc. of ICASSP pp. 311-314, Minneapolis, 1993.

[14] S. J. Young et al., *Tree-based state tying for high accuracy acoustic modeling*, ARPA Workshop on Human Language Technology, pp. 286-291, Plainsboro, New Jersey 1994.

[15] P. B. Denes et al., *The Speech Chain: The physics and biology of spoken language*, Wavery Press, Baltimore, 1963.

[16] Proakis et al., *Discrete-time processing of speech signals*, Macmillan 1993.

[17] Randy Goldberg et al., A practical handbook of speech coders, CRC Press LLC 2000

[18] S. B. Davis et al., *Comparison of parametric representations of monosyllabic word recognition in continuously spoken sentences*, IEEE Trans. Ac. Sp. and Sig. Proc., vol. 28 pp. 357-366, 1980.

[19] Athanasios Papoulis, *Probability random variable and stochastic processes*, McGraw Hill 1991

[20] L. Rabiner, *An introduction to hidden Markov models*, IEEE Ac. Sp. and Sig. Proc. Mag. 1986.

[21] G. D. Forney, *The Viterbi algorithm*, Proc of. IEEE 1973.

[22] L. E. Baum, *An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes*, Inequalities 1972

[23] A. L. Dempster et al., *Maximum likelihood for imcomplete data via the EM algorithm*, Journal of. Roy. Sta. Soc.1977.

[24] Schoten M. E. H. et al, *Vowels segments in consonantal contexts: a spectral study of co-articulation* – part 1, Journal of Phonetics, vol. 7, 1-23, 1979.

[25] Douglas O'Shaugnessy, *Speech Communication: Human and machine*, Addison Wesley, 1987.

[26] H. Hohne et al., *On temporal alignment of sentences of natural and synthetic speech*, IEEE Trans. on Ac. Sp. and Sig. Proc.31, pp. 807-813, 1983.

[27] H. Leung et al., *A procedure for automatic alignment of phonetic transcriptions in continuous speech*, Proc. of ICASSP, 1984.

[28] M. Hunt, *Time alignment of natural speech to synthetic speech*, Proc. of ICASSP, 1984.

[29] C. Meyers et al., *Performance tradeoffs in dynamic time warping algorithms for isolated word recognition*, IEEE Trans. on Ac. Sp. and Sig. Proc. vol. 28, pp. 622-635, 1980.

[30] B. Yegnanarayana et al., *Signal-dependent matching for isolated word speech recognition systems*, Signal Processing 7, pp. 161-173, 1984.

[31] L. Rabiner et al., *Isolated and connected word recognition theory and selected applications*, IEEE Trans. Comm. COM 29, pp. 621-659, 1981.

[32] S. Furui, *Speaker independent isolated word recognition using dynamic features of spectrum*, IEEE Trans. on Ac. Sp. and Sig. Proc.vol. 34, no. 1, pp.53-59, 1986.

[33] Steve Young et al., *HTK Handbook 3.0*, Microsoft Corporation 1995

[34] Li Deng, *A generalized hidden Markov model with state-conditioned trend functions of time for speech signa*l, Signal Proc. 27, pp. 65-78, Elsevier, 1992.

[35] Li Deng, *A stochastic model of speech incorporating hierarchical nonstationarity*, Proc. of ICASSP, 1993

[36] Li Deng et al., *Speech recognition using hidden Markov model with polynomial regression function as nonstationary states*, IEEE Trans. on Speech and Audio processing, vol.2, no.4, October, 1994.

[37] Y. Gong et al., *Stochastic trajectory modeling for speech recognition*, Proc. of ICASSP, 1994.

[38] Y. Gong et al., *Modeling long term variability information in mixture stochastic trajectory framework*, Proc of Int. Conf. Of Spoken Language, 1996

[39] O. Siohan, Y. Gong, *A semi-continuous stochastic trajectory model for phoneme-based continuous speech recognition*, Proc. of ICASSP.

[40] Y. Gong et al., *Elimination of trajectory folding phenomenon: HMM, trajectory mixture HMM and mixture stochastic trajectory model*, Proc. of ICASSP, 1997.

