Speech Recognition & Diphone Extraction for Natural Speech Synthesis

By

Hossein B. Darbandi

B.Sc. In Computer science, September 1999

BA.Sc. Electrical Engineering, January 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

In

THE FACULTY OF GRADUATE STUDIES

(Department of Electrical and Computer Engineering)

We accept this thesis as conforming

To the required standard

The University of British Columbia

February 2002

In presenting this thesis in partial fulfillment of the requirements for an advances degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purpose may be granted by the head of my department or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical and Computer Engineering
The University of British Columbia ,
Vancouver, Canada.

April 18, 2002

Abstract

Modern speech synthesizers use concatenated words and sub-word segments, such as diphones, to synthesize natural speech. Synthesizers available today can synthesize speech with only a limited selection of voices provided by the vendors. The voice segments (e.g. words & diphones) are often created using semi-manual processes that are prone to human error and make the segments non-uniform.

The main goal of this thesis is developing an automatic method to segment and label a natural speech into words, diphones, and phonemes. To segment speech into words and sub-words, I use a speech recognition engine. The commercially available speech recognition engines do not provide all the necessary functionality to segment the speech into diphones accurately. As a result, I have developed an engine to segment speech. For developing the engine, I have employed HTK tools provided by Cambridge University, available for free.

Table of Contents

ABSTRACT ..... **II**

TABLE OF CONTENTS III
LIST OF FIGURES VI
LIST OF TABLES VII
ACKNOWLEDGEMENTS VIII

CHAPTER 1 ..... **1**

INTRODUCTION 1
1.1 Motivation *1*
1.2 Implementation *3*
1.3 Outline of the Thesis *4*
1.4 Problem of Speech Recognition *5*
1.4 Mathematical Model of Speech Recognition *5*
1.5 Elements of a Speech Recognizer *6*
1.5.1 Acoustic Processing 7
1.5.2 Acoustic Modeling 7
1.5.3 Language Modeling 8
1.5.4 Recognition 9
1.6 Elements of Speech *9*
1.6.1 Phones and Phonemes 9
1.6.2 Phonetic and Phonemic Alphabets 10

CHAPTER 2 ..... **13**

ACOUSTIC PROCESSING 13
2.1 Sampling *13*
2.2 Pre-emphasize *14*
2.3 Hamming Window *15*
2.4 Fast Furrier Transformation (FFT) *16*
2.5 Filter Bank *17*
2.6 Log Energy Computation *18*
2.7 Discrete Cosine Transform *18*

CHAPTER 3 ..... **21**

ACOUSTIC MODEL 21
3.1 Hidden Markov Model *22*
3.2 Model of Phonemes *23*
3.3 Speech Recognition Using HMM *26*
3.4 Baum-Welch Algorithm *27*
3.5 Viterbi Algorithm *28*
3.6 Token Passing Method *29*
3.7 Training of HMM *30*
3.8 Baum-Welch or Forward-Backward Estimation *30*
3.9 Forward-Backward Algorithm *31*

3.10 Adaptation *34*
3.10.1 Model Adaptation Using MLLR 34
3.11 Language Model *36*

CHAPTER 4 ..... **39**

MODELING & TRAINING 39
4.0 Introduction *39*
4.1 Acoustic Processing *40*
4.2 Acoustic Modeling *41*
4.3 Training *43*
4.3.1 Mono-phone training 44
4.3.2 Triphone training 46
4.4 Evaluation Method *47*
4.5 Experiments *48*
4.5.1 Experiment conditions 49
4.5.2 Mono-Phoneme 49
4.5.3 triphone 52
4.5.4 Tied-Model 54
4.6 Comparison *55*

CHAPTER 5 ..... **61**

ADAPTATION 61
5.1 Triphone adaptation *61*
5.2 Tied-model adaptation *64*
5.3 Retraining *65*
5.4 Comparison *67*

CHAPTER 6 ..... **69**

DIPHONE & WORD SEGMENTATION AND EXTRACTION FOR NATURAL SPEECH SYNTHESIS 69
6.1 Diphone Segmentation *69*
6.2 Indirect Method *70*
6.2.1 Evaluation 71
6.2.2 Method I 72
6.2.3 Method II 73
6.2.4 Diphone Segmentation *76*
6.3 Direct Method *77*
6.4 Speech Synthesizer *80*
6.4.1 Program Options 80
6.4.2 Speech Output 1 81
6.4.3 Speech Output 2 82
6.4.4 Comparision 83
6.5 Word Segmentation *84*
6.6 Word Spotting *85*

CHAPTER 7 ..... **88**

CONCLUSIONS 88
7.1 Summary *88*

7.2 Future work *90*

APPENDIXES **92**

APPENDIX A 92
Regression Class Tree *92*
APPENDIX B 94
EM Algorithm *94*
Example of the EM Algorithm *97*
APPENDIX C 100
Tree-Based Clustering *100*
Tied-state triphone *101*
APPENDIX D 103
Question set employed for clustering *103*

REFERENCES **108**

List of Figures

Figure 1.1	Overview of a speech recognition system	7
Figure 2.1	Acoustic Processing	14
Figure 2.2	Mel Filter Bank	17
Figure 2.3	Part of the signal of phoneme "hh"	19
Figure 2.4	MFCC samples	19
Figure 3.1	A three-state Markov chain	21
Figure 3.2	A three-state Hidden Markov Model	22
Figure 3.3	Basic structure of a phonetic HMM	24
Figure 3.4	A modified network recognizes any permutation of Flip and Flop.	36
Figure 3.5	Word Internal Triphone Expansion of Flip-Flop Network	37
Figure 3.6	The modified version of Flip and	38
Figure 4.1	*sil* and *sp* model	43
Figure 4.2	The wave form of "Her hum became a gurgle of surprise."	44
Figure 4.3	The correctness of the mono-phone model recognizing *Train data.*	56
Figure 4.4	The correctness of the mono-phone model recognized *Test data*....	56
Figure 4.5	The correctness of triphone model	57
Figure 4.6	The correctness of the tied-model	58
Figure 4.7a	Comparing the triphone to the tied-triphone model	59
Figure 4.7b	Comparing the Accuracy of the triphone and tied-triphone	59
Figure 4.8	The size of the models in k bytes	60
Figure 5.1	Correctness of the model before and after adaptation	63
Figure 5.2	Correctness of the tied-triphone model before and after adaptation.	65
Figure 5.3	Correctness and accuracy of the untied-triphone model	66
Figure 5.4	Accuracy and correctness of the tied-triphone model	67
Figure 6.1	Block diagram of Diphone segmentation	70
Figure 6.2	Calculating the accuracy of segmentation	71
Figure 6.3	Distribution and accuracy of segmented phoneme samples	72
Figure 6.4	Mapping the model to Normal Distribution	73
Figure 6.5	C correctness of segmenting phonemes	74
Figure 6.6	Distribution and correctness of segmented samples	75
Figure 6.7	Distribution and accuracy of the segmented samples	76
Figure 6.8	Correctness of diphones segmented using Indirect-Method	77
Figure 6.9	Correctness of diphones segmented using Direct-Method	79
Figure 6.10	Wave form of utterance "His head flopped back" uttered by Keith.	82
Figure 6.11	Synthesized utterance "His head flopped back."	82
Figure 6.12	The utterance synthesized with three methods	83
Figure 6.12	Distribution and accuracy of word segmentation	84
Figure 6.13	An instance of Word Spot program	86
Figure 6.14	Another instance of Word Spot program	87
Figure A.1	A binary regression tree	92
Figure C1	Decision tree-bases state tying	100

List of Tables

Table 1.1	The CMU phoneme set used in the thesis.....	12
Table 4.1	Accuracy and correctness of a mono-phoneme model without *sp*	50
Table 4.2	Accuracy and correctness of the mono-phoneme model with *sp*.....	51
Table 4.3	Accuracy and correctness of compound model.....	51
Table 4.4	Accuracy and correctness of the triphone model trained with WET...	53
Table 4.5	Accuracy and correctness of triphone model trained with WIT.	53
Table 4.6	Accuracy and correctness of tied-triphone model trained with WIT...	55
Table 5.1	Accuracy and correctness of the adapted data for the triphone model	62
Table 5.2	Correctness of the adapted tied-triphone model.....	64
Table 5.3	A&C of the triphone model retrained after being adapted.....	65
Table 5.4	A&C of the tied-triphone triphone model retrained after adaptation...	66
Table D1	The set of questions are used for clustering.	103

Acknowledgements

I would like to thank all the people who have helped me in the accomplishment of this work. I appreciate the help of my supervisor, Babak Hamidzadeh, who has supported and guided me throughout my research. I am also grateful to my friends; especially to Ramin and Keith for their contributions to this project. I am also indebted to George Tasikin for his great help and advice throughout my education.

Finally, special thanks to my wife, who has supported me all this time, and to my brother who has always encouraged me to pursue higher education.

Hossein B. Darbandi

The University of British Columbia

February 2002

Chapter 1

Introduction

1.1 Motivation

Modern speech synthesizers generate a synthesized speech by concatenating segments of natural speech. The segments of a speech can be phrases, words, or sub-words that a speech synthesizer uses to create a speech. For concatenating words, a database that has an utterance for each word should be utilized. However, a database that keeps an utterance for every word in a language will be too big and too difficult to be implemented. As a result, synthesizers use sub-word segments in addition to common word segments to synthesize speech.

Words are made up of phonemes, and there are a limited numbers of phonemes in each language. For example American English is made up of about 40 phonemes. As a result, instead of concatenating words, a synthesizer may use sub-words such as phonemes and diphones to create speech. Studies [1] have shown that speech synthesized by diphones, which are a sub-word division that begins from middle of a phoneme and ends in the middle of the adjacent phoneme, provides more natural speech.

Segmenting and labeling diphones has, until now, been a manual and semi-manual process that demands linguistic skills, and it is prone to human error. Besides, it is a difficult process. Consequently, the voices created by speech synthesizers are limited to a few that are provided by the vendors. To synthesize a variety of voices, a set of

speech automatically for any given voice can speed up and facilitate segmentation process.

In my thesis research, I have developed an automatic method to segment and label natural speech into corresponding words and diphones. To segment speech into words and diphones accurately, a speech recognition engine with a specific Application Programming Interface is needed. However, the available commercial recognition engines do not provide such an interface. As a result, I have developed an engine to provide such functionalities.

I have used two methods for diphone segmentation. In the first method, I have developed a speech recognition engine based on phoneme recognition, and I have used the engine to recognize phonemes. Then the adjacent phonemes are segmented into diphones. In this method, I have assumed that a diphone begins from middle of a phoneme and extends into the middle of the adjacent phoneme. However, this assumption dose not always yield appropriate results, because a diphone may start any where inside a phoneme and end any where inside the adjacent phoneme.

To solve this problem, I have created a second method to segment natural speech into diphones directly. In this method, I have developed a diphone recognition engine by creating the acoustic model for each diphone and training the parameters of the model with diphone transcriptions of the training speech. Then the engine is employed to segment speech into diphones directly.

1.2 Implementation

To segment the speech accurately, an accurate speech recognition engine is required. To develop an accurate speech recognizer, I have used HTK tools and followed a step-by-step procedure. In each step, the parameters of the model are evaluated and modified with different training cycles. Then the model that provides the most accurate recognition is selected and passed to the next section for further processing.

First a mono-phoneme HMM for each phoneme is created, and the models are trained and tested. The model that provides the most accurate recognition is then passed to the next step. In that step, the mono-phoneme models are converted to triphone models to achieve more accurate modeling. They are then retrained and tested with different training cycles. Again, the model provided the most accurate recognition is passed to the next step of the process. In the final step, the parameters of the models are tied to create a compact model and are adapted with the voice of a person (Keith) as the test subject to achieve high accuracy.

The engine is then employed to segment and label the diphones automatically. For segmenting diphones, I have followed two different approaches. In the first approach, I have developed a speech recognition engine based on phoneme recognition, and then I have employed the engine to segment the speech into words and phonemes. Then the adjacent phonemes are segmented into diphones. In the second approach, I have developed an engine to segment the speech into diphones directly. For this purpose, the acoustic models of the diphones are created and trained. Then the model is used to recognize and segment the speech into diphones directly.

In addition to the above engine, a program is developed to demonstrate other practical uses of the engine created in this project. This program uses the speech recognizer to search media files for occurrence of an utterance. The program is able to look for an utterance of a word or a phrase in a speech by listening to the media files.

1.3 Outline of the Thesis

Chapter 1 begins with introducing the main parts of a speech recognition system and the major problems of developing a speech recognition engine. Chapter 2 discusses the digital signal processing needed to prepare the speech signal for recognition. The acoustic model employed in this thesis is explained in Chapter 3, and the methods employed for training and recognition are discussed.

The first part of implementation is explained in Chapter 4. This part consists of creating, modeling, and training the mono-phoneme models, and improving the models by converting them to triphone and tied models. The chapter ends by comparing the correctness and accuracy of the recognition achieved by different models.

Chapter 5 explains the adaptation process employed in this thesis to adjust the parameters of the models to the voice of the test subject (Keith). Chapter 6 explains the methods used for segmentation and discusses the accuracy of each method. This chapter continues by discussing other practical uses of the engine such as searching for occurrences of an utterance in a media file. Chapter 7 concludes the thesis and discusses the future works. Some detailed information is discussed in Appendixes *A* through *D*. Finally, the references used in this project are listed at the end of the thesis.

1.4 Problem of Speech Recognition

Automatic Speech Recognition is the process of mapping a speech signal to a sequence of discrete entities, for example, phonemes, words, and sentences. The major obstacle to accurate recognition is the large variability in speech signal characteristics. This variability in characteristics has three main components, Linguistic variability, Speaker variability, and Channel variability. *Linguistic Variability* includes the effects of phonetics and linguistic content of speech. *Speaker Variability* includes the effects of articulation, that is, the effects of neighboring sounds on the acoustic realization of a particular phoneme due to the continuity and motion constraints on the human articulatory apparatus [4]. *Channel Variability* includes the effects of background noise and the transmission channel, such as a microphone or telephone. All these variables impose layers of difficulty and uncertainty that must be addressed by the recognition process.

1.4 Mathematical Model of Speech Recognition

To discuss the process of speech recognition, we employ a mathematical model; then, an exact statement of the problem leads to decomposition of the problem into easier sub-problems. Our approach to designing a speech recognizer is statistical, so the mathematical model of our problem involves probabilities [5] [6].

Let **A** denote the data

$$A = a_1, a_2, \ldots, a_m \qquad a_i \in \mathrm{A} \tag{1.1}$$

on the basis of which the speech recognizer will decide which words were spoken. The symbol a_i is generated at time index I, and let

$$W = w_1, w_2, \ldots, w_n \qquad w_i \in \omega \tag{1.2}$$

denote the string of n words, each belonging to a vocabulary ω.

If $P(W|A)$ is the probability that the string W is spoken, given that the sequence of A is observed, then the corresponding mathematical formula is

$$\omega = \arg\max_{W} P(W \mid A) \tag{1.3}$$

that the recognizer will pick the most likely string of words, given the observed acoustic data.

Finding w in (1.3) is not feasible, because the permutation of possible words grows astronomically. For example, suppose our dictionary contains 4000 words, and w is the utterance of a sentence with only 3 words; then the formula (1.3) should be calculated for $(4000)^3$ possible combinations.

The well-known Bayes' formula [7] of probability theory allows us to rewrite the right hand side of (1.3) as (1.4):

$$P(W \mid A) = \frac{P(W)P(A \mid W)}{P(A)} \tag{1.4}$$

where P(W) is the probability of the word string W that will be uttered, and P(A|W) is the probability that the speaker says W and the evidence A is observed, and P(A) is the average probability that A will be observed.

Since A is fixed, the recognition problem is limited to finding the word string W that

$$\omega = \arg\max_{W} P(W)P(A \mid W) \tag{1.5}$$

maximizes the product of P(W) P(A|W).

1.5 Elements of a Speech Recognizer

Figure 1.1 is an overview of a speech recognition system. The main parts of the system are as follow.

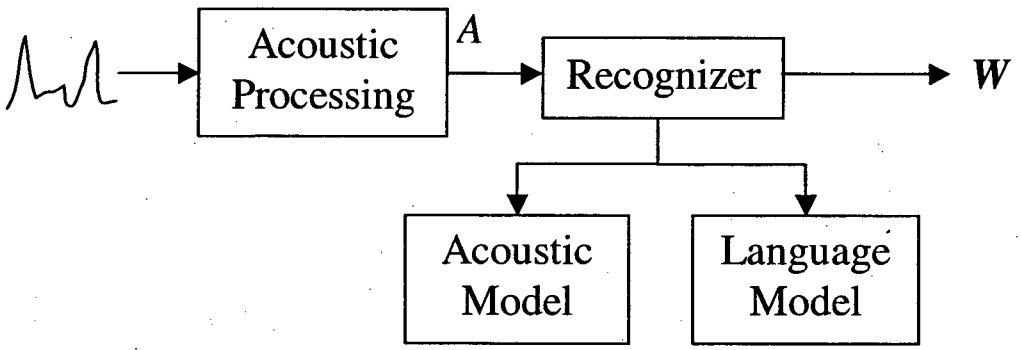

Figure 1.1 An overview of a speech recognition system.

1.5.1 Acoustic Processing

First, it is necessary to process the input signal (e.g. wave sound) and transform it into the symbol a_i that the recognizer deals with. The main requirement of speech recognition is the extraction of voice features, which may distinguish between different phonemes of a language. From a statistical point of view, this procedure is equivalent to finding sufficient statistical data to estimate phonemes. Furthermore, the process uses techniques to make the output data less sensitive to the speaker and the background noise.

1.5.2 Acoustic Modeling

Referring to formula (1.5), the recognizer needs to determine the value of P(A|W), the probability that the sequence A is observed, given that the word sequence W is uttered. Since the number of possible pairs for W and A are too large, it is not possible to create a simple lookup table. Thus, to compute the P(A|W), a statistical model is required. An acoustic model employed in speech recognition is the Hidden Markov Model (HMM).

Other models are also possible based on Neural Networks [3] [7] [8] and Dynamic Time Wrapping [2]. This project employs HMM, which is used most widely in modern speech recognition systems.

1.5.3 Language Modeling

The Formula (1.5) also requires P(W), the probability of string ***W*** that the speaker wishes to utter. The Bayes' formula allows us to decompose the P(W) as follows:

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_1,...,w_{i-1}) \tag{1.6}$$

Thus, the recognizer should determine the probability of $P(w_i \mid w_1,...,w_{i-1})$. It is logical to assume that the choice of W_i depends on the history of the previous words spoken. So (1.6) can be rewritten as

$$P(W) = \prod_{i=1}^{n} P(w_i \mid \Phi(w_1,...,w_{i-1})) \tag{1.7}$$

The art of language modeling consists of determining the appropriate classification of Φ and a method to estimate the probabilities of $P(w_i \mid \Phi(w_1,...,w_{i-1}))$.

There are many classifications in language modeling, such as *Finite State Language*[1], *Stochastic Models*[2], and *Uniform Models*. In this project we have chosen *Uniform Model*, so that every word is equiprobable and the probability of each word is $\frac{1}{V}$ where V is the dimension of the vocabulary.

---

[1] A simplified artificial language that uses finite state network to model the allowed word seqences.

[2] Based on the joint probability of a word and its preceding words.

1.5.4 Recognition

To find the transcription of W from the acoustic data A by Formula (1.5), the program must examine all possible word strings W. However, this is not possible, because the space of W is too huge to be calculated. To solve the problem, two more algorithms, Baum-Welch and Viterbi, will be introduced to make the search space feasible.

1.6 Elements of Speech

1.6.1 Phones and Phonemes

Words are natural units for the modeling of a speech recognizer, particularly since there are many applications for which isolated words are an adequate form of input. However, using words as fundamental linguistic units is wasteful of training data, and ignores any commonality between sounds within different words [9]. Thus, sub-word units are always used in speech recognition systems with large vocabularies.

Linguists have categorized the languages of the world into segments called phones, though not all linguists agree on the identity of these phones. Phones represent a base set of sounds that can be used to describe most languages. For instance, the word "*spat* " would be [s p ae t]. This indicates that the word is made up of an **s**, followed by an unaspirated **p**, a short vowel **a**, and an aspirated **t**. To determine if a consonant such as **p** or **t** is aspirated, one holds a hand in front of his or her mouth. If a breath of air is felt as the consonant in the word is uttered, then it is aspirated. For example compare "*spat*" versus "*pat.*" In the latter case, because a larger amount of air is produced, the **p** is said to be aspirated, as opposed to the unaspirated **p** in "*spat*" [9]. The set of phones is designed

to cover all languages, and the inventory of them is quite large. As a result, every language uses only a subset of the phone set. The set of unique sounds that a language uses is called its phonemes. Two sounds are considered to be different phonemes if they make a distinction between two words; these words are called minimal pairs.

There are two more sub-word segmentations defined in speech processing, diphones and triphones [10] [11]. Triphones are used in speech recognition systems while diphones are mainly used in speech synthesizers. Diphones are segments of speech that include the transition from a relatively stationary region of one phoneme to a similar region in an adjacent phoneme. Thus, diphones begin and end roughly in the middle of phonemes and span the transition between adjacent phonemes. For example, the word "spat" is made up of a set of three diphones [s-p p-ae ae-t]. As a result, there will be V^2 number of diphones in a language; however, all the diphones may not be included in a language. Furthermore, some combinations are used so rarely that they can be ignored.

A triphone is a set of phones defined by the preceding and following phonemes. For example, the word "spat" is segmented into two triphones and two diphones [s+p s-p+ae p-ae+t ae-t]. Thus, there will be V^3 number of triphones in a language[10].