[41] I. Illina, Y. Gong, *Stochastic trajectory model with state-mixture for continuous speech recognition*, Proc. of ICSLP, pp.338-341, 1996

[42] Y. Gong, *Stochastic trajectory modeling for continuous speech recognition*, IEEE Trans. on Sp. and Aud. Proc., vol. 5, no.1, January 1997.

[43] M. B. Priestley, *Non-linear time series analysis*, Academic Press, pp. 140-174, London, 1988.

[44] P. Price et al, *The DARPA 1000-word resource management database for continuous speech recognition*, Proc. of ICASSP, 1988.

[45] Garofolo J. et al, *TIMIT acoustic-phonetic continuous speech corpus*, Linguistic Data Consortium, 1993

Appendix: Program Code

```
%Main
list = 'mono.list';
fid_l = fopen(list,'r'); % open the file that contains the list of all phonemes
fid = fopen ('lbicoef_lsqlin_f','w');
%lb=-2*ones(1,26);
%ub=2*ones(1,26);

total_error=0;
while feof (fid_l) == 0
     phone = fgetl (fid_l);
     phone = sscanf (phone,'%s')
     [C,d]=get_coef4(phone);
     x0=[zeros(13*13,1),ones(13*13,1)];
     [x,fval]=lsqlin(C,d,[],[],[],[],[],[],x0);
     fprintf(fid,[num2str(phone),'\n']);
     %fprintf(fid,'x\n');
     fprintf(fid,' %e',x');
     fprintf(fid,'\n');
     %fprintf(fid,'fval\n');
     %fprintf(fid,' %e',fval);
     %fprintf(fid,'\n');
     total_error=total_error +fval;
  end %end outer while
fprintf(fid,'TOTAL_ERROR ');
fprintf(fid,'%e\n',total_error);
fclose(fid_l);
fclose(fid);
```

```
function [C,d] = get_coef4(phone)
```

```
rep=strcat('-',phone);
MODELS = 'MODELS_lbi'; %lbi_models
MODELS_m = 'MODELS_mono'; %monophone models
%list = 'mono.list';
%fid_l = fopen(list,'r'); % open the file that contains the list of all phonemes



%***********load all the monophones into structure mono(num)************
fid_mono = fopen (MODELS_m,'r');  %open the file containing the monophone models
for i=1:6
      fgetl(fid_mono);
end
num=0;
name='';
while feof (fid_mono) == 0
temp = fgetl (fid_mono);
name=strcat(name,temp); % temp
temp = strrep(temp,'~h "','');
temp = strrep (temp,'"','');
if strcmpi(temp,'sp')
   for i=1:13
      temp = fgetl(fid_mono);
   end
   if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end first\n');
      temp
      break;
   end
else
   num=num+1;
   mono(num).name = temp;
   for i=1:4
      fgetl (fid_mono);
   end
   temp = fgetl (fid_mono);
   mono(num).state2.mean = sscanf (temp,'%g');
   for i=1:5
      fgetl (fid_mono);
   end
   temp = fgetl (fid_mono);
   mono(num).state3.mean = sscanf (temp,'%g');
   for i=1:5
      fgetl (fid_mono);
   end
   temp = fgetl (fid_mono);
```

```
   mono(num).state4.mean = sscanf (temp,'%g');
   for i=1:10
      temp = fgetl (fid_mono);
   end
   if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end second\n');
      temp
      break;
   end
   end  % end if strcmpi(temp,sp)

end %end while

%*************************************************************************


%------------------READ THE MODELS MEANS INTO STRUCTURES--------------------
  %initial index - index of the first hmm enconter in the model file
  %while feof (fid_l) == 0
       phone_index=0;
      %phone = fgetl (fid_l);
      %phone = sscanf (phone,'%s');
      for k=1:num % get index of the target phone from the monophones list mono(k).
        if strcmpi(phone,mono(k).name) phone_index=k;
        end
      end
      fid = fopen (MODELS,'r');  %open the file containing the l_biphone models
      for i=1:6
        fgetl(fid);
      end