1.6.2 Phonetic and Phonemic Alphabets

Linguists have found that the alphabets of English and other languages are not optimal choices for linguistic description. For example, consider the two words "thing" and "that". In these words, the sounds made by the letters 'th' are different from each other. A way must be found to distinguish between them. The system that phoneticians have devised for this purpose is called the International Pronunciation Alphabet (IPA).

This alphabet has a base on 75 consonants and 25 vowels. In 1980, a speech database called TIMIT was created and manually labeled for English. The TIMIT phone set is smaller than IPA. This database is being used to train and test for speech analysis in English.

In this thesis, I have used the CMU[3] dictionary and phoneme set. The stress marks are removed from the dictionary, because they are not suitable for speech recognition purposes [12]. Table 2.1 shows the CMU phoneme set. The two phonemes **sil** and **sp** that stand for *silence* and *short-pause* are added to the end of the table. **sil** marks the beginning and end of a sentence that usually begins and ends with a silence, and **sp** marks the boundaries of words in an utterance, which usually separates words in speech.

Phoneme	Example	Translation
AA	Odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY

[3] Carnegie Mellon University,

P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER
Sp		Short Pause
Sil		Silence

Table 1.1 The CMU phoneme set used in the thesis.

Chapter 2

Acoustic Processing

Despite differences in speaker and environment characteristics, the aim of signal processing in speech recognition is to find a relatively stable representation for different examples of the same speech sound. To prepare a signal for speech analysis, the signal is transformed by mathematical models such as FFT, LPC [13] [14] and MFCC (Mel Frequency Cepstral Coefficients) [15]. Although many different models have been used for speech recognition over the past few decades, more recently the majority of systems have converged to use MFCC. For this thesis, I have used MFCC, which has been employed by almost all of the recent speech recognition engines. Figure 2.1 shows the overall acoustic processing of input audio to be transformed to MFCC.

2.1 Sampling

Digital speech processing is usually performed by frequency sampling, ranging from 8000 samples/sec to 32000 samples/sec. Speech sampled at 16Khz contains all necessary information needed for speech recognition [16] [17].

A sampler and an A/D converter are usually included inside a computer audio card. The signal is sampled in a window and pre-emphasized. Narrow windows have been proposed to estimate the rapidly varying parameters of the vocal tract, while large windows are used to estimate the fundamental frequency. A 20-30 ms long window is generally a good compromise. In our implementation the audio signal is sampled at every

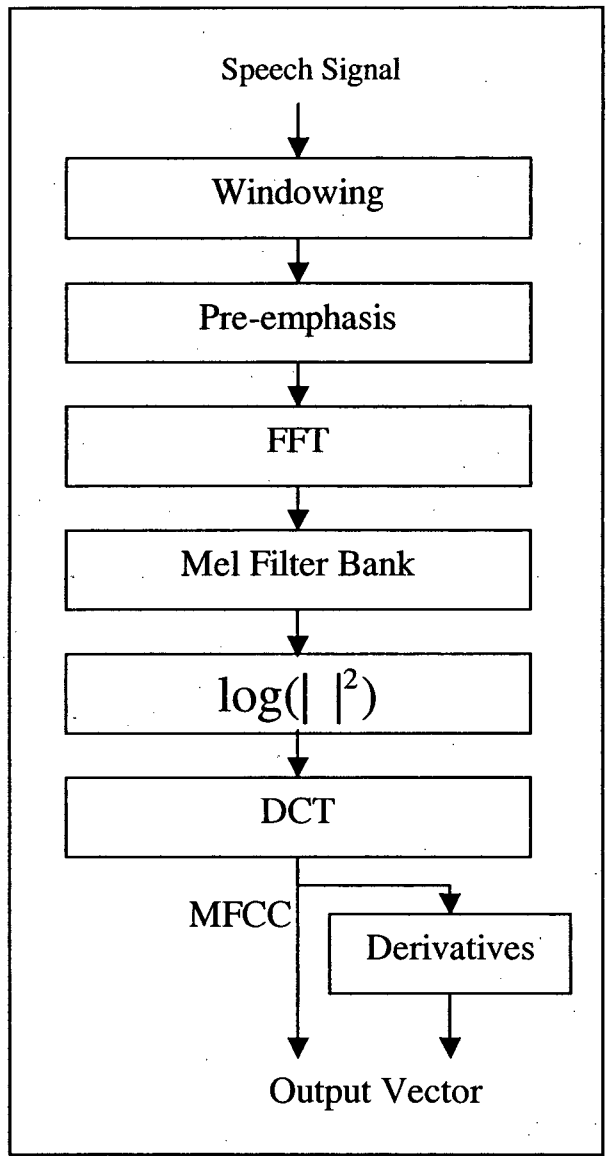

Figure 2.1 Overall picture of Acoustic Processing.

10 ms over a window of 25 ms. then its DC mean is removed and passed to the next module to be pre-emphasized.

2.2 Pre-emphasize

High frequency formants have smaller amplitude than low frequency formants. Pre-emphasis is therefore required to obtain similar amplitude for all formants. Such

processing is usually obtained by filtering the speech signal with a first order FIR filter, whose transfer function in the z-domain is

$$H(z) = 1 - a.z^{-1} \qquad 0 \le a \le 1 \tag{2.1}$$

a is the emphasis parameter [18]. In the time domain, the pre-emphasized signal is related to the input signal by this relation:

$$x'(n) = x(n) - ax(n-1)$$

A typical value for a is 0.95.

2.3 Hamming Window

The simplest window has a rectangular shape. This window is implicitly used when a sequence of N samples is retrieved from a signal:

$$w(n) = \begin{Bmatrix} 1 & 0 \le n \le N-1 \\ 0 & otherwise \end{Bmatrix}$$

The presence of a window provokes a distortion on the estimated spectrum, since the windowed signal is the convolution of the spectrum of the signal x'(n), and of the Fourier transform of a rectangular window w(n). The spectrum of w(n), $W(e^{j\omega})$ is composed of a higher energy main lobe, centered at zero frequency, and lower energy side lobes centered at higher frequencies. The main lobe spreads out in a wider frequency range than the narrow band power of the signal x'(n) represented by the formant, so the side lobes of the spectrum of the window swap energy from different and distant frequencies of x'(n). This problem is called *leakage*.

To reduce *leakage*, x(n) is multiplied by a properly shaped window, w(n). In this thesis, I have employed the Hamming Window [17], which has an impulse response as follows:

$$w(n) = \begin{Bmatrix} 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) & n = 0,...,N-1 \\ 0 & otherwise \end{Bmatrix}$$

The side lobes of this window are much lower than those of the rectangular window, and the leakage effect is decreased. The resolution of the Hamming window is less than the resolution of the rectangular window, because the main lobes of the Hamming window are wider than the main lobes of the rectangular window [19]. A Hamming window is a good choice for speech recognition, because a high resolution is not required. As indicated in Figure 2.1 the next block is FFT, and it integrates all the closest frequency lines.

2.4 Fast Furrier Transformation (FFT)

The standard methods for spectral analysis rely on the Fourier transformation of x(n). The Discrete Fourier Transform (DFT) of all frames of the signal is obtained by the following:

$$X_t(k) = X_t(e^{j2\pi N/N}) \qquad k = 0,...,N-1$$

If the number of samples, N, is a power of 2, $N=2^p$ with p as an integer, then the computational complexity can be reduced to an order of n*log*(n), resorting to the Fast Fourier Transform algorithm (FFT) [18]. Note that the phase information of the DFT samples of each frame is discarded. This is consistent with the fact that the phase does not carry useful information. Experiments have proven that the perception of a signal

reconstructed with random phases is almost indistinguishable from the original, if the phase continuity between successive frames is preserved [17].

2.5 Filter Bank

Human ears resolve non-linearly across the audio spectrum, and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves recognition performance [20]. A straightforward route to obtaining the desired non-linear frequency resolution requires a filter bank.

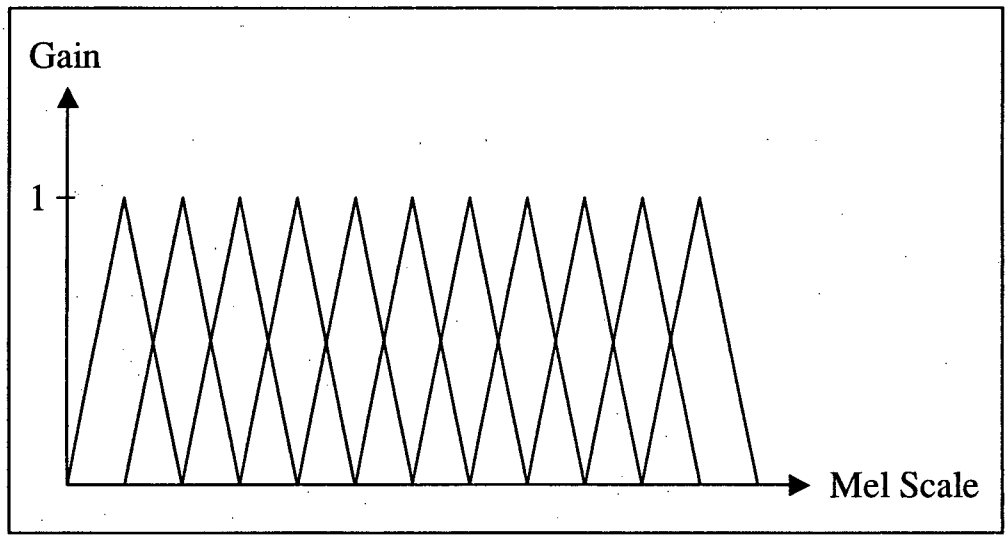

Figure 2.2 Mel Filter Bank.

Figure 2.2 illustrates the general form of a Mel-Filter bank. The filters are triangular and spaced equally along the *mel-scale* by the following:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

To implement this filter bank, the speech data is transformed using Fourier transformation, the magnitude coefficients are then multiplied by corresponding filter

gain, and the results are accumulated. In our project, I have used 22 filters equally spaced along the mel-scale [19].

2.6 Log Energy Computation

The previous procedure has the role of smoothing the spectrum performing processing, which is similar to that executed by the human ear. The next step is to compute the logarithm of the square magnitude of the signal obtained from the filter bank. Relevant benefits of this procedure are noted that the magnitude and logarithmic processing are performed by the ear as well. Furthermore, squaring the magnitude discards useless phase information, and calculating the logarithm of the result is a method of dynamic compression that makes feature extraction less sensitive to the variations of speech.

2.7 Discrete Cosine Transform

The final procedure for the Mel frequency cepstrum computation (MFCC) consists of performing a Discrete Cosine Transformation, DCT [17]. The DCT has the property of producing highly un-correlated features [18]. The zero order MFCC coefficient is approximately equivalent to the energy of the frame [21]. The DCT also has the effect of smoothing the spectrum, but only if the first coefficients are retained. The number of MFCC coefficients is generally lower than 15 in speech recognition. Typical values are from 9 to 15 coefficients. In Figure 2.3 and Figure 2.4 some partial results of MFFC computation are displayed.

A further improvement in recognition is obtained by considering that the Cepestral parameters do not take into account the dynamic evolution of the speech signal.

As the result, the first and second order differences of the MFCC may be used to capture such information. Hence, given vector U the in the time domain, the i-th order time differences can be computed as [11] :

$$\Delta^i\{u_t\} = \Delta^{i-1}\{u_{t+1}\} - \Delta^{i-1}\{u_{t-1}\}, \qquad \Delta^0\{u_t\} = u_t$$

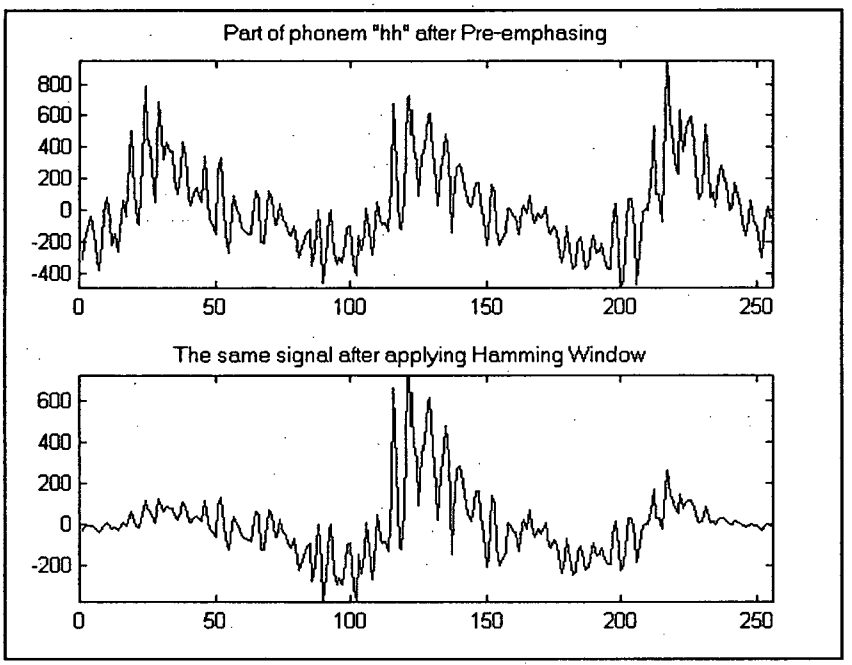

Figure 2.3 Part of the signal of phoneme "hh"

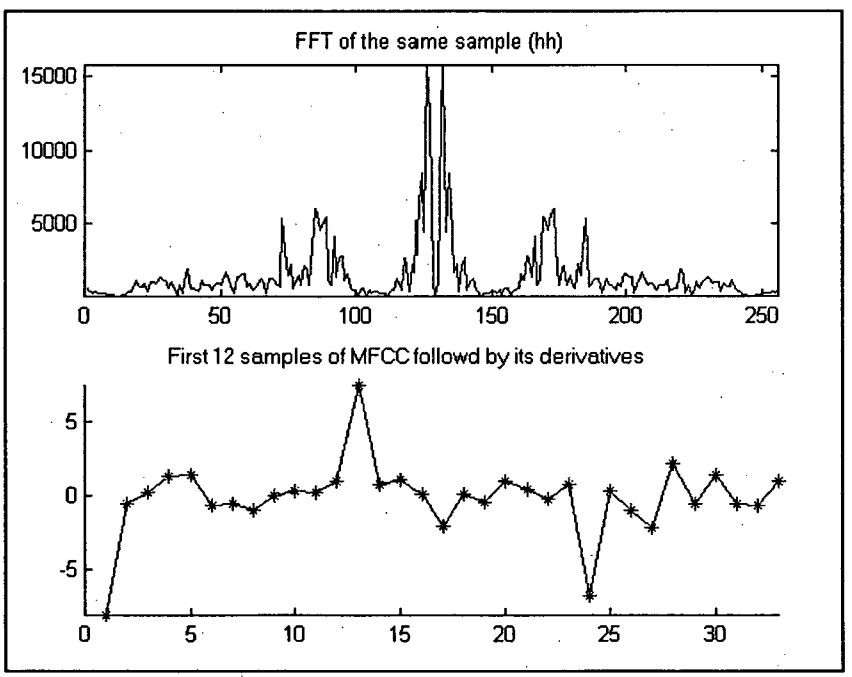

Figure 2.4 (Up) FFT of the signal at Figure2.3, **(Down)** The first 12 MFCC samples of the same signal are followed by its differences.

In this project the output vector for each frame, 25 ms, is composed of 13 first coefficients known as static parameters of MFCC, followed by first and second order differences, known as delta and acceleration coefficients respectively. So each frame is transferred to a vector of 39 elements of data.

Chapter 3

Acoustic Model

The Acoustic Model used in almost all advanced speech recognition systems is based on the Markov Chain. A Markov Chain consists of a number of states with transitions among them. A probability is associated with each transition and a symbol is

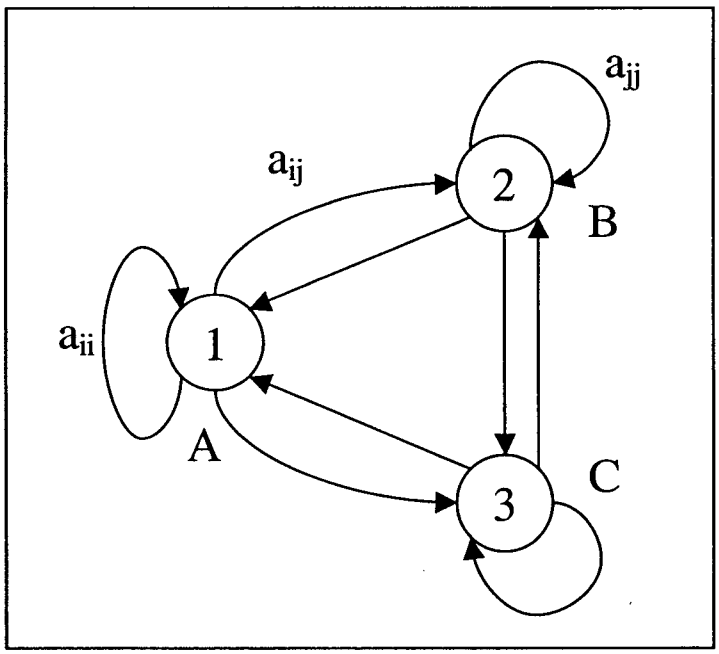

Figure 3.1 A three-state Markov chain.

associated with each state. Figure 3.1 shows a three-state Markov chain with transition probability a_{ij} between states *i* and *j*. The symbols A, B, and C are associated with states 1, 2, and 3 respectively. If a transition occurs from 1 to 2, symbol B will be produced as an output, or a transition from 3 to 1 will produce symbol C. Note that in a Markov Chain the transition between states are probabilistic, while the production of the output symbols

are deterministic. For example, the transitions (1 2 2 3 2 1) will produce symbols BBCBA as output [22].

3.1 Hidden Markov Model

A Hidden Markov Model (HMM) is the same as a Markov Chain, with one difference. In HMM, the output symbols are probabilistic too. Thus, instead of associating a symbol with each state, all symbols can be produced within all states with a different probability, and a probability distribution of all the output symbols is associated with each state. The probability associated with each state is known as the output probability.

Figure 3.2 shows a three state HMM. It has the same transition probabilities as the Markov Chain defined in Figure 3.1, but a probability distribution is associated with output symbols [A B C D E]. Now, when a transition occurs from one state to another

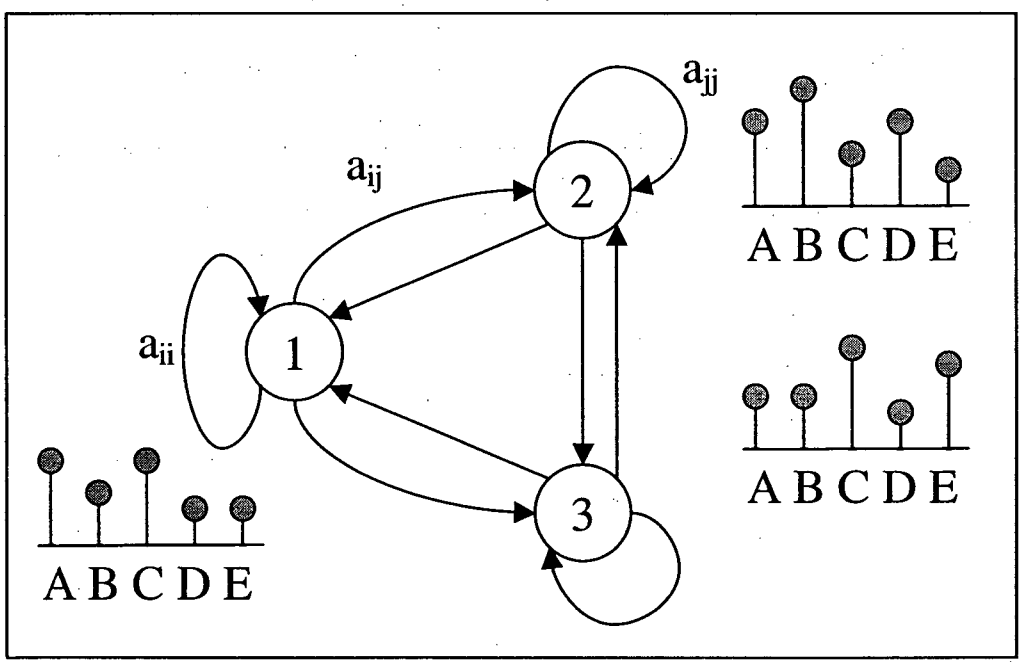

Figure 3.2 A three-state Hidden Markov Model.

state, an output symbol is generated according to the probability distribution of the corresponding state. Given a sequence of symbols generated by the HMM of Figure 3.2, it is not possible to know for certain what state sequences have generated the observed output. For example, if the output symbols are "A B B E C," there is no way to know for certain which sequences have produced them. In fact, every sequence of states with the same length of output symbols is a possible sequence with a different probability. It is said that the sequences of the symbols are hidden from the observer if the output symbols are the only things an observer sees. This is why this model is called a Hidden Markov Model [22].

Instead of having a discrete number of output symbols, a probability density function may be defined over all possible values of the output vectors.

3.2 Model of Phonemes

Figure 3.3 shows an example of a three state HMM for a single phoneme. This model has only three states with two null states, one in the start and one at the end of the model. The null states are only used as moderators to connect HMM models to each other, and they have no active role. The HMM is not limited to three states as shown in our example; it can be any size, and its use is not limited to speech applications. For instance, image recognition [23] [24], control systems [25], segmentation of DNA sequences, and gene recognition [26], are among the interesting topics conducted recently using HMM.