 count=0;
 name='';
 while feof (fid) == 0
temp = fgetl (fid);
name=strcat(name,temp); % temp
temp = strrep(temp,'~h "','');
temp = strrep (temp,'"','');
if strcmpi(temp,'sp')
   for i=1:10
      temp = fgetl(fid);
   end
   if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end first\n');
      temp
      break;
   end
```

```
elseif (findstr(temp,rep))
   [token,rem] = strtok(temp,'-');
   if strcmpi(rem,rep)
   count=count+1;
   hmm(count).name = temp;
   for i=1:4
      fgetl (fid);
   end
   temp = fgetl (fid);
   hmm(count).state2.mean = sscanf (temp,'%g');
   for i=1:5
      fgetl (fid);
   end
   temp = fgetl (fid);
   hmm(count).state3.mean = sscanf (temp,'%g');
   for i=1:5
      fgetl (fid);
   end
   temp = fgetl (fid);
   hmm(count).state4.mean = sscanf (temp,'%g');
   for i=1:5
      temp = fgetl (fid);
   end
   if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end second\n');
      temp
      break;
   end
else % else strcmpi(rem,rep)
   for i=1:22
      temp = fgetl(fid);
   end
  if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end third\n');
      temp
      break;
   end
end %end strcmpi(rem,rep)

else
   for i=1:22
      temp = fgetl(fid);
   end
  if strcmpi(temp,'<ENDHMM>')==0
      fprintf('the corresponding hmm was not read until its end third\n');
```

```
      temp
      break;
   end
end  % end if strcmpi(temp,sp)

end %end inner while

%code goes here
fclose(fid);

%end %end outer while
%fclose(fid_l);


%********************f(x)for phone**********************************
   temp=hmm(1).name;
   temp = strrep (temp,rep,'');
   cur_index=0;
   % get the index of the monophone which is the left context of hmm(i) cur_index
   for k=1:num
      if strcmpi(temp,mono(k).name) cur_index=k;
      end
   end
   A1=[diag(mono(cur_index).state4.mean(1)*ones(1,13))];
   A2=[diag(mono(cur_index).state4.mean(2)*ones(1,13))];
   A3=[diag(mono(cur_index).state4.mean(3)*ones(1,13))];
   A4=[diag(mono(cur_index).state4.mean(4)*ones(1,13))];
   A5=[diag(mono(cur_index).state4.mean(5)*ones(1,13))];
   A6=[diag(mono(cur_index).state4.mean(6)*ones(1,13))];
   A7=[diag(mono(cur_index).state4.mean(7)*ones(1,13))];
   A8=[diag(mono(cur_index).state4.mean(8)*ones(1,13))];
   A9=[diag(mono(cur_index).state4.mean(9)*ones(1,13))];
   A10=[diag(mono(cur_index).state4.mean(10)*ones(1,13))];
   A11=[diag(mono(cur_index).state4.mean(11)*ones(1,13))];
   A12=[diag(mono(cur_index).state4.mean(12)*ones(1,13))];
   A13=[diag(mono(cur_index).state4.mean(13)*ones(1,13))];
   B1=[diag(mono(phone_index).state2.mean(1)*ones(1,13))];
   B2=[diag(mono(phone_index).state2.mean(2)*ones(1,13))];
   B3=[diag(mono(phone_index).state2.mean(3)*ones(1,13))];
   B4=[diag(mono(phone_index).state2.mean(4)*ones(1,13))];
   B5=[diag(mono(phone_index).state2.mean(5)*ones(1,13))];
   B6=[diag(mono(phone_index).state2.mean(6)*ones(1,13))];
   B7=[diag(mono(phone_index).state2.mean(7)*ones(1,13))];
   B8=[diag(mono(phone_index).state2.mean(8)*ones(1,13))];
   B9=[diag(mono(phone_index).state2.mean(9)*ones(1,13))];
   B10=[diag(mono(phone_index).state2.mean(10)*ones(1,13))];
```

```
   B11=[diag(mono(phone_index).state2.mean(11)*ones(1,13))];
   B12=[diag(mono(phone_index).state2.mean(12)*ones(1,13))];
   B13=[diag(mono(phone_index).state2.mean(13)*ones(1,13))];


   d=(hmm(1).state2.mean)';
   for i=2:count % i=1 is define above
   temp=hmm(i).name;
   temp = strrep (temp,rep,'');
   cur_index=0;
   % get the index of the monophone which is the left context of hmm(i) cur_index
   for k=1:num
      if strcmpi(temp,mono(k).name) cur_index=k;
      end
   end
   A1=[A1,diag(mono(cur_index).state4.mean(1)*ones(1,13))];
   A2=[A2,diag(mono(cur_index).state4.mean(2)*ones(1,13))];
   A3=[A3,diag(mono(cur_index).state4.mean(3)*ones(1,13))];
   A4=[A4,diag(mono(cur_index).state4.mean(4)*ones(1,13))];
   A5=[A5,diag(mono(cur_index).state4.mean(5)*ones(1,13))];
   A6=[A6,diag(mono(cur_index).state4.mean(6)*ones(1,13))];
   A7=[A7,diag(mono(cur_index).state4.mean(7)*ones(1,13))];
   A8=[A8,diag(mono(cur_index).state4.mean(8)*ones(1,13))];
   A9=[A9,diag(mono(cur_index).state4.mean(9)*ones(1,13))];
   A10=[A10,diag(mono(cur_index).state4.mean(10)*ones(1,13))];
   A11=[A11,diag(mono(cur_index).state4.mean(11)*ones(1,13))];
   A12=[A12,diag(mono(cur_index).state4.mean(12)*ones(1,13))];
   A13=[A13,diag(mono(cur_index).state4.mean(13)*ones(1,13))];
   B1=[B1,diag(mono(phone_index).state2.mean(1)*ones(1,13))];
   B2=[B2,diag(mono(phone_index).state2.mean(2)*ones(1,13))];
   B3=[B3,diag(mono(phone_index).state2.mean(3)*ones(1,13))];
   B4=[B4,diag(mono(phone_index).state2.mean(4)*ones(1,13))];
   B5=[B5,diag(mono(phone_index).state2.mean(5)*ones(1,13))];
   B6=[B6,diag(mono(phone_index).state2.mean(6)*ones(1,13))];
   B7=[B7,diag(mono(phone_index).state2.mean(7)*ones(1,13))];
   B8=[B8,diag(mono(phone_index).state2.mean(8)*ones(1,13))];
   B9=[B9,diag(mono(phone_index).state2.mean(9)*ones(1,13))];
   B10=[B10,diag(mono(phone_index).state2.mean(10)*ones(1,13))];
   B11=[B11,diag(mono(phone_index).state2.mean(11)*ones(1,13))];
   B12=[B12,diag(mono(phone_index).state2.mean(12)*ones(1,13))];
   B13=[B13,diag(mono(phone_index).state2.mean(13)*ones(1,13))];

   d=[d,(hmm(i).state2.mean)'];
end
A1=(A1)';
A2=(A2)';
```

```
A3=(A3)';
A4=(A4)';
A5=(A5)';
A6=(A6)';
A7=(A7)';
A8=(A8)';
A9=(A9)';
A10=(A11)';
A11=(A11)';
A12=(A12)';
A13=(A13)';
B1=(B1)';
B2=(B2)';
B3=(B3)';
B4=(B4)';
B5=(B5)';
B6=(B6)';
B7=(B7)';
B8=(B8)';
B9=(B9)';
B10=(B11)';
B11=(B11)';
B12=(B12)';
B13=(B13)';

C=[A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,B1,B2,B3,B4,B5,B6,B7,B8,B9,
B10,B11,B12,B13];
d=d';
```

```
%This is a matlab interface to read models
```

```
function prog(MODELS)
fid = fopen (MODELS,'r');  %open the file containing the monophone models
%--------GET THE FEATURE VECTOR DIMENSION IN 'vecSize'-------------------------
fgetl (fid);
data1 = fgetl (fid);
remainder = data1;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
   vecSize = sscanf(chopped,'%d');
%-----------------------------------------------------------------------------------
%----------GET THE NUMBER OF MIXTURES OF THE MODELS--------
for i=1:8
fgetl (fid);
end
data1 = fgetl (fid);
fclose(fid);   %close the models file
remainder = data1;
   [chopped,remainder] = strtok(remainder);
   isMixture = chopped;
   [chopped,remainder] = strtok(remainder);
   numMix = sscanf(chopped,'%d');
   if strcmp(isMixture,'<NUMMIXES>')
   % code for multiple mixtures
    readModel2(MODELS,vecSize,numMix);
   elseif strcmp (isMixture,'<MEAN>')
   % code for single mixture
     readModel(MODELS,vecSize);
  else
      fprintf('error in the model file \n');
   end
%-----------------------------------------------------------------------------------------
```

```
function readModel(MODELS,vecSize)
```

```
%-------------------------------------------------------------------------