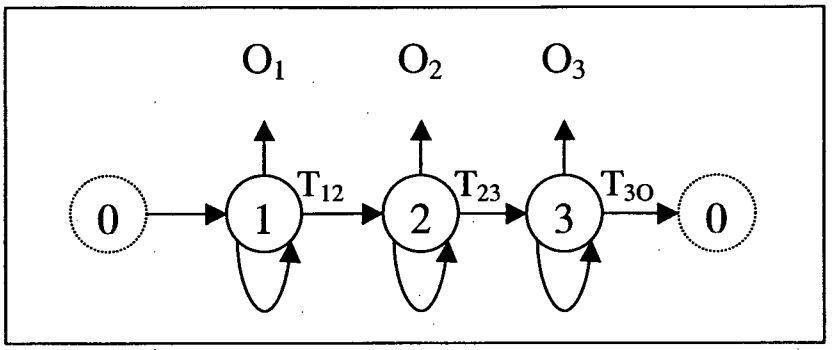

Figure 3.3 Basic structure of a phonetic HMM.

Now, let us examine how HMM works. We start with state 0. The first frame is read from the input and a transition is made from state 0 to state 1. The O_1 , mfcc of the frame is computed. Then $P(O_1)$, the probability of the observation O_1 from states 1, is calculated. Then the next frame is read from the input, and if we assume that a transition is made from state 1 to state 2, the previous probability is multiplied by the transition probability from state 1 to state 2, that is $P(O_1) * P(T_{12})$. Then, the mfcc of the frame is computed and the probability of the observed O_2 , $P(O_2)$, from state 2 is calculated and multiplied with the previous product, $P(O_1) * P(T_{12}) * P(O_2)$. The process is continued until the model is exited through T_{30}. At this point we can assume that the phoneme modeled by the given HMM is pronounced. Multiplying the sequence of output and transition probabilities gives the total probability that the input spectral sequences were generated by the HMM, using a specific sequence of states. For every sequence of states, a different probability value results. For recognition, the probability computation just described is calculated for all possible phoneme models and all possible state sequences. The sequence that provides the highest probability value is assumed to be the recognized sequence of states and phonemes. However, this approach is not totally realistic because of the very large number of sequences involved. In general, there will be $(N)^T$ sequences,

where N is the number of states in the model and T is the number of frames in the observed sequence. For example, if there are n states in the model, then the number of probable sequences for speech that lasts only one second will be equal to N^{100} with the parameters we have used in this project. To overcome this obstacle, we will introduce two common algorithms, Baum-Welch and Viterbi algorithms, and an alternative approach to deal with this problem.

The HMM shown in Figure 3.3 is known as the first order left-to-right Hidden Markov Model. Mathematically, if 1, 2,..., N is the number of the frames observed and $o_1, o_2,\ldots, o_n$ are the observed outputs, then [27],

$$P(o_t = j \mid o_{t-1} = i, o_{t-2} = k,...) = P(o_t = j \mid o_{t-1} = i) \qquad (3.1)$$

That means the output at time j depends only on the value at the preceding time and on nothing that went on before. Also,

$$P(o_t = j \mid o_{t-1} = i) = P(o_{t+l} = j \mid o_{t+l-1} = i) \qquad (3.2)$$

indicates that the Markov Chain is time invariant.

This model is adopted for speech modeling because in speech, time flows in a forward direction. The first node at the left-side of the phoneme stands for the beginning of a phoneme, the middle is where the phoneme reaches a steady state, and the third is the last transition of the phoneme. Transitions from any state back to itself serve to model the variability of the speech, since different utterances made by different people, or by the same person in different contexts, have different duration.

3.3 Speech Recognition Using HMM

Mathematically speaking, each HMM model M_i, i=1, 2,...m, is defined by a parameter set M=[A, B, π] where

$\pi = \{ \pi_i \}$ is a column vector denoting the initial state--the probability of the model of starting at state i.

$A=\{a_{ij}\}$ is a square matrix indicating the probability of transition from state i at time t to state j at time $t+1$.

$B=\{b_{jk}\}$ is a column vector indicating the probability of the model emitting output O_k at state j.

The likelihood of each model M_i having produced the observation O_t is obtained by computing $P_i\{O_t|M_i\}$; that is the probability of observing sequence O_t given model M_i. Then the recognized phoneme is given by

$$P = \underset{i=1,2,\ldots,m}{ArgMax}[P_i\{O_t \mid M_i\}] \qquad (3.3)$$

where ArgMax denotes the value of the argument that maximizes the expression.

The obvious way to calculate P is to consider all possible state sequences and then select the sequence that produces the maximum probability. As discussed above, this approach is not feasible, if there are N states and P frames then the total number of the possible states will be N^P. Fortunately, there are two recursive algorithms to reduce computation to a tractable amount.

3.4 Baum-Welch Algorithm

The Baum-Welch algorithm is based on calculating *Forward Probability*. Forward probability, $\alpha_t(j)$, is the probability of observing the partial sequence (o_1, o_2,…,o_t) and being in state j at time t.

$$\alpha_t(j) = P(o_1, o_2, ..., o_t, q_t = j) \tag{3.4}$$

Thus, the total probability of observing O, P{O|M} can be obtained by summing $\alpha_T(j)$ across all N states. When P{O|M} is calculated in this way, it is called the Baun-Welch probability.

$$P_{BW} = \sum_{j=1}^{N} \alpha_T(j) \tag{3.5}$$

To calculate P_{BW}, suppose that $\{\alpha_T(j), j=1, 2,...N\}$ has been calculated at some time instance t. Then the probability of observing sequence (o_1, o_2,…,o_t) and being at state i at time t and transferring to state j at time $t+1$ is equal to $\alpha_t(i).a_{ij}$. Thus, the probability of being at state j at time $t+1$ and observing sequence (o_1, o_2,…,o_t) may be obtained by summing $\alpha_t(i).a_{ij}$ over all states, and the equation 3.5 changes to

$$P_{BW} = \sum_{i=1}^{N} \alpha_t(i).\alpha_{ij} \tag{3.6}$$

Consider that the observation o_{t+1} is produced by state j at time $t+1$, so we have

$$\alpha_{t+1}(j) = \left\{ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right\} b_j(o_{t+1}), \qquad t = 1,2,...,T-1 \tag{3.7}$$

where $b_j(o_{t+1})$ is the probability of producing o_{t+1} from state j. The recursion in (3.6) is initialized by computing the $\alpha_1(j)$ in (3.7);

$$\alpha_1(j) = \pi\{j\} b_j(o_1) \tag{3.8}$$

is the probability of observing the fisrt output and being in state *j* at *t=1*.

3.5 Viterbi Algorithm

In computing P_{BW} in recursion 3.5, the forward probabilities of all states are accumulated at time *t*. In the Viterbi Algorithm, P_V, only the likelihood of the most likely state sequence emitting the observation O is calculated.

$$\phi_{t+1}(j) = \underset{i=1,2,...N}{Max}\{\phi_t(i) a_{ij}\} b_j(o_{t+1}), \qquad t = 1,2,...,T-1 \tag{3.9}$$

$$\phi_1(j) = \pi\{j\} b_j(o_1) \tag{3.10}$$

This equation is identical to (3.7), except that the summation is replaced by the *Max* operator and the algorithm is initialized using (3.10), where α is replaced by ϕ. Thus, the probability of observing sequence O is given by

$$P_V = \underset{j=1,2,...,N}{Max}\{\phi_T(j)\} \tag{3.11}$$

In practice, all the probabilities are on a logarithm base. Having the probability in *log* base reduces the multiplication to addition, which is faster and prevents the results from falling too low, causing an under flow problem. Multiplying numbers smaller than 1 will result in even smaller numbers. Thus (3.9) becomes

$$\phi_{t+1}(j) = \underset{i=1,2,...,N}{Max}\{\phi_t(i) + \log(a_{ij})\} + \log(b_j(o_{t+1})), \qquad t = 1,2,...,T-1 \tag{3.12}$$

3.6 Token Passing Method

The token passing method is based on the Viterbi algorithm. A token represents a partial path through the network, extended from time zero to time *t* [20]. At time zero, a token is placed in every possible start node. Then for each frame, tokens are moved to the next node along connected transitions. When there are multiple exits from a node, the token is copied to explore all possible paths. As the token passes through the transitions and nodes, its log probability increases according to the corresponding transition and emission probabilities. An HMM can have at most N tokens. Hence, at the end of each time step, all but N best tokens in each HMM are discarded. Each token has a history that records its path as it propagates through the nodes. The token that has the highest probability will be declared the winner, and its path will become a recognized route.

To reduce the number of tokens and hence speed up processing, only the tokens that have a chance of being among the winners are propagated, and others are deleted from memory, known as *pruning*. Pruning is implemented at each time step by removing all tokens whose probabilities fall below a *beam-width*[4] [20]. Setting the beam-width is crucial; if it is too small, then the most likely path might be pruned before its token reaches the end of the utterance. If it is too large, the processing time will be long.

The extension to continuous speech recognition simply involves connecting models of phonemes together in a sequence. The reason for including the null nodes at the entry and exit states should now be evident; these nodes provide the glue needed to join the models together.

[4] The Beam-width is the distance of the log probability of the nodes from the node that has the highest probability.

3.7 Training of HMM

Now we have reached the practically difficult problem; how to estimate the HMM parameters in the first place. Before discussing the parameter estimation in detail, let us clarify the output distribution probability, $P(b_j(o_t))$. The following formula is used to calculate the output probability.

$$b_j(o_t) = N(o, \mu_j, \Sigma_j) \tag{3.13}$$

Where $N(o, \mu, \Sigma)$ is a multivariate Gaussian with mean vector μ_j and covariance matrix Σ_j [20], and n is the dimension of the output vector, O.

$$N(o, \mu_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{\frac{1}{2}(o-\mu_j)' \Sigma_j^{-1} (o-\mu_j)} \tag{3.14}$$

3.8 Baum-Welch or Forward-Backward Estimation

If there were only one state j, then the HMM parameter estimation would be a simple average (3.15).

$$\bar{\mu}_j = \frac{1}{T} \sum_{t=1}^{T} o_t \tag{3.15}$$

$$\bar{\Sigma} = \frac{1}{T} \sum_{t=1}^{T} (o_t - \mu_j)(o_t - \mu_j)'$$

In practice, there are multiple states and no direct assignment of observation vectors to the individual states. However, equations 3.15 are used to make an initial estimation of the parameters.

Now, let $L_j(t)$ denote the probability of being in state *j* at time *t*, that is,

$$L_j(t) = P(q_t = j \mid O) \tag{3.16}$$

Then, the equations (3.15) become

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \tag{3.17}$$

and

$$\bar{\Sigma} = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)} \tag{3.18}$$

Equations 3.17 and 3.18 are Baum-Welch estimation formulae for the means and covariance of a HMM. These equations can be applied if the $L_j(t)$ is known for state *j*. The $L_j(t)$ is calculated using the *Forward-Backward* algorithm.

3.9 Forward-Backward Algorithm

We have discussed *Forward probability* before, however, we repeat it here for context cohesion. The forward probability is defined as,

$$\alpha_j(t) = P(o_1, o_2, ..., o_t, q_t = j) \tag{3.19}$$

$\alpha_j(t)$ is the joint probability of observing the first *t* frame vectors (o_1, o_2,... o_t) and being in state j at time t. The forward probability can be calculated with the following recursion:

$$\alpha_j(t) = [\sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij}] b_j(o_t) \tag{3.20}$$

Note that the first and N_{th} states are null nodes, non-emitting nodes. Initial conditions are

$$\alpha_1(1) = 1, \quad \alpha_j(1) = a_{1j} b_j(o_1) \tag{3.21}$$

and the final condition is

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T-1)a_{iN} \tag{3.22}$$

as we have shown before. Comparing (3.22) and (3.6) indicates that the calculation of the forward probability also yields the total likelihood P(O|M).

Backward Probability, $\beta_j(t)$ is the probability of observing ($o_{t+1},\ldots,o_T$) and being at state j at time t.

$$\beta_j(t) = P(o_{t+1},\ldots,o_T \mid q_t = j) \tag{3.23}$$

The backward probability is named as a *conditional probability*, and can be computed using the following recursion,

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1})\beta_j(t+1) \tag{3.24}$$

with the initial condition given by

$$\beta_i(T) = a_{iN} \qquad 1 < i < N \tag{3.25}$$

and, the final condition is

$$\beta_1(1) = \sum_{j=2}^{N-1} a_{1j} b_j(o_2)\beta_j(2) \tag{3.26}$$

This symmetric definition is deliberate, since it allows the probability of the state occupation to be determined by production of the two probabilities. Thus,

$$P(O, q_t = j) = \alpha_j(t)\beta_j(t) \tag{3.27}$$

is the joint probability of observing O and being at state j at time t.

Referring to the definition $L_j(t)$, we have

$$L_j(t) = P(q_t = j \mid O, M) \tag{3.28}$$

Hence[5]

$$L_j(t) = \frac{P(O, q_t = j \mid M)}{P(O \mid M)} \tag{3.29}$$

By substituting from equation (3.27) in (3.29), we have

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{P_{BW}} \tag{3.30}$$

where P_{BW}=P(O|M).

The training of the HMM model involves assuming an initial estimate of the model, M=[A, B, π], and re-estimating it with known training sequences. For each sequence O, the parameters of a new model M_{new} are re-estimated from those of the old model M_{old}, until

$$P\{O \mid M_{new}\} \geq P\{O \mid M_{old}\} \tag{3.31}$$

At each iteration, the old model is replaced by a new model, M_{new}, and another re-estimation takes place, while equation 3.31 is satisfied. According to the Baum-Welch algorithm, the transition matrix $\{a_{ij}\}$ is calculated as follows:

$$\bar{a}_{ij} = \frac{A_{ij}}{\sum_{k=2}^{N} A_{ik}} \tag{3.32}$$

where A_{ij} represents the total number of transitions from state *i* to state *j*.

In this style of training, a set of training observations, O, is used to estimate the parameters of a single HMM. The basic formula for the re-estimation of the transition probabilities is

---

[5] The conditional probability rule: $P(A \mid B) = \frac{P(AB)}{P(B)}$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\sum_{t=1}^{T-1} \alpha_i(t) \beta_i(t)} \tag{3.33}$$

For details of formula and proof of convergence of the algorithm, refer to [20] [28].

3.10 Adaptation

To achieve accurate recognition, the parameters of the model should be trained by a specific user. However, providing enough training data for each user to tailor the system to his or her voice is difficult in practice. To achieve an accurate recognition engine, a huge amount of training data has to be provided for each user. An alternative to this training strategy is adaptation. In this case, the system is trained with different voices as a user independent system; then the parameters of the model are tuned to a specific user. In this training system, the amount of adaptation data is much less than the data needed to train the system from the start as mentioned in the first case.

3.10.1 Model Adaptation Using MLLR

Maximum Likelihood Linear Regression or MLLR computes a set of transformations that will reduce the mismatch between an initial model set and the adaptation data. More specifically, MLLR is a model adaptation technique that estimates a set of linear transformations for the mean and variance parameters of the Gaussian mixture of the HMM system. The effect of these transformations is to shift the component means and alter the variances in the initial system, so each state in the HMM is more likely to generate the adaptation data. MLLR uses a Transform-Sharing method, known as a Regression Class Tree, to adapt the parameters of the model. This method

provides a means of dealing with the small amount of adaptation data. Under this scheme, the system adapts even the parameters for speech not presented in the adaptation data. As a result, the system is able to adapt its parameters with only a small amount of adaptation data. For more information about *Regression Class Tree,* refer to Appendix A.

The transformation matrix used to give a new estimate of the adapted mean is given by

$$\mu = W\xi \tag{3.34}$$

Where W is a n x (n+1) transformation matrix and ξ is the extended mean vector (n is the dimension of the vector).

$$\xi = [w\mu_1\mu_2...\mu_n]^T \tag{3.35}$$

w is the bias offset whose value is fixed and is usually equal to 1. As a result W can be written

$$W = [b \quad A] \tag{3.36}$$

The A represents an [n x n] transformation matrix, and b a bias vector. The transformation matrix, W, is obtained by solving a maximization, using the *Expectation-Maximidation* (EM) algorithm. This technique is also used to compute the variance transformation matrix. The same rules, with some modifications, apply to finding the variance transformation matrix.

$$\Sigma = B^T HB \tag{3.37}$$

Where H is the linear transformation to be estimated and B in the inverse of the Choleski factor of Σ^{-1},

$$\Sigma^{-1} = CC^T \tag{3.38}$$

$$B = C^{-1} \tag{3.39}$$

For details of the EM algorithm and an example of calculation adapted data refer to Apendix B.

3.11 Language Model

One of the recognition components is the language model. The language model is a network of words arranged according to some rules, for example, grammar rules of the language to be recognized. The simplest word-network model is a list of parallel nodes connected by arcs. The nodes represent words and the arcs represent the transitions between words. Figure 3.4 shows a simple network. The top figure is used to recognize the "Flip Flop," or "Flop Flip," while the bottom diagram can be used to regonize any combination of the two words, such as "Flip Flip FlipFlop."

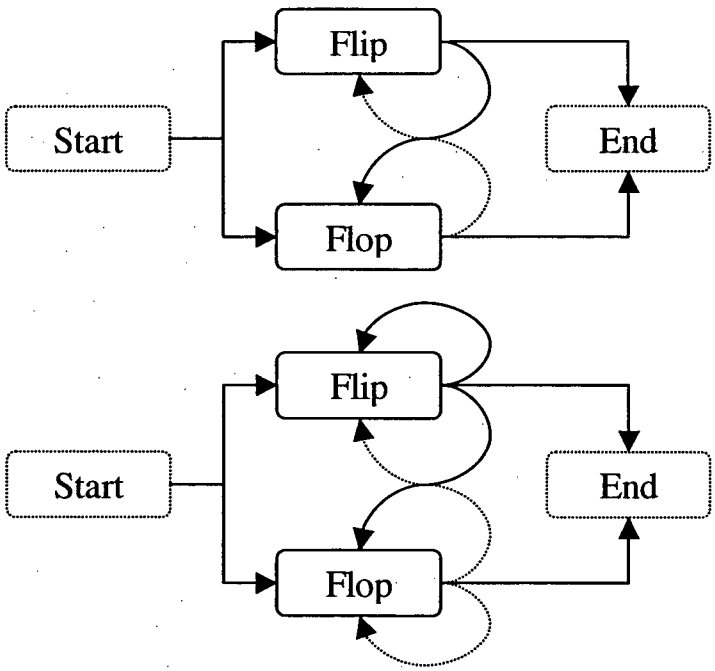

Figure 3.4 **(Top)** A simple network recognizes "Flip Flop" or "Flop Flip." **(Bottom)** A modified network recognizes any permutation of "Flip" and "Flop."

A probability factor may be assigned to each arc to indicate the probability of the word sequences. However, the simplest and most common form is a network in which each word has the same possibility of being pronounced. The network may contain any number of words; however, increasing the number of words will increase the processing time and decrease the accuracy of recognition. The probability of selecting the correct word from the words listed in the network will decrease as the network grows.

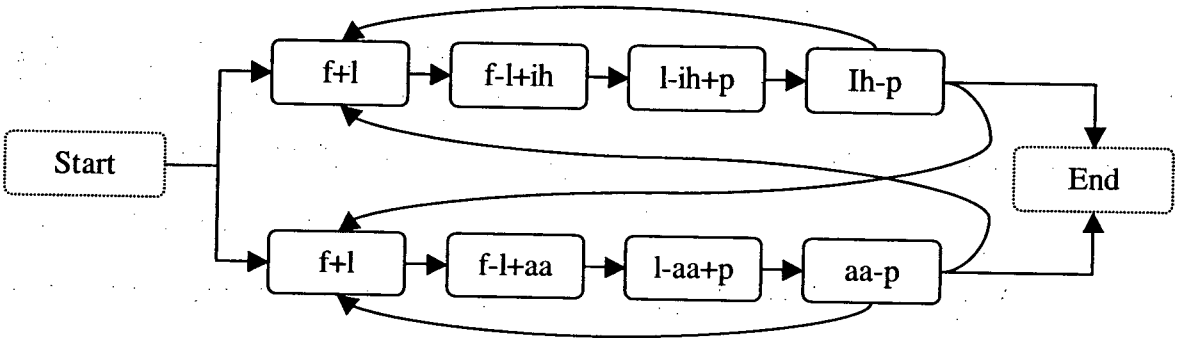

Figure 3.5 Word Internal Triphone Expansion of Flip-Flop Network.

For the recognition process, the recognizer loads the network and creates the HMM equivalent of the network. Then it employs the algorithms defined previously to find the best possible route as recognized speech. Figure 3.5 shows the same network as indicated in Figure 3.4, expanded to its tri-phone equivalent by the recognizer for the recognition process.

The Network is stored in SLF format. For example, the network shown in Figure 3.5 is stored as follows:

```
# Define size of network: N=num nodes and L=num arcs
N=4   L=8
# List Nodes: I=node-number, W=word
I=0   W=start
I=1   W=end
I=2   W=Flip
I=3   W=Flop
# List arcs: J=arc-number, S=start-node, E=end-node
J=0   S=0   E=2
J=1   S=0   E=3
```

```
J=2   S=3   E=1
J=3   S=2   E=1
J=4   S=2   E=3
J=5   S=3   E=3
J=6   S=3   E=2
J=7   S=2   E=2
```

The first line defines the size of the network. The *Start* node is a node without a predessor and the *End* node is a node without a successor. There should be one, and only one, *Start* and *End* node in a network.