fid_c = fopen ('lbicoef_lsqlin','r');
count=0;
while feof (fid_c) == 0
   count=count+1;
temp = fgetl (fid_c);
coef(count).name=temp;
temp = fgetl (fid_c);
coef(count).value = sscanf (temp,'%g');
end
fclose(fid_c);
%-------------------------------------------------------------------------
fid = fopen (MODELS,'r'); % open the file for read by the next two modules
%---------------------THIS STRUCTURE CONTAINS THE HEADER INFORMATION
OF THE MODELS----------
%---------------------REQUIRES VARIABLE 'vecSize'---------------------------------------

HInfo.info1 = strcat(fgetl (fid),'\n');
HInfo.info2 = strcat(fgetl (fid),'\n');
HInfo.info3 = strcat(fgetl (fid),'\n');
HInfo.info4 = strcat(fgetl (fid),'\n');
HInfo.info5 = strcat(fgetl (fid),'\n');
HInfo.info6 = strcat(fgetl (fid),'\n');
%-----------------------------------------------------------------------------------------


%------------------READ THE MODELS INTO STRUCTURES-------------------------------
k=0;  %initial index - index of the first hmm enconter in the model file
 while feof (fid) == 0
 temp = fgetl (fid);
temp = strrep(temp,'~h "','');
temp = strrep (temp,'"','');
for i=1:4
fgetl (fid);
end
if (strcmp(temp,'sil')|strcmp(temp,'sp'))
if strcmp(temp,'sil')  % defines a hmm structure for 'sil'
       % code goes here
      sil.name = temp;
temp = fgetl (fid);
sil.state2.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
sil.state2.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
```

```
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
sil.state2.gconst = sscanf(chopped,'%g');
fgetl (fid);
fgetl (fid);

temp = fgetl (fid);
sil.state3.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
sil.state3.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
sil.state3.gconst = sscanf(chopped,'%g');
fgetl (fid);
fgetl (fid);

temp = fgetl (fid);
sil.state4.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
sil.state4.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
sil.state4.gconst = sscanf(chopped,'%g');
fgetl (fid);

sil.transPmacro = '~t "trP_sil"' ;% TEMP
for i=1:5
temp = fgetl (fid);
sil.transP(i,:) = sscanf(temp,'%g');
end
fgetl (fid);
else   % defines a hmm structures for 'sp'
       % code goes here
       sp.name = temp;
temp = fgetl (fid);
sp.state2.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
sp.state2.variance = sscanf (temp,'%g');
temp = fgetl (fid);
```

```
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
sp.state2.gconst = sscanf(chopped,'%g');
fgetl (fid);

for i=1:3
temp = fgetl (fid);
sp.transP(i,:) = sscanf(temp,'%g');
end
fgetl (fid);
end  %end nested if

else % defines all hmm that is different from 'sil' and 'sp'
   k=k+1;  % index of the current hmm structure
hmm(k).name = temp;
temp = fgetl (fid);
hmm(k).state2.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
hmm(k).state2.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
hmm(k).state2.gconst = sscanf(chopped,'%g');
fgetl (fid);
fgetl (fid);

temp = fgetl (fid);
hmm(k).state3.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
hmm(k).state3.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
hmm(k).state3.gconst = sscanf(chopped,'%g');
fgetl (fid);
fgetl (fid);

temp = fgetl (fid);
hmm(k).state4.mean = sscanf (temp,'%g');
fgetl (fid);
temp = fgetl (fid);
```

```
hmm(k).state4.variance = sscanf (temp,'%g');
temp = fgetl (fid);
remainder = temp;
   [chopped,remainder] = strtok(remainder);
   [chopped,remainder] = strtok(remainder);
hmm(k).state4.gconst = sscanf(chopped,'%g');
fgetl (fid);

hmm(k).transPmacro = ['~t "trP_',num2str(k),""] ;% TEMP
for i=1:5
temp = fgetl (fid);
hmm(k).transP(i,:) = sscanf(temp,'%g');
end
fgetl (fid);
end  %end if
end  %end while
%temp=''; get rid of this variable
fclose(fid);
%-------------------------------END READ THE MODELS INTO STRUCTURES---------


%----------------------------------    WRITE THE TRIPHONE MODELS IN A MASTER
MODELS FILE ----------------------------------------
%code goes here
fid2 = fopen ('tri_MODELS','w');
fid3 = fopen ('tri_list','w');
%-----------------------------------------------------------------------------------------------------


%-----------------------------------------WRITE THE HEADER INFORMATION OF THE
%----------TRIPHONE MODELS-------------------------------------
fprintf (fid2,HInfo.info1);
fprintf (fid2,HInfo.info2);
fprintf (fid2,HInfo.info3);
fprintf (fid2,HInfo.info4);
fprintf (fid2,HInfo.info5);
fprintf (fid2,HInfo.info6);
%-----------------------------------------------------------------------------------------------------

%-------------------WRITE TRANSITION PROBABILITIES MACRO FOR 'sil'----------
fprintf(fid2,[num2str(sil.transPmacro),'\n']);
fprintf(fid2,'<TRANSP> 5\n');
fprintf(fid2,' %e %e %e %e %e\n',sil.transP(1,:));
fprintf(fid2,' %e %e %e %e %e\n',sil.transP(2,:));
fprintf(fid2,' %e %e %e %e %e\n',sil.transP(3,:));
```

```
fprintf(fid2,' %e %e %e %e %e\n',sil.transP(4,:));
fprintf(fid2,' %e %e %e %e %e\n',sil.transP(5,:));
%-------------------------------------------------------------------------------------

%------------------------------WRITE TRANSITION PROBABILITIES MACRO----------
for i=1:k
fprintf(fid2,[num2str(hmm(i).transPmacro),'\n']);
fprintf(fid2,'<TRANSP> 5\n');
fprintf(fid2,' %e %e %e %e %e\n',hmm(i).transP(1,:));
fprintf(fid2,' %e %e %e %e %e\n',hmm(i).transP(2,:));
fprintf(fid2,' %e %e %e %e %e\n',hmm(i).transP(3,:));
fprintf(fid2,' %e %e %e %e %e\n',hmm(i).transP(4,:));
fprintf(fid2,' %e %e %e %e %e\n',hmm(i).transP(5,:));
end


%-------------------------------------------------------------------------------------------------



%-----------------------------------------------WRITE 'sp' MODELS TO FILE -------------------
fprintf(fid3,'sp \n');
fprintf(fid2,'~h "sp"\n');
fprintf(fid2,'<BEGINHMM>\n');
fprintf(fid2,'<NUMSTATES> 3\n');
fprintf(fid2,'<STATE> 2\n');
fprintf(fid2,['<MEAN> ',num2str(vecSize),'\n']);
fprintf(fid2,' %e',sp.state2.mean);
fprintf(fid2,'\n');
fprintf(fid2,['<VARIANCE> ',num2str(vecSize),'\n']);
fprintf(fid2,' %e',sp.state2.variance);
fprintf(fid2,'\n');
fprintf(fid2,'<GCONST> ');
fprintf(fid2,'%e\n',sp.state2.gconst);
fprintf(fid2,'<TRANSP> 3\n');
fprintf(fid2,' %e %e %e\n',sp.transP(1,:));
fprintf(fid2,' %e %e %e\n',sp.transP(2,:));
fprintf(fid2,' %e %e %e\n',sp.transP(3,:));
fprintf(fid2,'<ENDHMM>\n');

%-------------------------------------------------------------------------------------------------------



%-----------------------------------------------WRITE 'sil' MODELS TO FILE -------------------
write2file(sil,fid2,fid3,vecSize);
```

```
%code goes here
%-------------------------------------------------------------------------------------------------

%-----------------------------------------------WRITE monophones MODELS TO FILE --------
for c=1:k
write2file(hmm(c),fid2,fid3,vecSize);
end
%code goes here