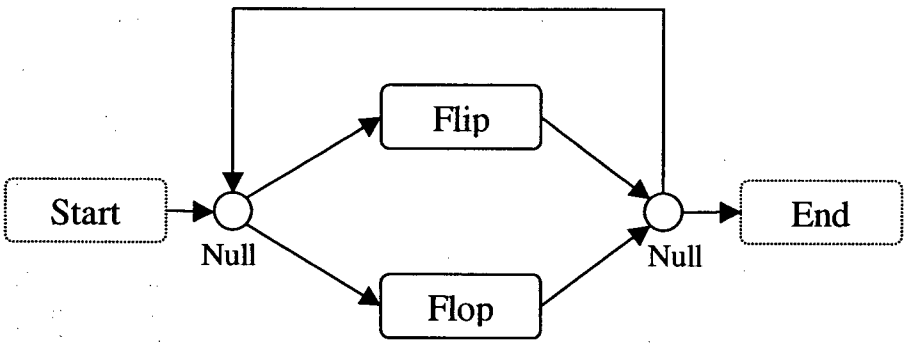

Figure 3.6 The modified version of the network shown in the bottom of the last figure.

To simplify the network, a NULL node is itroduced. For example, the network defined in the bottom of Figure 3.4 is modified in Figure 3.6 and its equivalent SLF file is as follows:

```
# Define size of netword: N=num nodes and L=num arcs
N=6   L=7
# List Nodes: I=node-number, W=word
I=0   W=start
I=1   W=end
I=2   W=Flip
I=3   W=Flop
I=4   W=!NULL
I=5   W=!NULL
# List arcs: J=arc-number, S=start-node, E=end-node
J=0   S=0   E=4
J=1   S=4   E=2
J=2   S=4   E=3
J=3   S=2   E=5
J=4   S=3   E=5
J=5   S=5   E=4
J=6   S=5   E=1
```

Chapter 4

Modeling & Training

The goal of this thesis is to develop a speech recognizer. Then the recognizer will be employed to develop a search engine to spot and play words in an AVI file. The recognizer will also be used to develop a Diphone extractor program to segment the speech into diphones.

To develop a speech recognizer, I have followed a step-by-step procedure. First, a simple mono-phoneme model has been developed and trained with different training cycles; the best model is then transformed into a triphone model and retrained. Finally, the best triphone model is tied and adopted to the speech of the test speaker to get the best possible accuracy. For developing the speech recognizer, I have employed HTK tools, provided by Cambridge University. The tools are available in C source code. I have modified some parts of the code to fit the project, and also developed some tools as needed. HTK tools are used for preparing the speech files, modeling the acoustic data, training the HMM, and testing the system. Modifications of the tools are made for recognition. Finally, a program is developed to demonstrate some applications of the system, such as word spotting, word and diphone segmenting.

4.0 Introduction

To develop an accurate speech recognizer, I have set different experiments. In each set of experiments, the model that results in the best recognition output has been chosen and the experiments continue with the new model. Experiment begins by defining

and training a set of mono-phoneme HMMs. It continues by selecting the best mono-phoneme model, converting it to triphone HMM, and retraining. The model that results the best recognition output is chosen and the parameters of the HMM are tied and adapted to a subject's (Keith's) voice. The results of each step are compared with the previous step's results, and the best model is chosen to continue. It is noted that all of these experiments are based on previous experiments that I have set and tested for this project, and the set of experiments shown here are employed to indicate the logical flow of the project.

4.1 Acoustic Processing

The project employs the speech utterances provided by **TIMIT** and one student, **Keith**, as the subject of the experiment. The data provided by Keith is in AVI format. The WAVE part of the file is extracted and used for processing and recognition. Each audio file, both TIMIT and Keith's utterances, passes through the following process to create its MFCC equivalent.

1. Sample each file for every 10 ms in the window of 25 ms, so each sampled frame overlaps with the adjacent frames for a duration of 15 ms.
2. Pre-emphasize the sample according to section (2.1) with $\alpha = 0.95$
3. Apply Hamming Window to each frame.
4. Find the Furrier Transform of each frame.
5. Pass each vector through a Mel-filter bank with gain equal to 1 and the number of filters equal to 22.
6. Calculate the Log Energy of each vector as described in section (2.6)

7. Find the DCT of each vector and discard all except the first 13 coefficients, and then add its first-order and second-order derivatives to the end of the vector.

8. Store the output in the same file name with an "mlf" extension.

According to the process described above, each frame of each sample, 25 ms, is converted to a vector of 39 elements and each second of speech is converted to 100 vectors, which are stored in a file with an "mlf" extension.

HTK provides a flexible tool, HCopy, for converting files to different formats, such as converting WAVE to MFCC. For detailed information and the parameter list needed for such transformations, refer to the HTK manual.

4.2 Acoustic Modeling

Each phoneme is modeled using a three state left-to-right HMM, as shown in Figure 3.3, and each observation probability distribution is represented with a mixture of Gaussian density as described in formula (3.14). For each phoneme listed in Table 1.1 except for *sp*, which stands for *short pause*, a similar model is created, and the parameters of each model are stored separately.

```
~h "hmm1"
<BeginHMM>
 <NumStates> 5
 <State> 2
  <Mean> 39
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  <Variance> 39
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 <State> 3
  <Mean> 39
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  <Variance> 39
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 <State> 4
  <Mean> 39
```

```
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  <Variance> 39
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 <TransP> 5
   0.0 1.0 0.0 0.0 0.0
   0.0 0.6 0.4 0.0 0.0
   0.0 0.0 0.6 0.4 0.0
   0.0 0.0 0.0 0.7 0.3
   0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

This is a HMM prototype model used by HTK tools. The first symbol in the model is **~h**, followed by a phoneme name, **hmm1**. Then the HMM definition is bracketed by the symbols **<BeginHMM>** and **<EndHMM>**.

<NumStates> defines the number of states in the HMM. In our project the number of states is equal to 5, with the first and last states set to NULL states. For each emitting state *j*, a single vector μ_j is introduced by the keyword **<Mean>,** and a diagonal variance vector Σ_j is introduced by the keyboard **<Variance>**. Finally, the definition ends with the transition matrix $\{\alpha_{ij}\}$, introduced by the keyword **<TransP>**.

I modified the **sil** model by adding two extra transitions, one from state 2 to state 4, and another from state 4 back to state 2. The idea here is to make the model more robust by allowing individual states to absorb the noise in the training data. Also, at this point, a one state short pause (*sp*) model is added to the list of HMMs. This model is called *Tee-Model* and makes a direct transition from the entry node to the exit node. The emitting parameters of the *sp* model are set to the emitting parameters of state 2 of the *sil* model. Figure 4.1 shows the modified *sil* model and *sp* model. The definition of the *sp* model is shown below:

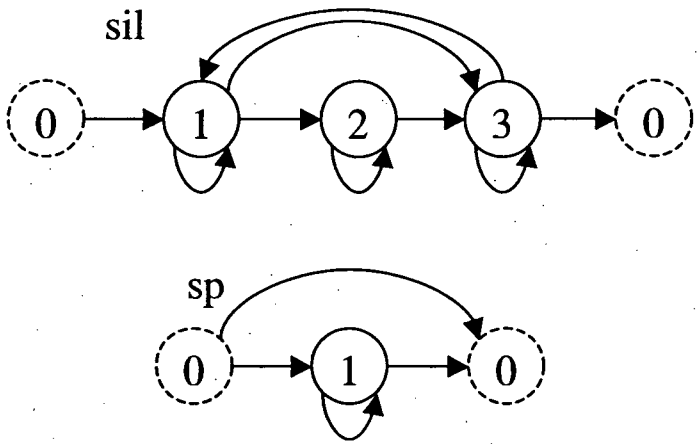

Figure 4.1 *sil* and *sp* model.

```
~h "sp"
<BeginHMM>
 <NumStates> 3
 <State> 2
  <Mean> 39
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  <Variance> 39
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
<TransP> 5
   0.0 0.5 0.5
   0.0 0.5 0.5
   0.0 0.0 0.0
<EndHMM>
```

4.3 Training

The Mean and Variance values, set to 0 and 1 respectively, are only for demonstration purposes. In fact, to begin the training process, the mean and variance of all the models are set to the global mean and variance, computed by scanning the set of training data. HTK provides a tool HcompV, for computing the global mean and variance by scanning *.mlf files. The details of how to use the tool are not provided here; interested readers should refer to the HTK manual for detailed information.

4.3.1 Mono-phone training

The training process needs both *.mlf files and their phonetic equivalents. The phonetic equivalent of each utterance is extracted from the dictionary[6] and a *sp* is inserted between word boundaries to separate words in a sentence. For example, the utterance "Her hum became a gurgle of surprise" (file si1837.lab from SI training list of TIMIT) is converted to the following phonetic equivalent:

```
sil hh er sp hh ah m sp b ih k ey m sp ah sp g er g
ah l sp ah v sp s er p r ay z sil
```

The sentence begins and ends with a *sil* (silence) and a *sp* is inserted to indicate the boundaries of each word. Inserting *sil* and *sp* could be clarified by referring to Figure

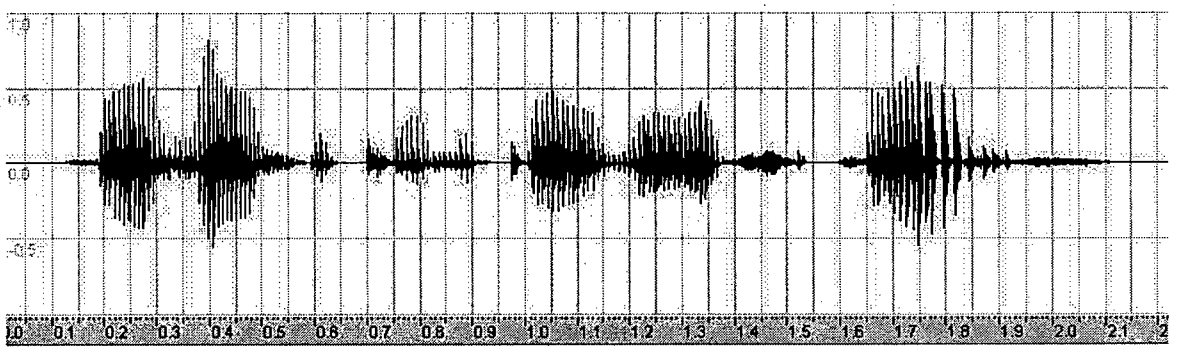

Figure 4.2 The wave form of "Her hum became a gurgle of surprise."

4.2, which shows the wave form of the same sentence uttered by a male in the New England region in the United states. As indicated in the audio file, the speech begins and ends with a silence and each word is separated by a short pause, *sp*. Notice that the *sp* may have a zero duration, which means there is no short pause between the words. This is why we have created a direct transition from the first node to the last node in the *sp* model.

The training module creates the HMM equivalent of each utterance by sticking the HMMs as indicated to the equivalent label file. Then it employs a *forward-backward* algorithm, described in section (3.9), to calculate the parameters of the model, and replaces the old model with a new one, and stores the result. HTK provides a very flexible tool, *HRest*, for training purposes. This tool accepts different parameters for training. In this project I have employed *HRest* with the following parameters "`-t f [a b]`".

Selecting the parameters for "`-t f [a b]`" is very important, because without setting them, the cycles consumed by the processor for training become enormous. `-t f` sets the pruning level to `f`. The default value for the pruning level is zero; that means no pruning at all. During the backward probability calculation, at each time all (log) values falling more than `f` below the maximum value, at that time, are ignored. During the subsequent forward pass, the *log* values are only calculated if there are corresponding valid values. Tight pruning results in failing to process an utterance, and a high value for `f` requires more processing time. If *a* and *b* options are given, then a pruning error results in the threshold being increased by *a* and utterance processing restarts. If the error continues, this procedure will be repeated until the limit *b* is reached. In this project the f, a, and b are set to 250, 150, and 1000 respectively.

[6] This project employs the dictionary provided by the Carnegie Mellon University by removing the stress marks that are not suitable for speech recognition [12].

4.3.2 Triphone training

In this step the set of mono-phone HMMs will be converted to a context dependent triphone model by converting the mono-phone transcription of the data to triphone transcription and creating a set of triphone models by copying the mono-phones. Then the model is retrained to tune the triphone model. For example, the mono-phone transcription of the file SI1837.lab,

```
sil hh er sp hh ah m sp b ih k ey m sp ah sp g er g
ah l sp ah v sp s er p r ay z sil
```

will be converted to its equivalent triphone transcription,

```
sil hh+er hh-er sp hh+ah hh-ah+m ah-m sp b+ih b-ih+k
ih-k+ey k-ey+m ey-m sp ah sp g+er g-er+g er-g+ah g-
ah+l ah-l sp ah+v ah-v sp s+er s-er+p er-p+r p-r+ay
r-ay+z ay-z sil
```

This style of triphone transcription is referred to as ***Word Internal Transcription* (WIT)**. Note that some diphones are also generated as a result of word boundaries marked by *sil* and *sp*, because the context reduces to only two phonemes next to the word boundaries. For example, the mono-phone transcription of "`sil hh er sp`" is converted to "`sil hh+er hh-er sp`"; *sil* marks the beginning of an utterance and *sp* marks the boundary of the words. The '-' and '+' represent predecessor and successor respectively. For example, `hh+er` means the *hh* phoneme followed by *er*, and `hh-er` means the phoneme *er* that is preceded by the phoneme *hh*.

This conversion continues for all training data, and in the next step all combinations of triphones and diphones appearing in the transcription are created by duplicating the model of each corresponding mono-phoneme model. For example, a triphone model `s-er+p` will be created by duplicating the model of *er* in the HMM file.

The diphones are duplicated by following the predecessor and successor signs as explained earlier. For example, the `hh+er` will be created by duplicating the model of `hh`, and `hh-er` will be created by duplication of the model of `er`.

There is another triphone transcription known as **Word External Transcription (WET)**. In this transcription, the word boundaries are not marked, or if they are marked, they are neglected. For example, the word external transcription of the file SI1837.lab without marking word boundaries is

```
sil hh+er hh-er+hh er-hh+ah hh-ah+m ah-m+b m-b+ih b-
ih+k ih-k+ey k-ey+m ey-m+ah m-ah+g ah-g+er g-er+g er-
g+ah g-ah+l ah-l+ah l-ah+v ah-v+s v-s+er s-er+p er-
p+r p-r+ay r-ay+z ay-z sil
```

As the final step, the converted transcriptions and the MFCC equivalent of each utterance are used to retrain the triphone models, exactly the same way as described for the mono phoneme model in section (4.3.1).

4.4 Evaluation Method

The final goal of modeling and training is recognition. To recognize an utterance, the preprocessed speech, along with the HMM and language model, are needed by the recognizer. For this project I have employed a Token-Passing method, with one token per model, and a parallel word-network as a language model with an equal probability for each word, as described in Figure 3.3. For this section I have used the HVite tool, provided by HTK. The correctness and accuracy of the recognition is evaluated by comparing the recognized speech with the true transcription of the speech. For correctness and accuracy, the following formulas have been defined.

$$Correctness = \frac{N - D - S}{N} * 100\%$$

$$Accuracy = \frac{N - D - S - I}{N} * 100\% \ .$$

N is the total number of labels in the reference transcription, D is the number of deletion errors, S is the number of substitution errors, and I is the number of insertion errors. For example, if the true transcription is

AGAIN THESE BLOCKS WERE SET **IN** RESIN SATURATED **GLASS** CLOTH AND NAILED.

and the recognised text is

AGAIN THESE BLOCKS WERE SET **INTO THE** RESIN SATURATED CLOTH AND NAILED.

there is one substitution error, **INTO** for **IN**, one insertion error, **THE**, and one deletion error, **GLASS**, in the recognised text, then

$$Correctness = \frac{12-1-1}{12} * 100\% = 83.33\%$$

$$Accuracy = \frac{12-1-1-1}{12} * 100\% = 75.00\%$$

4.5 Experiments

This experiment focuses on the modeling and training process, the most important part of developing a speech recognition system. Correctness and accuracy, however, also depend on the *word-network* that is provided to the recognizer (the bigger the network the less correctness and accuracy). In spite of the importance of the selection of the word-network, network selection plays a secondary role in accurate recognition. The most important component, and in fact the most challenging part, is the modeling and training.

4.5.1 Experiment conditions

In all the following experiments the models are trained with male utterances of all SI and SX training data, provided by the TIMIT database (equal to 2608 utterances), plus Keith's training speech (97 utterances), for a total of 2705 sentences and 4170 distinct words. The recognition data, selected from Keith's speech, is grouped into two categories, **Train-Data** and **Test-Data**. Train-Data is the data used in the training process and employed for recognition too, but the Test-Data is the speech not used in the training process and is employed only for recognition. The language model is selected as a parallel word-network from all words (550 distinct words) that appear in both Keith's Train-Data and Test-Data.

4.5.2 Mono-Phoneme

In the following experiments the HMM for each phoneme listed in Table 1.1 is created and trained with TIMIT and Keith's training data, after which the correctness and accuracy of the model is tested by recognizing both Keith's Train- and Test- Data.

4.5.2.1 Training without SP model

In this experiment, the models are trained without inserting the *sp* model between the word boundaries. For example, the utterance "`Her hum became a gurgle of surprise`" selected from file SI1837.lab (SI training list of TIMIT), is converted to the following phonetic equivalent:

```
sil hh er hh ah m b ih k ey m ah g er g ah l ah v s
er p r ay z sil
```

Table 4.1 shows the number of iterations used for training and the correctness and accuracy of the recognition.

Number of training iterations	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	9.15	9.09	-45.07	-17.17
2	20.42	23.23	-95.07	-55.56
3	21.83	27.27	-53.52	-26.26
4	**27.46**	27.27	-40.11	-22.22
5	26.76	27.27	-35.51	-16.16
6	25.35	**28.28**	-35.92	**-13.13**
7	24.65	28.28	-35.92	-13.13
8	25.35	27.27	-35.92	-14.14
9	24.65	27.27	-34.51	-14.14
10	24.65	27.27	**-33.80**	-14.14

Table 4.1 Accuracy and correctness of a mono-phoneme model without *sp* as word boundaries.

As shown in Table 4.1, increasing the number of iterations does not increase the correctness and accuracy of the model as may be expected. In fact, after a few iterations the model reaches a condition known as ***over-training***, after which the model is corrupted, and the correctness and accuracy of the model decreases, employing more training iterations. For this model iteration 6 is optimum. After that the accuracy and correctness of the model decreases. It is worth mentioning that for each training iteration all 2705 utterances of TIMIT and Keith's Train-Data are employed.

4.5.2.2 Training With sp Model

In this experiment the system is trained by inserting the *sp* model between the word boundaries. For example, the utterance in the previous experiment is converted to the following phonetic equivalent:

```
sil hh er sp hh ah m sp b ih k ey m sp ah sp g er g
ah l sp ah v sp s er p r ay z sp sil
```

Table 4.2 shows the number of iterations used for training and the correctness and accuracy of the model for recognition.

Number of training iterations	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	7.04	9.09	-38.73	-14.14
2	21.83	24.24	-96.48	-59.60
3	22.54	26.26	-57.04	-27.27
4	24.65	25.25	-42.96	-24.24
5	24.65	26.26	-38.73	-18.18
6	24.65	26.26	-36.62	-15.15
7	25.35	27.27	-35.92	**-14.14**
8	**26.06**	**28.28**	-33.80	-15.15
9	25.35	27.27	-30.99	-15.15
10	25.35	27.27	-30.99	-15.15

Table 4.2 Accuracy and correctness of the mono-phoneme model with *sp* as word boundaries.

As indicated in Table 4.2, the best results occur in iterations 7 and 8. Another important fact is that the accuracy and correctness of this model, with *sp* between word boundaries, do not differ from the accuracy and correctness of the previous model, without *sp* as word boundaries.

4.5.2.3 Compound Model

In this experiment, the five first iterations are trained exactly as described in section 4.5.2.1, without *sp* as word boundaries. Then the transcription is modified by inserting *sp* as word boundaries (section 4.5.2.2), and the training process continues for five more iterations. Table 4.3 shows the number of iterations used for training and the correctness and accuracy of the recognition. The first five rows are omitted since they are exactly the same as in Table 4.1.

Number of training iterations	Correctness %		Accuracy %	
	Train	Test	Train	Test
6	**26.76**	**28.28**	**-32.39**	**-13.13**
7	26.76	26.26	-33.80	-16.16
8	26.76	26.26	-33.10	-16.16
9	26.26	24.24	-31.69	-18.18
10	26.06	24.24	-31.69	-18.18

Table 4.3 Accuracy and correctness of compound model.

As shown in Table 4.3, the model is not improved with this method and the same scenario recurs when the number of training iterations increases.

4.5.3 triphone

The following experiments further the experiments in the previous section. In these experiments, the models created by the mono-phoneme model in iteration 6 of Tables 4.1 and 4.2 are converted to the triphone model by making copies of the models, as described in section (4.3.2). Then, the mono-phoneme transcriptions are also converted to the triphone equivalent, and the models are retrained. The training and test data and word-network are exactly the same as in the previous section, described in 4.5.1

4.5.3.1 Training with Word External Transcription

In this section, the trained model of iteration 6 of section 4.5.2.1(The Training Without *sp* Model) is converted to the triphone model and retrained without *sp* forming the boundaries of words. For example, the mono-phoneme transcription of utterance "`Her hum became a gurgle of surprise`", will be converted to the following equivalent triphone transcription from iteration 7:

```
sil hh+er hh-er+hh er-hh+ah hh-ah+m ah-m+b m-b+ih b-
ih+k ih-k+ey k-ey+m ey-m+ah m-ah+g ah-g+er g-er+g er-
g+ah g-ah+l ah-l+ah l-ah+v ah-v+s v-s+er s-er+p er-
p+r p-r+ay r-ay+z ay-z sil
```

Table 4.4 shows the number of iterations used for training and the correctness and accuracy of the model for recognition.

Iteration of re-training with triphone model	Correctness %		Accuracy %	
	Train	Test	Train	Test
$6^7 + 1$	49.03	24.24	**16.09**	**-2.02**
6 + 2	**49.30**	**25.25**	15.49	-9.09
6 + 3	49.30	23.23	14.08	-9.09
6 + 4	49.30	22.22	15.49	-9.09
6 + 5	48.59	24.24	18.31	-6.06

Table 4.4 Accuracy and correctness of the triphone model trained with WET.

As indicated in Table 4.4, both the correctness and accuracy of the model is improved when compared with the correctness and accuracy obtained with the mono-phoneme model, tested with the same data in the same conditions.

4.5.3.2 Training with Word Internal Transcription

In this section, the trained model of iteration 6 of section 4.5.2.2 (The Training With *sp* Model) is converted to a triphone model and retrained with *sp* as the boundary of words. For example, the mono-phoneme transcription of utterance "`Her hum became a gurgle of surprise`", is converted to the following Word Internal Transcription:

```
sil hh+er hh-er sp hh+ah hh-ah+m ah-m sp b+ih b-ih+k
ih-k+ey k-ey+m ey-m sp ah sp g+er g-er+g er-g+ah g-
ah+l ah-l sp ah+v ah-v sp s+er s-er+p er-p+r p-r+ay
r-ay+z ay-z sp sil
```

The Table 4.5 shows the number of iterations used for retraining and the correctness and accuracy of the model for recognition.

Iteration of re-training with triphone model	Correctness %		Accuracy %	
	Train	Test	Train	Test
6 + 1	67.61	31.31	50.00	4.04
6 + 2	69.01	**34.34**	51.41	**4.04**
6 + 3	**70.42**	32.32	**52.11**	0
6 + 4	69.72	32.32	48.59	3.03
6 + 5	68.31	31.31	46.48	0

Table 4.5 Accuracy and correctness of triphone model trained with WIT.

As indicated in Table 4.5, the correctness and accuracy of this model is much better than the accuracy and correctness of the triphone model trained with WET.

4.5.4 Tied-Model

The model discussed in the previous section suffers from low performance in recognizing Keith's Test-Data, compared with its relatively high performance in recognizing his Train-Data. This is because the previous model is not tailored suitably for data not provided in the training list. To solve this problem, we employ an algorithm known as *Tree-Based Clustering* [Appendix C] to classify, and tie the triphone to reduce the number of parameters, so the remaining parameters can be estimated more robustly. The model created is called the **tied-model**, and the previous models are now known collectively as the **untied-models**.

4.5.4.1 Training Tied-Model

The last model, the model trained with WIT, provided the best accuracy and correctness compared with the other models. As a result, the next experiment will be built on top of this model. In this experiment, the result of iteration 9 of Table 4.5 is converted to the *tied-triphone model* and retrained. The data and word-network are exactly the same as described in section (4.5.1).

It is worth mentioning that the transcriptions provided for training the tied-triphone model and triphone model are not different, and that only the parameters of the HMMs are tied to create a compact model. The transcription provided for training the

[7] 6 indicates that this iteration begins from the 6th iteration of the referenced experience, section (4.4.1.3).

tied-triphone model is WIT. Table 4.6 shows the number of iterations used for retraining and the correctness and accuracy of the recognition.

Number of re-training iterations with tied-triphone model	Correctness %		Accuracy %	
	Train	Test	Train	Test
$9^{8}+1$	31.69	21.21	-21.13	-20.20
9 + 2	50.70	47.47	9.15	14.14
9 + 3	54.93	47.47	18.31	14.14
9 + 4	**55.63**	**47.47**	**19.01**	**14.14**
9 + 5	55.63	47.47	18.31	13.13

Table 4.6 Accuracy and correctness of tied-triphone model trained with WIT.

As indicated in the Table 4.6, the correctness and accuracy of recognizing the Train-Data decreases, while the correctness and accuracy of recognizing the Test-Data increases. For a general recognition system, such as a dictation program, it is not possible to provide all training data to cover all possible utterances of recognition, so it is clear that for such a system, a tied-triphone model is a better choice compared with a triphone model. However, for systems that will be used for recognizing limited utterances, the triphone model seems superior compared to the tied-triphone model[9]. Another difference between the tied-triphone models and triphone models is size. We will discuss this in the next section.

4.6 Comparison

Figure 4.3 compares the correctness and accuracy of the recognition of Keith's train-data with mono-phoneme models (with *sp*, without *sp*, and a compound model). Figure 4.4 shows the same experiments with Keith's test-data, the data that was not used in training.

---

[8] 9 indicates that this iteration begins from the 9th iteration of the referenced experience, section (4.4.2.3).

[9] These experiments are conducted with only 2705 utterances. In the case of greater availability of training data, however, the conclusions may be different; this possibility has yet to be explored.

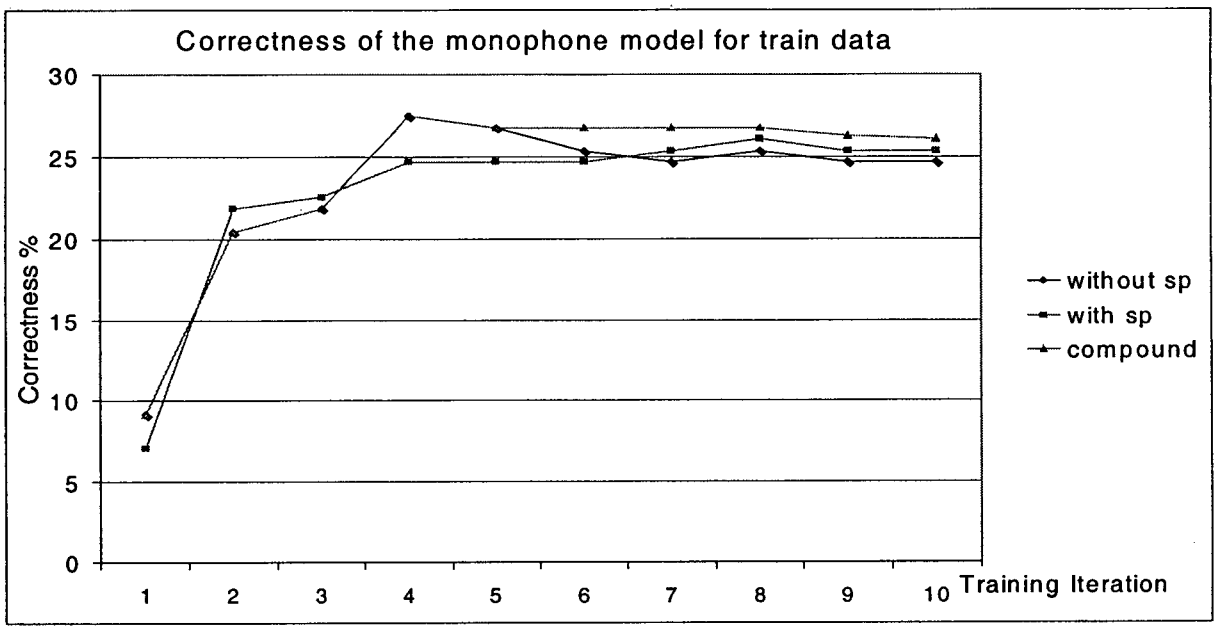

Figure 4.3 The correctness of the mono-phone model recognizing *Train data.*

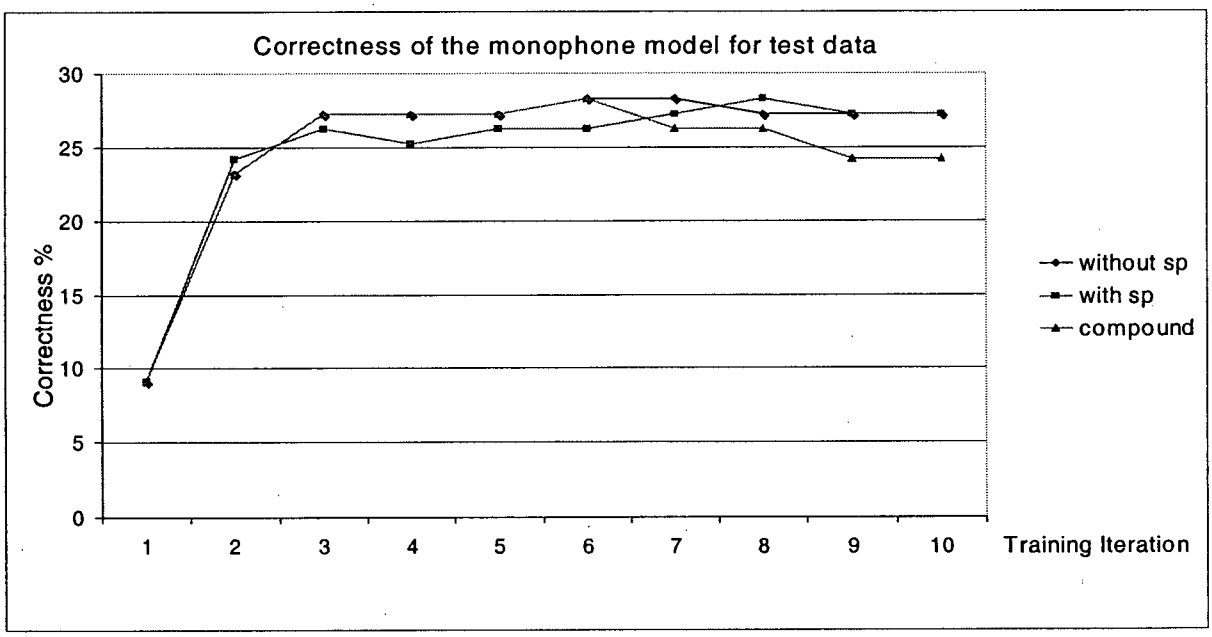

Figure 4.4 The correctness of the mono-phone model recognized *Test data.*

Both figures indicate that increasing the number of training iterations does not improve the model parameters as may be expected; instead it reveals the fact that extra

training may even decrease the correctness and accuracy of the models. Furthermore, both figures indicate that there are no significant differences in correctness between the three mono-phoneme models. The same result is achieved for accuracy in the three mono-phoneme models, and there is no significant difference between them.

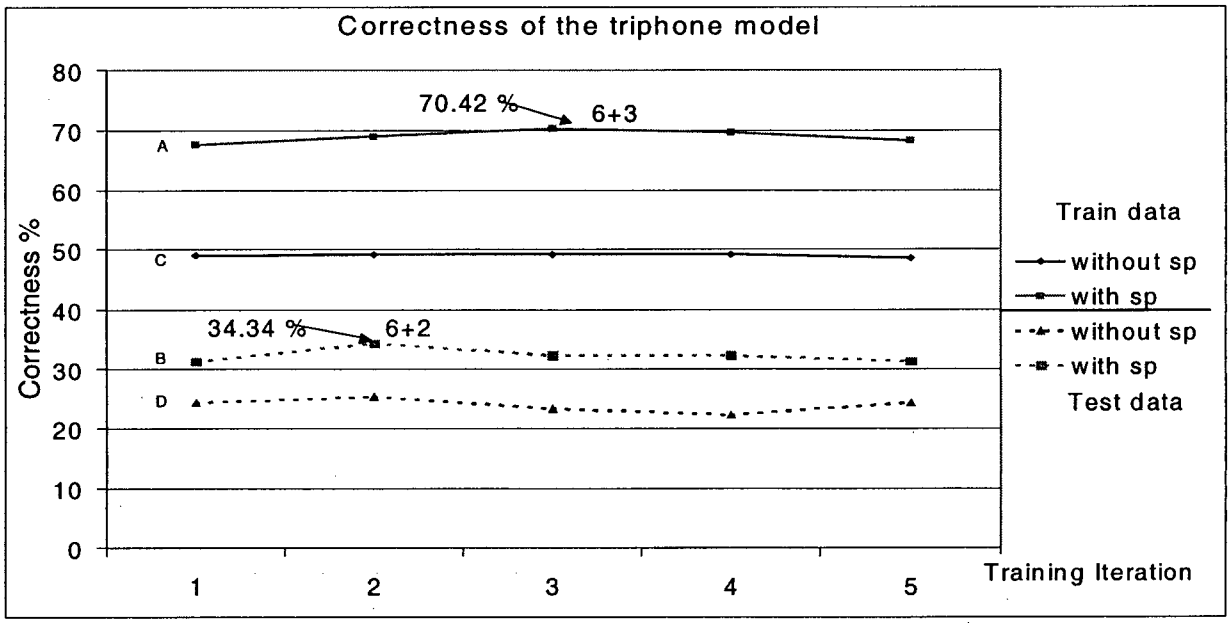

Figure 4.5. The correctness of triphone model in recognizing train and test

To achieve better accuracy, the mono-phoneme model is converted to a triphone model and retrained from iteration 6 of the mono-phoneme models. Figure 4.5 shows the correctness of the triphone models in recognizing the train and test data.

As indicated in Figure 4.5, the correctness of recognition is different between the two models trained with WIT and WET. Furthermore, the correctness of recognizing both train-data and test-data is significantly improved in the model trained with WIT. As indicated in the figures, the model has achieved 70.42% correctness in recognizing train-data, and 34.34% correctness in recognizing test-data, which shows an improvement compared with precious models.

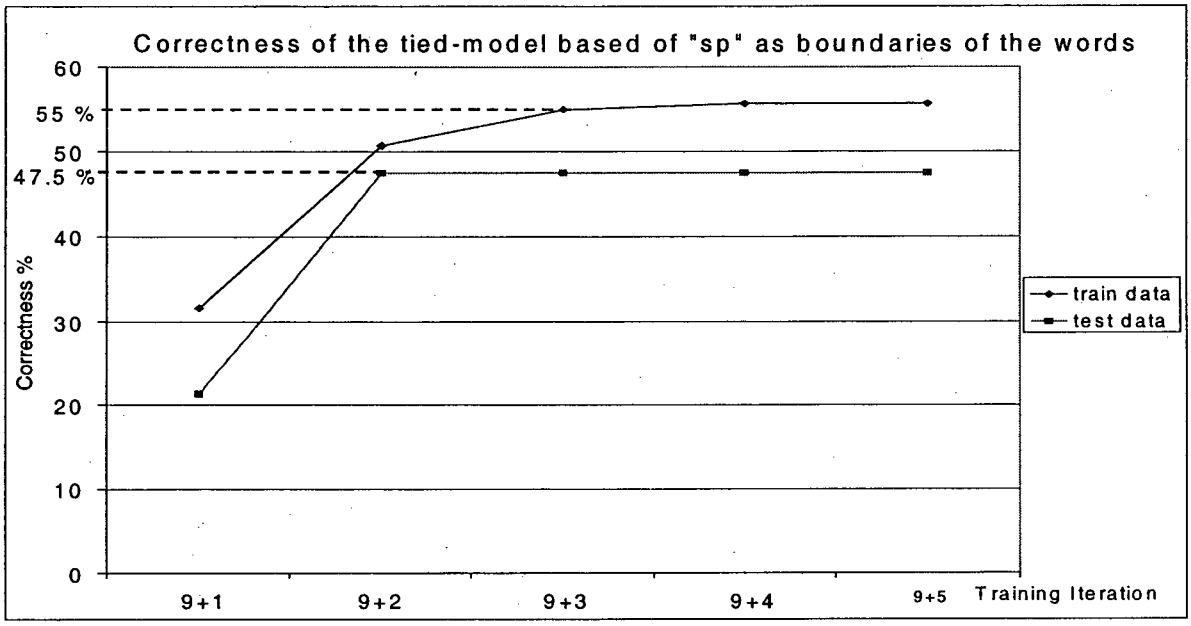

Figure 4.6. The correctness of the tied-model in recognizing train and test data.

Figures 4.6 and 4.7a shows the correctness of the tied-triphone model. The model is converted to a tied-triphone model from iteration 9 of the triphone model trained with WIT (Table 4.5), then the system is retrained. Comparing Figures 4.5 and 4.6 indicates that although the correctness of recognition of the train-data decreases from 70.42% in the triphone model to 55% in the tied-triphone model (-15%), the correctness of recognition of the test data increases from 34.34% in the triphone model to 47.5% in the tied-triphone model (+13%).

The same differences can be seen when comparing the accuracy of the two models. For example, Figure 4.7b compares the accuracy of the triphone model and the tied-triphone model for both train and test data. As indicated in the bar-charts, the accuracy in recognizing the train-data decreases from 52.11% in the triphone model to 19.01% in the tied-triphone model, while the accuracy of recognizing the test data increases from 4.04% in the triphone model to 14.14% in the tied-triphone model.

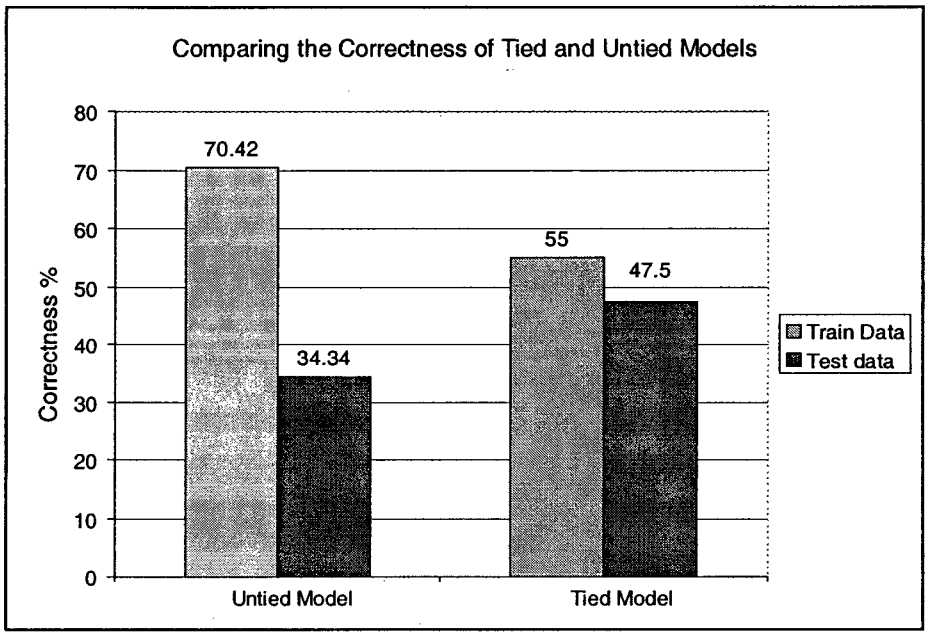

Figure 4.7a Comparing the Correctness of the triphone model to the tied-model.

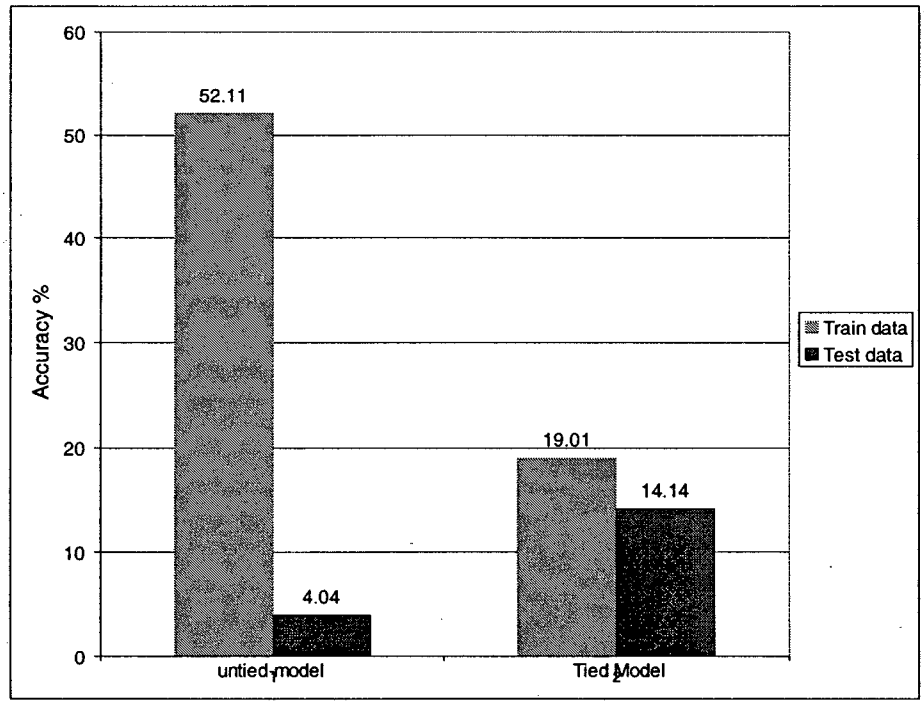

Figure 4.7b Comparing the accuracy of the triphone model to the tied-model.

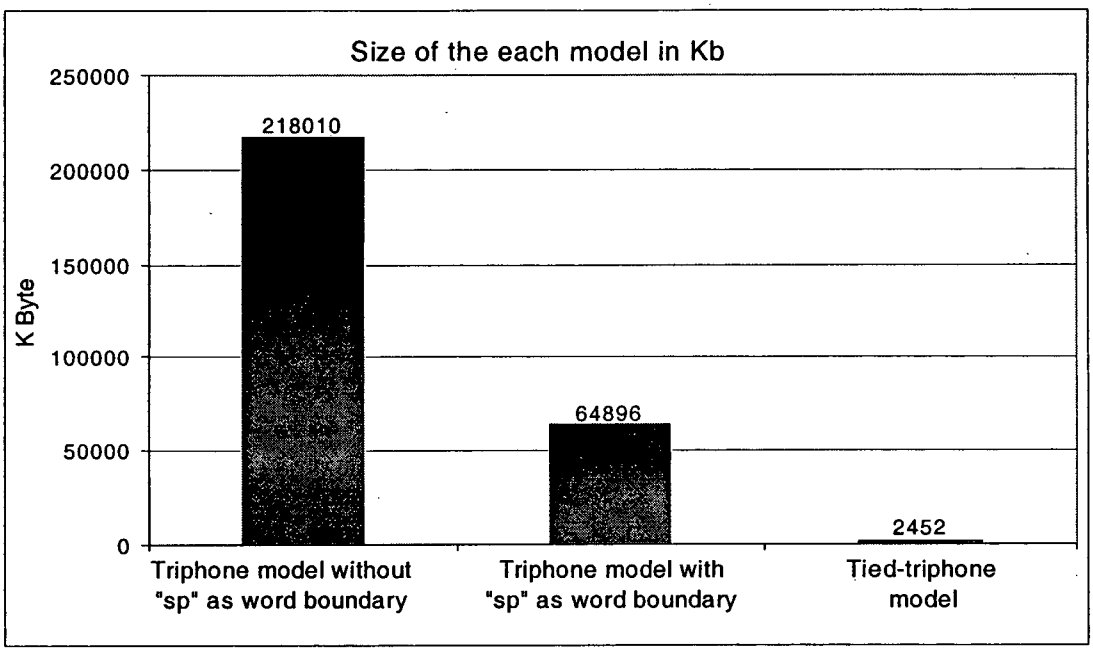

Figure 4.8 The size of the models in k bytes.

The size of the memory needed by each model is also different. Figure 4.8 compares the size of the models in Kbytes for each model. As indicated in Figure 4.8, the size of the model trained with WET is about 218 Mb, while the size of models trained with WIT and tied-triphone model are about 64Mb and 2.5 Mb respectively. As a result, while the correctness and accuracy of the system increases for recognizing test data with a tied-triphone model, the memory needed for the tied-triphone model decreases.

Chapter 5

Adaptation

The models developed so far are known as *user-independent* systems in that the models are trained with the utterances of 439 different speakers from different regions of the United States. To increase the correctness and accuracy of the system, the model should be converted to a *user-dependent* system by adjusting the parameters of the model to the voice of a specific user. There are two methods for adjusting the parameters of HMM to a specific user, known as the direct method and the adaptation method. The direct method involves training the model with just a single speaker's speech, the speaker with whom the system will be tested. Training the model, however, requires a lot of training data that may not be available in most cases. For example, in this project I have used 2705 different utterances to train the HMMs. As a result, I have adapted an alternative method known as an adaptation method, discussed in section (3.10).

5.1 Triphone adaptation

The model discussed in section (4.5.3.2) is being adapted with Keith's train data from iteration 8 (refer to table 4.5). In this experiment I used a regression class tree with 32, 64, 96, and 128 nodes to classify the acoustic models [Appendix A]. Then the models are adapted for a maximum of 4 iterations. The results of the correctness and accuracy of recognition in each model, for both test and train data, are shown in Table 5.1.

Adapted with 32 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	**81.69**	**43.43**	**70.42**	**10.1**
2	80.99	41.41	68.31	5.05
3	81.69	41.41	69.72	7.07
4	81.69	41.41	69.72	7.07
Adapted with 64 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	81.69	**43.43**	**71.13**	**9.09**
2	82.39	41.41	69.72	5.05
3	82.39	41.41	70.42	6.06
4	**83.10**	39.39	71.13	4.04
Adapted with 96 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	81.69	43.43	71.13	**9.09**
2	82.39	41.41	69.72	5.05
3	**83.10**	**41.41**	**71.13**	6.06
4	83.10	39.39	71.13	4.04
Adapted with 128 nodes				
Iteration	Correctness %		Accuracy $	
	Train	Test	Train	Test
1	81.69	43.43	**71.13**	**9.09**
2	82.39	41.41	69.72	5.05
3	**83.10**	**41.41**	71.13	6.06
4	83.10	39.39	71.13	3.03

Table 5.1 The correctness and accuracy of the adapted data for the triphone model.

As indicated in the table, increasing in the number of adaptation iterations does not necessarily increase the correctness and accuracy of the model; the model seems adapted after third iteration, and the accuracy and correctness of the model do not change. Table 5.1 also indicates that there is no significant difference in recognition achieved by selecting different nodes for the *regression class tree*. However, close inspection of Table 5.1 shows that the tree with 32 nodes shows slightly better results for the test data, while the tree with more nodes has better output for train data. This is because more acoustic information will result in better classification of the regression tree. Therefore, if we

provide more training data to cover more test space, the tree with more nodes will provide better acoustic classification. However, if the test space is too huge to be covered by the training data, the tree with less nodes will provide better classification results. The selection of the number of nodes for the regression class tree will depend on the availability of training data.

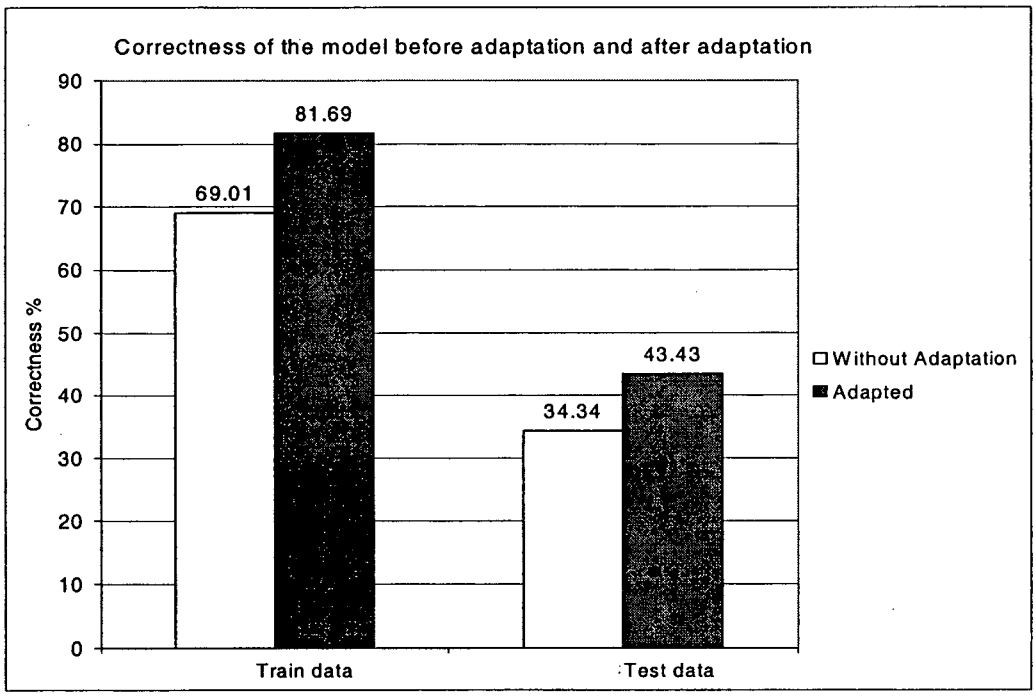

Figure 5.1 The correctness of the model before and after adaptation.

Figure 5.1 compares the correctness of the model before and after adaptation for the model with a regression tree with 32 nodes. As indicated in the figure, the correctness of the model increased from 69.01% to 81.89% in recognizing train-data, and from 34.34% to 43.43% for recognizing test-data.

5.2 Tied-model adaptation

The same adaptation process is repeated for the tied-triphone model, and the results are indicated in Table 5.2. The model is adapted from iteration 9+4 in section 4.5.4.1 (Training Tied-triphone model) table 4.6.

Adapted with 32 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	78.87	52.53	61.27	29.29
2	79.58	51.52	64.08	29.29
Adapted with 64 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	78.87	48.48	60.56	29.29
2	80.28	49.49	64.08	29.29
Adapted with 96 nodes				
Iteration	Correctness %		Accuracy 5	
	Train	Test	Train	Test
1	78.87	48.48	60.56	29.29
2	80.28	49.49	64.08	29.29
Adapted with 128 nodes				
Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	78.87	48.48	60.56	29.29
2	80.28	49.49	64.08	29.29

Table 5.2 Correctness of the adapted tied-triphone model.

Figure 5.2 compares the correctness of the tied-triphone model before and after adaptation. As indicated, the correctness of recognizing train-data is increased from 55.63% to 78.87% after adaptation, and the correctness of recognizing the test-data is increased from 47.47% to 52.53% after applying adaptation.

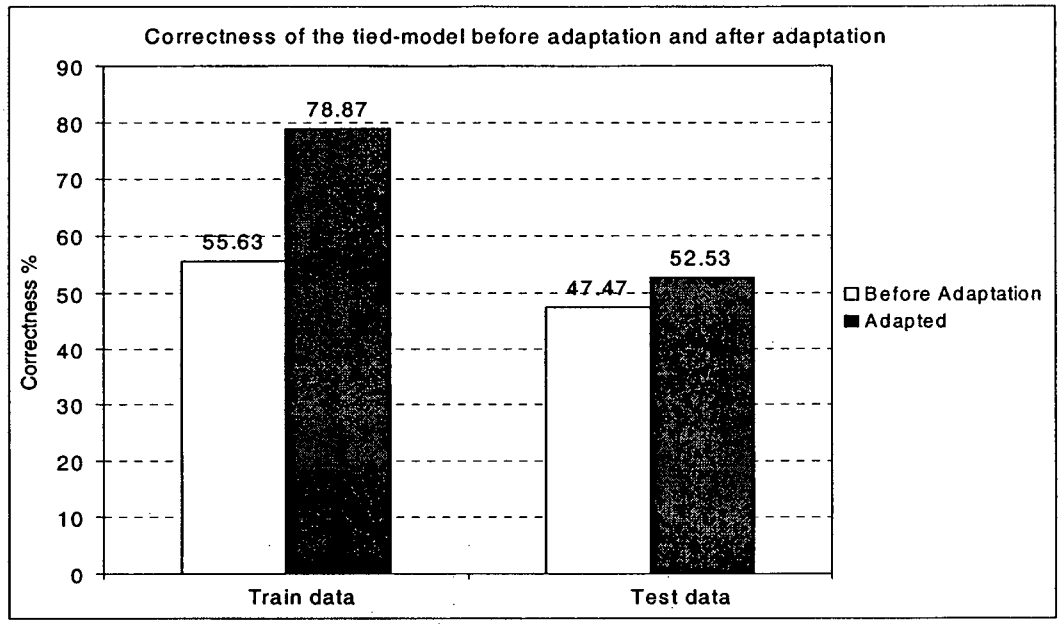

Figure 5.2 The correctness of the tied-triphone model before and after adaptation.

5.3 Retraining

I have selected the first iteration of the adapted model with 32 nodes of the triphone model and tied-triphone model, and retrained them with Keith's train-data. Then the model is tested by recognizing Keith's test and train data, as in previous experiments.

Table 5.3 shows the correctness and accuracy of recognition of the triphone model retrained after adaptation. Table 5.3 indicates that the best result is achieved in the 4th iteration. Comparing this result with the results achieved before retraining (Table 5.1) indicates that although the correctness and accuracy for recognizing train-data is

Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	95.07	40.4	88.03	-2.02
2	96.48	42.42	88.73	1.01
3	96.48	42.42	89.44	5.05
4	**96.48**	**42.42**	**90.14**	**5.05**
5	96.48	42.42	90.14	3.03

Table 5.3 Correctness and accuracy of the triphone model (triphone) retrained after being adapted.

increased, the same parameters are decreased for recognizing test-data. Figure 5.3 compares the correctness and accuracy of the model before and after retraining.

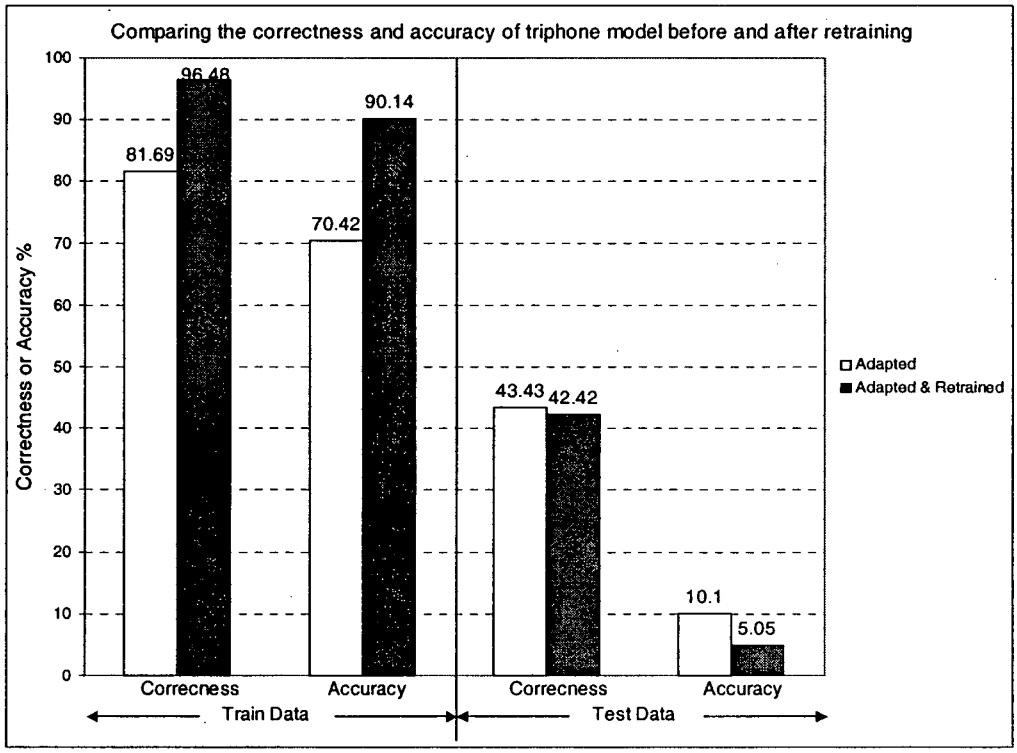

Figure 5.3 The correctness and accuracy of the untied-triphone model before and after retraining.

Figure 5.3 shows that both the correctness and accuracy of the model in recognizing the train-data increases with the model retrained after adaptation, while the same parameters decrease for recognizing the test-data with the same model.

Table 5.4 shows the correctness and accuracy of the tied-triphone model, retrained after being adapted, and Figure 5.4 compares the correctness and accuracy of the model

Iteration	Correctness %		Accuracy %	
	Train	Test	Train	Test
1	91.55	53.54	80.99	30.3
2	92.95	52.53	83.1	32.32
3	92.25	52.53	83.8	31.31
4	**92.25**	**52.53**	**84.51**	**32.32**
5	92.25	51.52	84.51	28.28

Table 5.4 Correctness and accuracy of the tied-triphone triphone model retrained after adaptation.

before and after retraining. As indicated in Figure 5.4, the model is improved for recognizing training and test data after retraining. Although there is no significant improvement for recognizing training data, the result is better than in the untied triphone model.

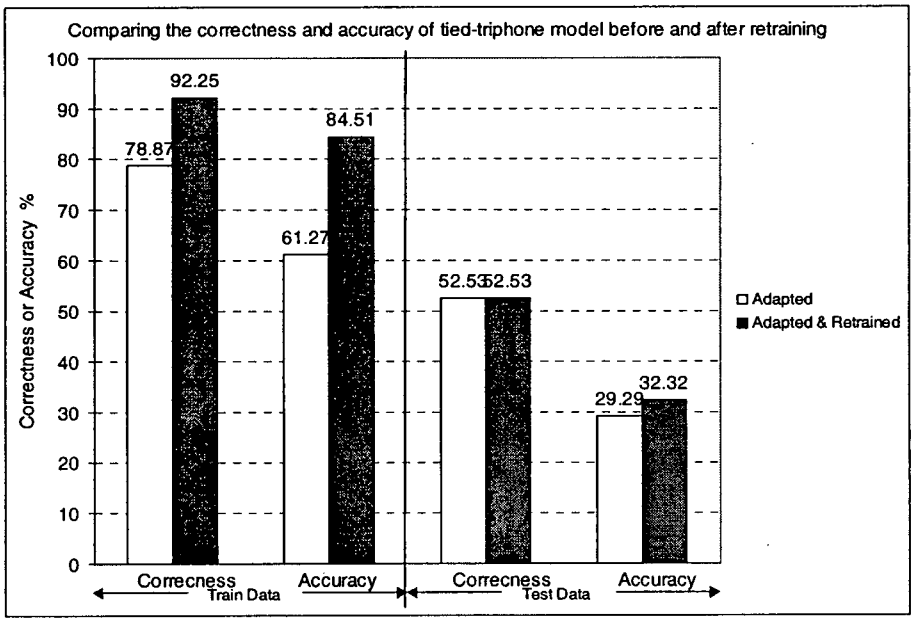

Figure 5.4 Accuracy and??correctness of the tied-triphone model before and after retraining.

5.4 Comparison

The bar-charts in Figures 5.3 and 5.4 indicate that the models are improved when retrained after adaptation, except for the recognition of test data in the untied-triphone model. The reason may be an insufficient amount of retraining data and number of triphone models in the untied-triphone model. If we had more of Keith's training data instead of only 97 utterances, then the values of the chart might be different from the ones shown in Figure 5.3.

Another fact we may notice when comparing the two charts, is the superiority of the tied-triphone model compared with the triphone model. Both the correctness and accuracy in recognizing the test-data are better than with the untied-triphone model (gaining 10% to 25%). However, the correctness and accuracy of the model obtained for recognizing train-data in the tied-triphone model is slightly less than the correctness and accuracy of recognizing the same data in the triphone model (losing 3.5% to 5.5%), but the memory needed for the tied-triphone model is only 2452/64896 = 3.8% of the memory needed for the triphone model. Comparing the loss of a maximum of 3.5% to 5.5% in correctness and accuracy, with saving 96.2% of memory in the tied-triphone model, reveals the advantage of the tied-triphone model.

Chapter 6

Diphone & Word Segmentation and Extraction for Natural Speech Synthesis

In this section, I have used the speech recognizer developed in the previous sections to create a program to demonstrate the functionality of the system. This program has many capabilities, such as diphone and word segmenting and word spotting in a multimedia file. The diphone and word segmenting programs can be employed to prepare the sub-words needed by a natural speech synthesizer. A natural speech synthesizer can recreate the speech of a person by concatenating words and sub-words, to create a talking machine. A word-spotting program can be employed to search a multimedia file for an utterance of a word.

This chapter begins by introducing the methods developed to segment the words and sub-words and discuss the accuracy and correctness of the segmentation. It then continues by demonstrating the speech synthesized with diphones segmented using different methods, and discusses the accuracy of the synthesized speech. Next, a very important functionality of the program for finding an utterance in a media file is demonstrated.

6.1 Diphone Segmentation

Diphones are sub-word elements mainly used in speech synthesizers. As mentioned in the first chapter of this thesis, segmenting of diphones is a semi-manual

process. This section of the thesis attempts to employ the speech recognizer developed in the previous section to segment speech into diphones automatically.

To segment speech into diphones, I have developed two methods, an *indirect method* and a *direct method.* In the indirect method, the diphones are segmented from the middle node of a triphone to the middle node of the adjacent triphone. In the direct method, the speech recognizer is modeled and trained by diphones instead of by triphones, and the speech recognizer recognizes diphones directly. In both methods, we assume that both speech and the equivalent text are available for segmentation.

6.2 Indirect Method

In this method, first the triphones are recognized. Then the program segments

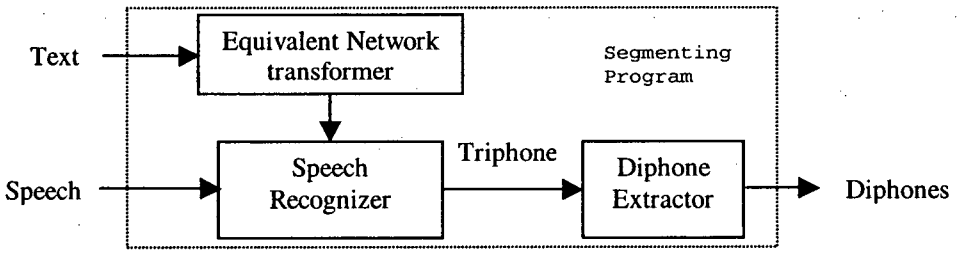

Figure 6.1 Block diagram of Diphone segmentation.

triphones into diphones by segmenting the middle node of one triphone to the middle node of its adjacent triphone. As shown in Figure 6.1, the speech and its equivalent text are provided to the *segmenting-Program.* The program transforms the text into equivalent word-networks and provides the speech and the word-network to the recognizer. The recognizer recognizes the provided text uttered by the speaker; then the recognized triphones are passed to the diphone extractor to be segmented into diphones, as described in section 1.3.1. Finally, the output is stored in a database to be accessed by a speech synthesizer program.

The model used for recognition is the tied-triphone model without adaptation, trained with WIT and *sil* at the beginning and ending of the transcription. The appearance of *sil* and *sp* in the output are not mandatory; it depends on whether the recognizer finds frames that stand for *sp* or *sil*. As a result, there are many possible output transcriptions for the speech, even though the equivalent text is provided for the module. The total possible output for each utterance is equal to the permutations of *sp* and *sil* in the transcription.

6.2.1 Evaluation

To evaluate the accuracy of the diphones segmented with this method, first the accuracy of segmenting the triphones is studied. For this purpose, I simply provide the utterance available in the TIMIT database to the program and compare the segmented results with the segmentation provided by the TIMIT for the same utterance. However, there is no method to evaluate the correctness and accuracy of the segmented phonemes. To address this, I introduce two comparison methods.

$$Accuracy\% = \frac{2 \times \Delta T_{intersection}}{\Delta T_{reference} + \Delta T_{recognized}} \times 100$$

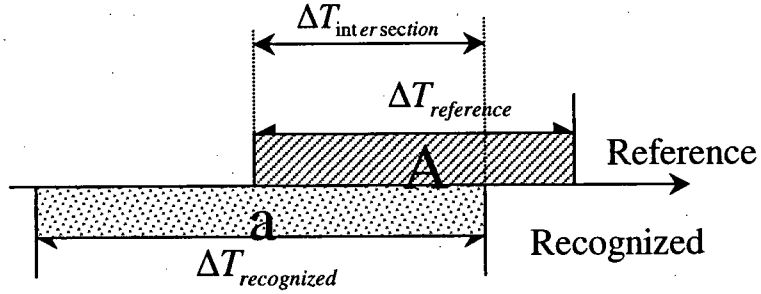

Figure 6.2 Calculating the accuracy of segmentation.

6.2.2 Method I

The accuracy of segmentation is set to the ratio of the intersection of the recognized segments and reference segments, TIMIT, as shown in Figure 6.2.

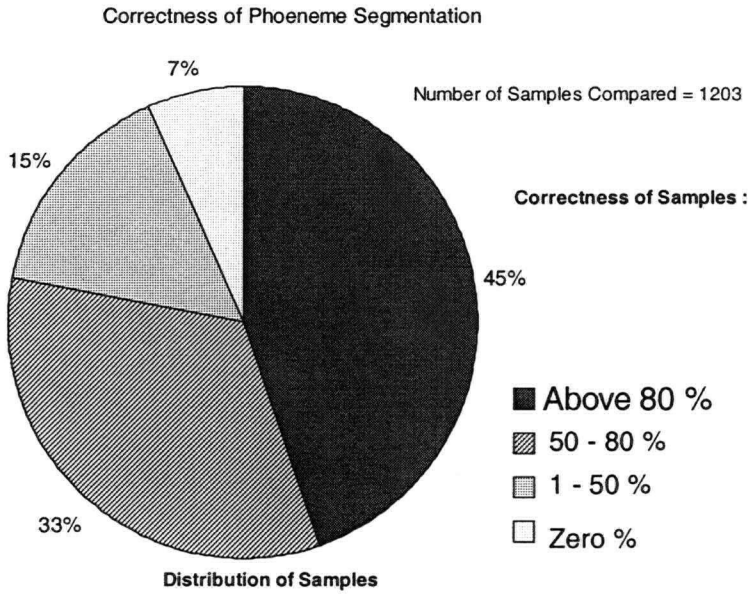

Figure 6.3 Distribution??and accuracy of segmented phoneme samples.

Figure 6.3 shows the results of the experiments that calculated the accuracy of the phonemes segmented with this formula. As indicated in Figure 6.3, 45% of the phonemes are segmented with an accuracy above 80%; 33% of the phonemes are segmented with an accuracy of between 50% and 80%, and 15% of the phonemes are segmented with an accuracy of less than 50%. The pie chart indicates that 7% of the phonemes are segmented with zero accuracy, meaning there is no intersection between the segmented and reference phonemes. The accuracy evaluated here is the result of the recognition model without adaptation, so the result is expected to improve with a model adapted for the speech of a specific user.

6.2.3 Method II

In the previous method, I have assumed that all parts of a phoneme are equally significant, but studies show that the middle node of a phoneme stands for its steady state has a more important role in identifying a phoneme than the two transient nodes. As a

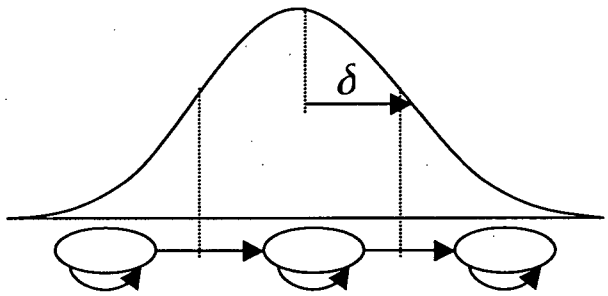

Figure 6.4 Mapping the model to Normal Distribution.

result, the two transient nodes are less significant, when compared to the middle node. Therefore, I have employed another method of evaluating the correctness of a segmented phoneme, by modeling its duration with Normal Distribution.

6.2.3.1 Correctness

In this method, I modeled a phoneme duration with a Normal Gaussian distribution, and compared the correctness of the segmented triphone with the boundary of the same phoneme in the same utterance provided by TIMIT. In this method, I

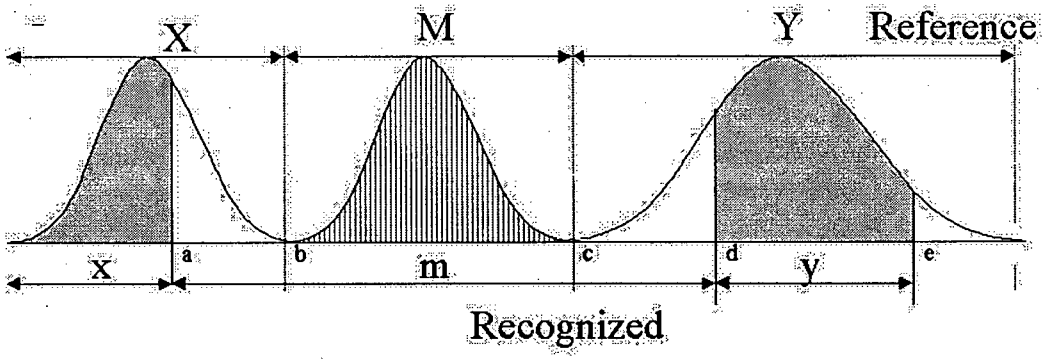

Figure 6.5 An example of calculating the correctness of segmenting phonemes.

considered the fact that the middle node of a phoneme is its most significant part, while the two other nodes, which stand for the transition parts of the phoneme, are its less significant parts. Figure 6.4 shows the mapping method used. As indicated in Figure 6.4, the center point of the curve is mapped to the middle of the phoneme and the standard deviation of the model is set to half the duration of the middle node. Figure 6.5 shows an example of calculating the correctness of the segmentation using this model.

The top line in Figure 6.5 shows the boundaries of the phonemes as provided by TIMIT, and the bottom line shows the boundaries of the recognized phonemes. As indicated in the figure, the segmented phoneme **m** begins from a and ends at d, and the duration of the reference phoneme **M** begins at b, and ends at c, so that $a < b, c < d$. Therefore, the correctness of segmenting phoneme **m**, compared with the reference **M**, is 100%. The correctness of the segmented phoneme **y** in the same utterance is equal to the shaded area of distribution **Y**. However, phoneme **m** is extended from its boundary, compared with the reference, and has overlapped with the two adjacent phonemes. The error is calculated against the correctness of the adjacent phonemes, so the error is considered only once in the calculation. Note that this calculation only shows the correctness of the segmentation and it does not indicate the accuracy of the method. In fact, this method cannot be used to evaluate the accuracy of the segmentation.

Figure 6.6 shows the results of this method. As indicated in the pie-chart, 69% of the segmented phonemes are more than 80% correct. 11% of the samples are between 50 and 80% correct, 14% of the segmented phonemes are less than 50% correct, and finally,

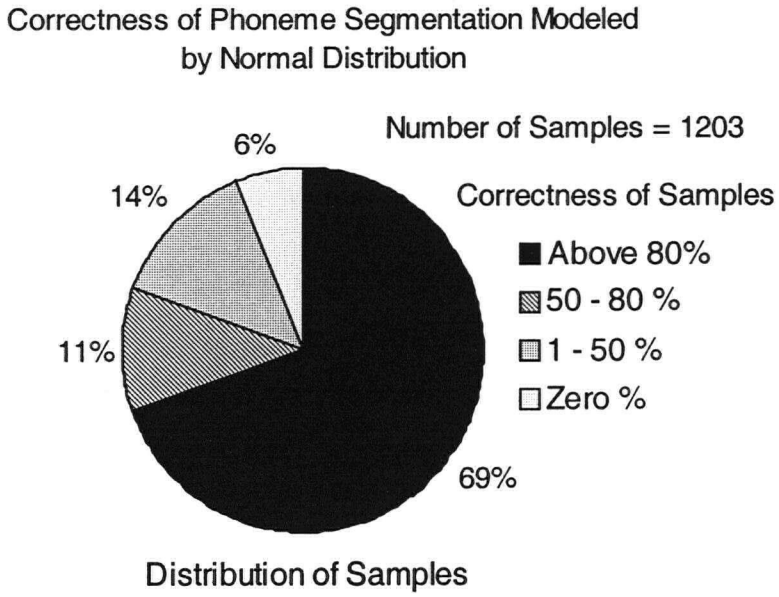

Figure 6.6 Distribution and correctness of segmented samples.

6% of the segmented phonemes are completely outside of the reference boundaries, at zero percent correct.

6.2.3.2 Accuracy

In evaluating correctness, I have discarded the effects of the intersection of the segmented phoneme with its adjacent reference phonemes. For example, the phoneme **m** shown in Figure 6.5 is segmented 100% correctly, even though it is extended beyond its boundaries and overlaps with the **x** and **y** phoneme boundaries. However, this is considered an error in calculating the accuracy of the segmentation. The accuracy of the segmentation is calculated as

$$Accuracy = \frac{Correctness}{Correctness + Left\_Error + Right\_error}$$

In this formula, correctness is calculated from the previous section and

Left_error is equal to the area between ***a*** and ***b***

Right_Error is equal to the area between ***c*** and ***d***

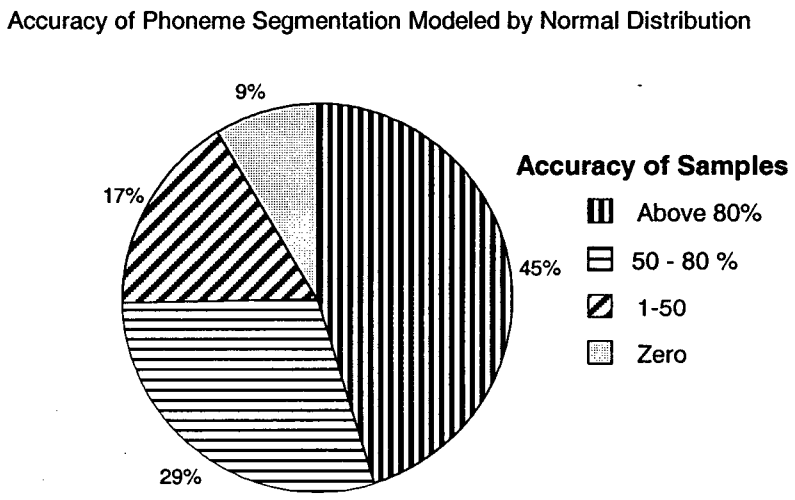

Figure 6.7 Distribution and accuracy of the segmented samples.

Figure 6.7 shows the result of calculating the accuracy with the above formula. As indicated in the figure, 45% of the phonemes are segmented with an accuracy above 80%, 29% of the phonemes are segmented with an accuracy between 50 and 80%, 17% of the phonemes are segmented with an accuracy below 17%, and finally, 9% of the samples are segmented with an accuracy equal to zero.

Comparing Figures 6.7 and 6.3 reveals that the accuracy we have calculated with the two methods are very close. However, the second method shows that 9% of the samples are segmented with zero accuracy, while this amount in the first method is equal to only 7%. The reason for the difference (9-7= 2%) is the error inducted to the calculation by modeling the reference models using a Normal Distribution function.

6.2.4 Diphone Segmentation

When evaluating the correctness of the diphone segmentation, I considered the fact that a diphone should begin from somewhere inside a phoneme and end somewhere

inside an adjacent phoneme. The result of comparing the beginning and end points of the phonemes, with the reference phoneme provided by TIMIT, is indicated in Figure 6.8. Figure 6.8 shows that 80% of the segmented diphones are confined to the defined boundaries and only 20% of the segmented diphones have either their start or end, or both start and end points, outside of the defined boundaries.

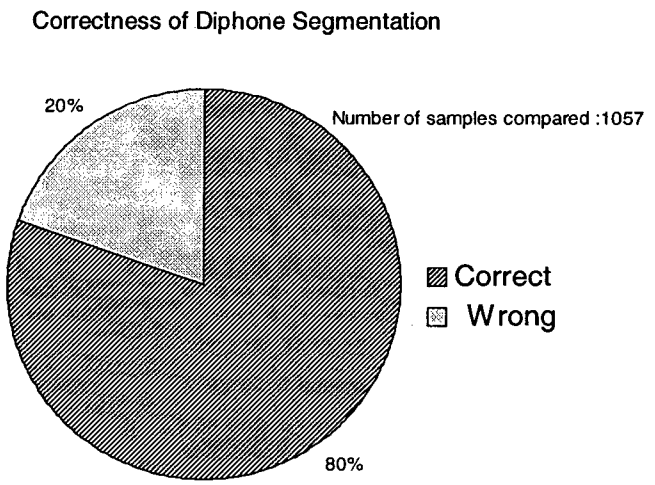

Figure 6.8 Correctness of diphones segmented using the Indirect-Method, compared with the TIMIT phoneme segmentation.

It is important to note that the model used for segmentation is not an adapted model, so if we adapt the model to a specific user and then segment the phonemes and diphones, we will gain much better results than those shown in Figures 6.8 and 6.7.

6.3 Direct Method

In the previous method, it was implicitly assumed that the phonemes were symmetrically balanced in their middle nodes, and that the frame times of rising and falling phoneme tone in the middle node were equal. As a result, the diphones were

segmented from their middles in the middle nodes. However, this assumption was incorrect, because the rising and falling time of a phoneme may not be equal. To address this problem, instead of segmenting triphones first and extracting diphones from them, I modeled and trained the system with diphones from the beginning, forcing the recognizer to directly recognize diphones.

In this method I have modeled and trained the system with diphones with *sp* as word boundaries and *sil* at the beginning and at the end of each utterance. For example, the utterance "`Her hum became a gurgle of surprise`", selected from file SI1837.lab, SI training list of TIMIT, is converted to the following phonetic equivalent:

```
sil-hh hh-er er-sp sp-hh hh-ah ah-m m-sp sp-b b-ih
ih-k k-ey ey-m m-sp sp-ah ah-sp sp-g g-er er-g g-ah
ah-l l-sp sp-ah ah-v v-sp sp-s s-er er-p p-r r-ay ay-
z z-sil
```

The dictionary used for this section is also transformed to the diphone equivalent. For example, the entries

```
A               ah
ABBREVIATE      ah b r iy v iy ey t
ABILITY         ah b ih l ah t iy
   :
```

from the CMU dictionary change to

```
A               sp-ah ah-sp
ABBREVIATE      sp-ah ah-b b-r r-iy iy-v v-iy iy-ey ey-t t-sp
ABILITY         sp-ah ah-b b-ih ih-l l-ah ah-t t-iy iy-sp
  :
```

Here the speech recognizer provides diphone transcription of the speech directly in the output as recognized speech, and the *Diphone Extractor* uses the information

provided to extract the diphones from the media file. A comparison of the correctness of segmented diphones using this method with the segmentation provided by the TIMIT database is indicated in Figure 6.9. As indicated in the pie chart, the boundaries of only 59% of the segmented diphones are in the expected region, and either the start or end points, or both, of 41% of the segmented diphones are beyond the boundaries of the phonemes provided by TIMIT.

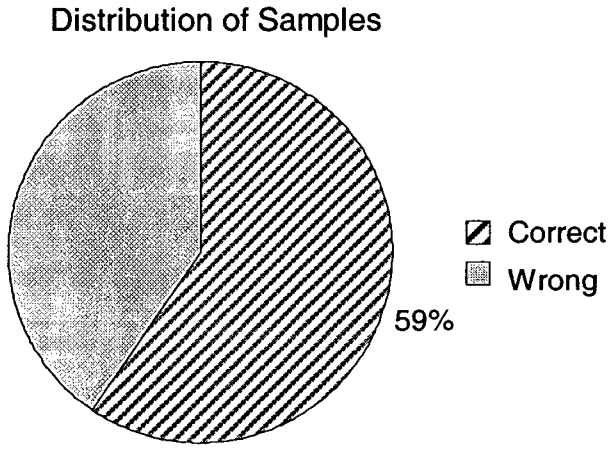

Figure 6.9 Correctness of diphones segmented with the Direct-Method compared with TIMIT phoneme segmentation.

Comparing Figures 6.9 and 6.8 indicates that the correctness of segmenting the diphones with the indirect method is about 80 –59 = 21% more than the correctness of segmenting the diphones with the direct method. The difference is probably due to insufficient training data. In fact, the training method for the triphone model begins with calculating the parameters of only 41 distinct phonemes, and then the training extends to the triphone model. The training for diphones, however, begins with calculating the same parameters for about 1600 distinct models, and it clearly needs much more data to adjust the parameters of all the models correctly.

Although the indirect method provides better correctness as compared to the correctness of the direct method, the accuracy of segmented diphones in the indirect method is questionable. To have a sense of the accuracy of segmentation, I have developed a speech synthesizer program to pronounce a text by concatenating the segmented diphones, provided by both direct and indirect methods.

6.4 Speech Synthesizer

Concatenation of the diphones does not provide high quality synthesized speech; instead, synthesizers use different methods of coupling and filtering to create a smooth voice [29] [30]. However, these details are beyond the scope of this thesis. For evaluation of the accuracy of the segmentation, I simply join the diphones to synthesize the speech. It is obvious that the quality of the synthesized speech will be poor, but the output provides a sense of the accuracy of the diphones that are segmented according to the above-mentioned methods. Curious readers may refer to [31] for practical methods of speech synthesis.

6.4.1 Program Options

For synthesizing a text, the program converts the text to its equivalent diphones with *sil* at the beginning and ending of the text and *sp* inserted in between the words; the same as WIT, but for diphones. For example, the utterance "His head flopped back" is converted to