%-------------------------------------------------------------------------------------------------


%-------THIS STRUCTURE IS THE TEMPORARY DESTINATION OF  THE
%TRIPHONES - WRITE TRIPHONES TO OUTPUT FILE--------------------------
t=k+1;
for c=1:k %INDEX OF CENTER HMM
   temp = strrep(hmm(c).name,'~h "',");
   temp = strrep (temp,"",");
   cur_index=0;
   for i=1:count
      if strcmpi(temp,coef(i).name);
         cur_index=i;
      end
   end
   alpha=coef(cur_index).value(1:13);
   beta=coef(cur_index).value(14:26);

   %if strcmpi(temp,'ch');
   %alpha=zeros(13,1);
   %beta=ones(13,1);
   %end

   %if strcmpi(temp,'oy');
   %alpha=zeros(13,1);
   %beta=ones(13,1);
   %end

    %INDEX OF HMM THAT SHOULD MAKE THE RIGHT AND LEFT CONTEXT
OF THE CENTER HMM
   for l=1:k %INDEX LEFT

         % buildTriphone(hmm(E(j)),hmm(i),hmm(E(k));
         hmm(t).name=[hmm(l).name,'-',hmm(c).name];

         hmm(t).state2.mean = (hmm(c).state2.mean).*beta + (hmm(l).state4.mean).*alpha;
         hmm(t).state2.variance = hmm(c).state2.variance;
         hmm(t).state2.gconst = hmm(c).state2.gconst;
```

```
        hmm(t).state3.mean = hmm(c).state3.mean;
        hmm(t).state3.variance = hmm(c).state3.variance;
        hmm(t).state3.gconst = hmm(c).state3.gconst;

        hmm(t).state4.mean = hmm(c).state4.mean ;
        hmm(t).state4.variance = hmm(c).state4.variance;
        hmm(t).state4.gconst = hmm(c).state4.gconst;

        hmm(t).transPmacro = hmm(c).transPmacro;
        hmm(t).transP=hmm(c).transP;

        %fprintf(['writing model ',num2str(l),'-',num2str(c),'+',num2str(r),'\n']);
        write2file(hmm(t),fid2,fid3,vecSize);

    end
    fprintf(['writing model ',num2str(c+1),'\n']);
end

%hmm(k).name = 'triphone';  temp*
%------CODE FOR BUILDING TRIPHONES GOES HERE
%
%
%-----write models to file
% for i=1:44  %temp
% write2file(hmm(i),fid2,vecSize);
% end  %temp
%-------------------------------------------------------------------------------------------------------



 fclose(fid2);
 fclose(fid3);
%-----------------------------------------------------------END-----------------------------------------



function write2file(model,file,file2,vecSize)
```

```
%This function write the constructed triphone to the output model file
fprintf(file2,[num2str(model.name),'\n']);
fprintf(file,['~h "',num2str(model.name),'"\n']);
fprintf(file,'<BEGINHMM>\n');
fprintf(file,'<NUMSTATES> 5\n');
fprintf(file,'<STATE> 2\n');
fprintf(file,['<MEAN> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state2.mean);
fprintf(file,'\n');
fprintf(file,['<VARIANCE> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state2.variance);
fprintf(file,'\n');
fprintf(file,'<GCONST> ');
fprintf(file,'%e\n',model.state2.gconst);

fprintf(file,'<STATE> 3\n');
fprintf(file,['<MEAN> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state3.mean);
fprintf(file,'\n');
fprintf(file,['<VARIANCE> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state3.variance);
fprintf(file,'\n');
fprintf(file,'<GCONST> ');
fprintf(file,'%e\n',model.state3.gconst);

fprintf(file,'<STATE> 4\n');
fprintf(file,['<MEAN> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state4.mean);
fprintf(file,'\n');
fprintf(file,['<VARIANCE> ',num2str(vecSize),'\n']);
fprintf(file,' %e',model.state4.variance);
fprintf(file,'\n');
fprintf(file,'<GCONST> ');
fprintf(file,'%e\n',model.state4.gconst);


fprintf(file,[num2str(model.transPmacro),'\n']);
fprintf(file,'<ENDHMM>\n');
```