```
sil-hh hh-ih ih-sp sp-hh hh-eh eh-d d-sp sp-f f-l l-
aa aa-p p-t t-sp sp-b b-ae ae-k k-sil
```

The program then looks into the diphones database and copies the diphones' binary file into a buffer and saves it as a WAVE format file.

In this program, I have developed two notations (M and D) for two different versions of speech synthesis using diphones.

M This notation stands for synthesizing speech from the diphones segmented with the indirect method. It uses **M** database.

D This notation stands for synthesizing speech from the diphones segmented with the direct method. It uses **D** database.

6.4.2 Speech Output 1

Figure 6.10 shows the original wave form of "His head flopped back" that is uttered by Keith. The Top picture of the Figure 6.11 shows the synthesized wave form of the same utterance from diphones segmented with the *Indirect Method*, and the bottom one shows the same utterance synthesized from the diphones segmented with the *Direct Method.*

As indicated in both figures, the four words are marked with circles; arrows show the corresponding words in each waveform. Furthermore, the two synthesized wave files indicate that the diphones segmented with the direct method create a waveform closer to the original wave file than the diphones segmented with the indirect method.

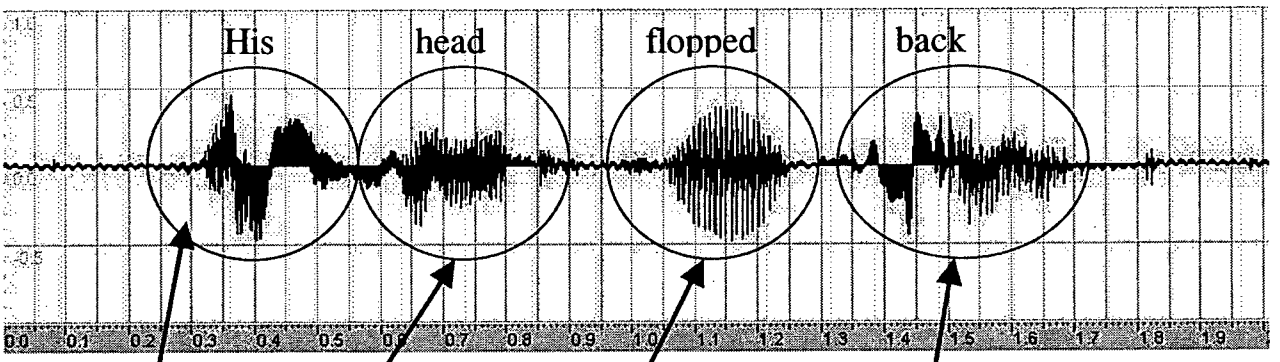

Figure 6.10 Original wave form of utterance "His head flopped back" uttered by Keith.

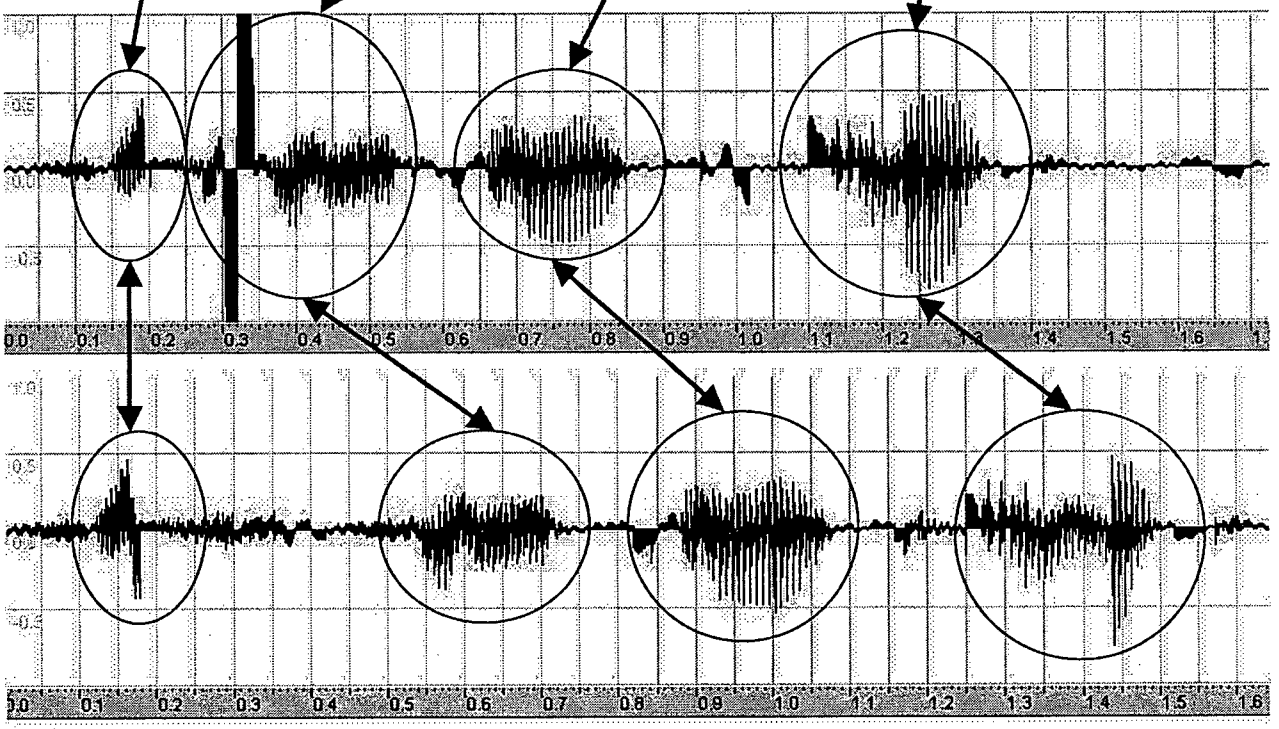

Figure 6.11 Synthesized utterance "His head flopped back." **(Top)** From diphones segmented with Indirect Method. **(Bottom)** From diphones segmented with Direct Method.

6.4.3 Speech Output 2

This is the same experiment as the previous one, but the test speech, "this is not really me, " was not used either in training or in the segmentation process. As shown in Figure 6.12, the utterance "this is not really me" is synthesized with two methods.

Comparing the two waveforms, except at the beginning and end of the waveforms, the method shows a more natural appearance for method D than method M. Furthermore, the speech created by method D has a more natural sound.

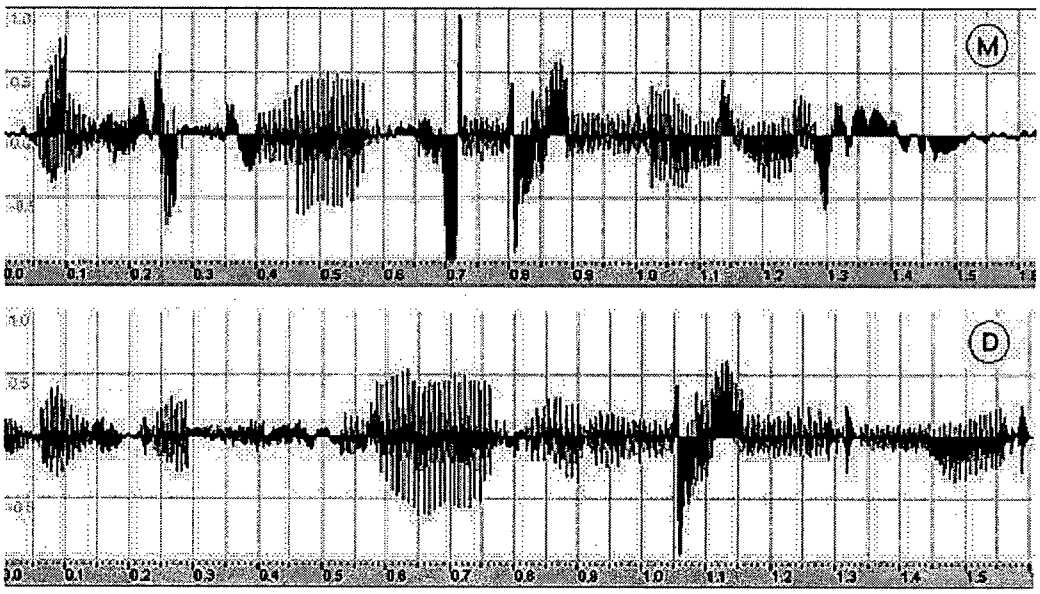

Figure 6.12 The utterance "this is not really me" synthesized with three methods. **(Top)** version M, **(Bottom)** Version D.

6.4.4 Comparision

Both Figures 6.11 and 6.12 indicate that speech synthesized by diphones segmented with the direct method has a more natural appearance and smoother tone than speech synthesized by the diphones segmented with the indirect method. However, my experiment with lab-mates indicates that the quality of the speech provided by Model D is limited to some words and in most cases, Model M shows a better quality of speech.

The reason may reside in the accuracy and correctness of the two models. In fact, the indirect method provides more correct segmentation than the direct method, but the direct method segments the diphones more accurately than the indirect method. Further, if there were enough training data to set the parameters of the diphones in the direct

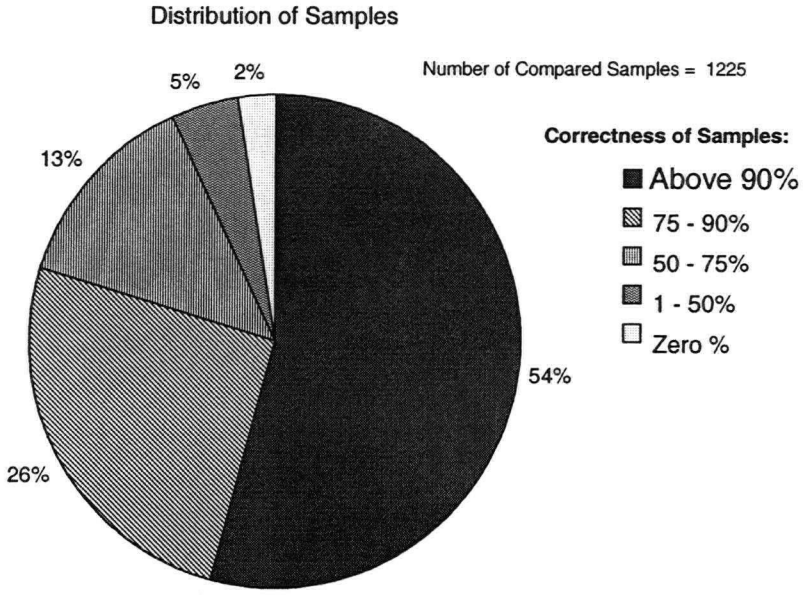

Figure 6.12 Distribution and accuracy of word segmentation.

method, the results might show the superiority of this method as compared with the indirect method for segmenting diphones. This is a possibility worth exploring if enough data were available to train the diphone model.

6.5 Word Segmentation

The same function shown in Figure 6.2 is employed to determine the accuracy of the program to segment words. Figure 6.12 shows the distribution and accuracy of the segmented words. As indicated in the figure, 54% of the words are segmented with an accuracy above 90%, 26% of the samples are segmented with an accuracy above 75%, and 18% of the words are segmented with an accuracy below 50%. However, 2% of the words are segmented completely wrongly, compared to the segmentation provided by TIMIT. These words are of very short duration, such as 'a', 'at', and similar words.

It is important to note that the TIMIT database, which is being used as a reference, is segmented manually and is prone to human error. Furthermore, the model used for all of the previous segmentations is not adapted. It is clear that the model, which

will eventually be used for this process, will be adapted to the speech of a specific user to achieve better accuracy.

6.6 Word Spotting

Other potential uses for the speech recognizer developed in the previous chapters are speech search applications. Because of the increasing use of multimedia to store and file data, tools that search and index media files without involving humans will be in high demand in the near future. The following simple application shows the functionality of such a system for searching and retrieving a word in an AVI file, without having the equivalent text of the media file.

The program shown in Figure 6.13 is a dialog designed to find the occurrence of words in AVI files. The dialog accepts a text and uses the speech recognizer developed so far to find, load, and play the occurrence of a text in the media file. The program is invoked by typing a word in the Query Edit box and pushing the Submit Query button. Then, the program processes the media file to find the occurrence of the input text. If the text is found, the part of speech containing the text appears in the Display box at the bottom of the dialog, and the queried text is highlighted. The AVI file containing that part of speech is displayed in the File box, and the program is then ready to play the AVI file, which utters the requested text. Otherwise, the phrase "Not found" will be displayed in the Response box. The recognition process takes considerable time, so I have developed an ODBS[10] to accelerate processing and decrease the response time of the program.

[10] Open Database System

The first time the program wants to search a media file, it separates the speech section of the AVI file and passes it to the speech recognizer developed in the previous section[11]. Then it indexes and stores the recognized text into the database. The next time a user inputs a text, the program looks into the database and matches the input text with the recognized speech. This process is much more efficient because the program has to process each media file only once.

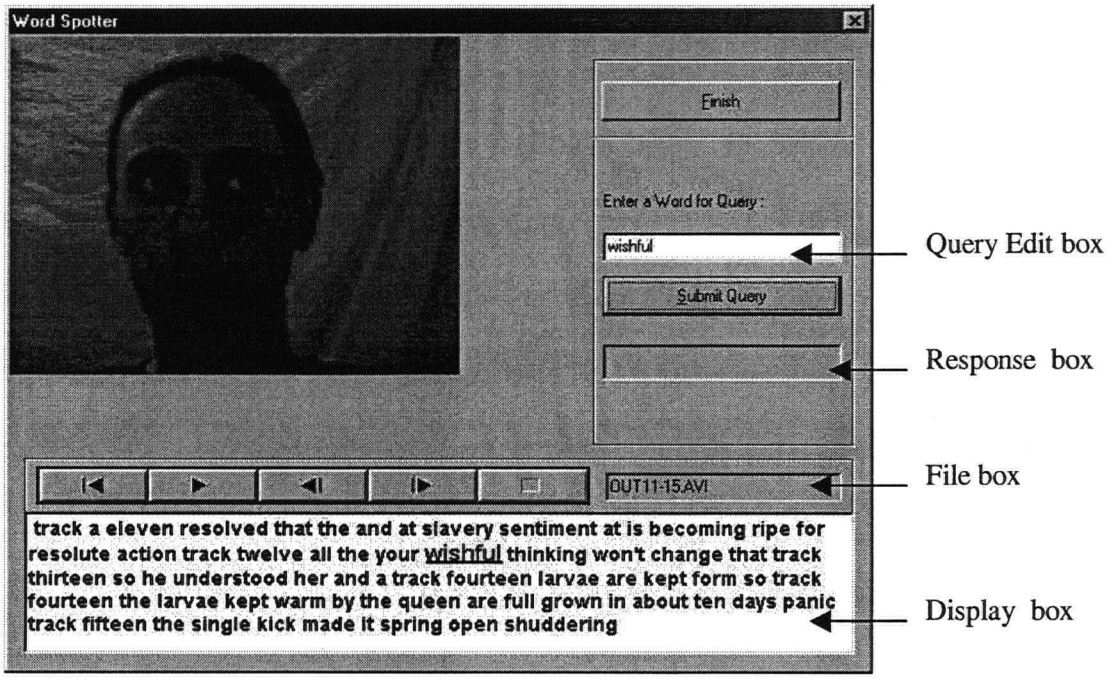

Figure 6.13 The dialog was asked to look for the occurrence of the word "wishful".

Figure 6.13 shows an instance of the dialog. In this dialog, the program is asked to find the word "wishful", by typing the word in the Query Edit box and pushing the Submit Query button. The program finds the word in the file OUT11-15.AVI and

[11] This program uses the adapted tied-model developed in the previous section.

highlights the word in the paragraph found by the speech recognizer. The program then loads the AVI file and is ready to play the media file to utter "wishful".

The recognized text is not completely correct. As indicated in the Display Box at the bottom of the dialog, the recognized text begins with "**track a eleven**" while the speech was "**track eleven,**" and the speech "**all your wishful thinking**" has been recognized as "**all the your wishful thinking.**" It is clear that the accuracy of the application for retrieving text is dependent upon the accuracy of the speech recognizer developed in the previous section. Figure 6.14 shows another instance of the program that is asked to look for occurrences of the word "family" in the media files. As indicated in the figure, the program finds the word in the file OUT01-04.AVI, highlights the word "family" in the Display box, and it is then ready to play and utter the word.

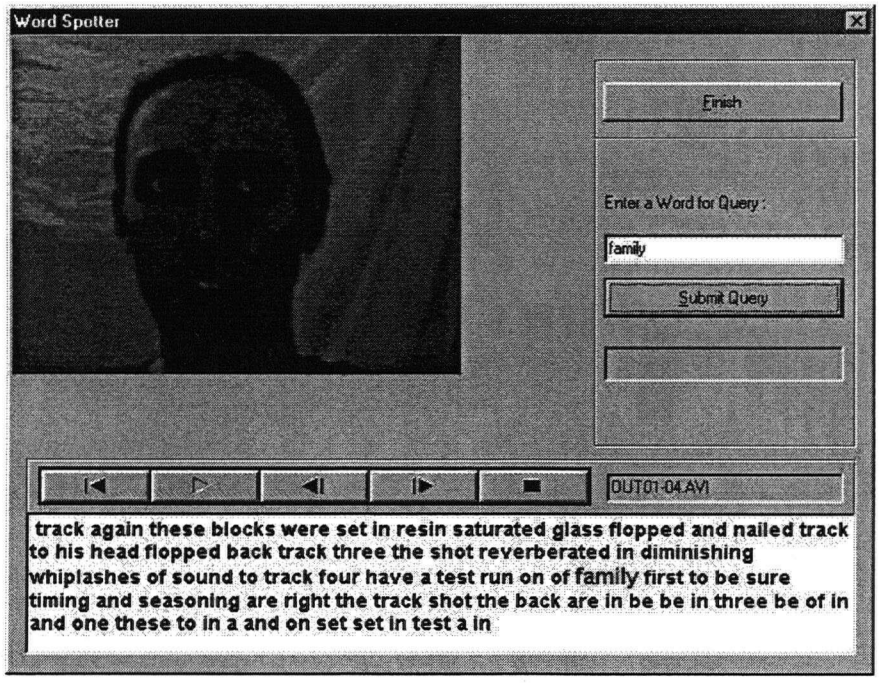

Figure 6.14 The dialog was asked to look for the occurrence of the word "family".

Chapter 7

Conclusions

7.1 Summary

The primary contribution of this thesis is to develop methods and to extend speech recognition technology to segment natural speech into words and diphones, so that the segments can be recombined to synthesize speech based on user input.

In this thesis, I used the MFCC method to preprocess speech signals, and uses HMM to model speech. However, there are a variety of methods for preparing speech signals for recognition and different methods for speech modeling and recognition.

The language model used in this project is a parallel network with the same probability for all words used, although, the possibility of uttering a word depends on the context and logical flow of the speech in each sentence.

For segmenting speech into diphones, the speech recognition engine is provided with both the speech to be recognized and its equivalent text. In this approach, the engine is able to recognize and segment the speech more accurately.

I have introduced two methods to evaluate the correctness and accuracy of phoneme segmentation. The discussion of correctness and accuracy of phoneme segmentation is introduced in this thesis for the first time. In the first method, the accuracy of segmentation is set to the ratio of the intersection of the recognized segments and reference segments provided by TIMIT. In the second method, I modeled a phoneme

duration with a Normal distribution, and compared the correctness of the segmented phoneme with the boundaries of the same phoneme in the same utterance provided by TIMIT. The decision to set the standard deviation of the model to half of the duration of the middle node of the acoustic model, was based on the fact that the middle node of a phoneme has a more important role in identifying a phoneme than the two transient nodes.

I have developed two methods to segment speech into diphones. In the first method (Indirect method), I have employed a phoneme-based speech recognition engine to segment speech into phonemes, and then adjacent phonemes are segmented into diphones. In the second method (Direct method), I have developed a speech recognition engine trained based on diphone recognition. In this method, input speech is segmented into diphones directly.

By considering the fact that a diphone begins some where inside a phoneme and ends somewhere inside the adjacent phoneme, I have compared the correctness of diphone segmentation, segmented with the two methods. Experiments show that 80 percent of the diphones segmented with the first method have both their start and end points located inside the defined boundaries, while only 59 percent of the diphones segmented with the second method have their both start and end points located inside the expected regions. This indicates that the Indirect method segments the diphones more correctly than the Direct method. The reason may be that the training process of the first method begins with estimating parameters of only 41 phonemes, while the same training data is employed to estimate the parameters of about 1300 diphones. This means that, on average, only 3.1 percent of the training data that is used to estimate the parameters of a

phoneme in the first method is employed to estimate the parameters of the diphones in the second method. In fact, the result may me different if the same amount of data per model are used to estimate the parameters of models in both methods.

To evaluate the accuracy of segmentation, I concatenate the diphones to synthesize speech with the diphones segmented with both methods. By examining the wave form of the synthesized speech, I noticed that the speech synthesized with the diphones segmented with second method is smoother than the speech synthesized with the diphones segmented with the first method. Furthermore, the speech created by second method has a more natural sound. It seems that, although the correctness of segmentation with the Direct method is less than the correctness of the segmentation with Indirect method, the boundaries of the correctly segmented diphones are more accurate in Direct method than the boundaries of the diphones segmented with Indirect method.

7.2 Future work

The main challenges facing speech recognition engines are speed and accuracy. Processing speed depends on the size of vocabulary being recognized and the speed of the processor that the program is running. However, the accuracy of recognition is a matter of software.

A dynamic word network is a good subject for further work. As recognition moves forward, the system changes the word network and modifies its transition probability to fit the contents recognized so far. Another possibility for improving accuracy is to model phonemes with different states and transitions, as we did for *sp* and *sil* in this project. This is a possibility worth exploring.

Diphone recognition engine is first introduced in this thesis, and each diphone is modeled with three left-right Markov Chain. However, studying acoustic behavior of each diphone and modeling each diphone with more accurate number of states and transitions can be a topic of further projects.

User-independent speech recognition will be a necessity in the future. As indicated in the project, to achieve better recognition, the model should be adjusted and tailored to the voice of a specific user. However, this may not be practical for a speech recognizer installed for public use; for example, a banking machine. As a result, a main goal of speech recognition is the development of a system that is able to recognize the speech of different users with the same, high accuracy. One possible approach to such a system would involve classifying different users into different groups, and loading models that most appropriately reflect the speech of the user. The main difficulty of developing such a system is classifying users and dynamically adapting the parameters of models to achieve better accuracy.

Appendixes

Appendix A

Regression Class Tree

A common approach using a binary regression class tree is shown below.

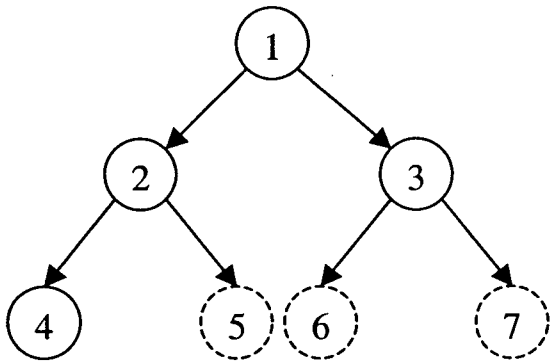

Figure A.1 A binary regression tree

The leaves of the tree are termed the *base regression classes*, and each Gaussian mixture component of a model set belongs to a single base class. For example, the class in Figure A.1. has four base classes, C4, C5, C6, and C7. During adaptation, occupation counts are accumulated for each of the base classes. The solid lines in the figure indicate that there are sufficient data for adaptation, and the dotted lines show the insufficiency of the data for the adaptation process. For example, neither nodes 6 nor 7 has sufficient data for adaptation. However, when they pool at node 3, there is enough data. The amount of data that determines sufficiency is definable.

The regression class tree is built using a centroid splitting algorithm, which yields clusters that lie in a similar portion of the acoustic space. The following algorithm provides a method to cluster and create a *Regression Class Tree*.

1. Select a terminal node that is to be split.
2. Calculate the mean and variance from the mixture components clustered at this node.
3. Create two children. Initialize their means to the parent mean perturbed in opposite directions for each child by a fraction of the variance.
4. For each component at the parent node assign the component to one of the children by using Euclidean distance, to which the mean is closer.
5. Once all the components have been assigned, calculate the new means for the children.

This algorithm is repeated until the desired number of child nodes is found.

Appendix B

EM Algorithm

EM determines the estimated parameters of a model such that

$$f(M_{new}) \geq f(M_{old}) \tag{B.1}$$

where M is the parameter of the model.

For implementing the EM algorithm, an auxiliary function is required. For speech recognition systems the function typically used is [32]

$$Q(M_{old}, M_{new}) = \sum_{s \in S} q(O, s \mid M_{old}) \log\{O, s \mid M_{new}) \tag{B.2}$$

where S contains all possible state sequences leading to the recognition of the O.

The equation (B.2) expands to

$$Q(M_{old}, M_{new}) = \sum_{s \in S} q(O, s \mid M_{old}) \left(\sum_{t=1}^{T} \log(transition\_prob.) + \sum_{t=1}^{T} \log b_{s_t}(o_t) \right) \tag{B.3}$$

Since we are interested only in the transformation matrix, we can ignore the first part of the right hand side of the equation, and (B.3) reduces to

$$Q(M_{old}, M_{new}) = \sum_{s \in S} \sum_{t=1}^{T} q(O, s \mid M_{old}) \log b_{s_t}(o_t) \tag{B.4}$$

After substituting (3.14) and (3.34) into (B.4), and the state occupancy count from (3.30), differentiate the $Q(M_{old}, M_{new})$ with respect to W and the right hand side equal to zero, and group the terms of W. We will then have the following[12]:

[12] The details are too many to be referenced in this thesis; curious readers should refer to [20] [32] [33] [34] [35] for details.

$$\sum_{t=1}^{T} L(t)\Sigma^{-1}o(t)\xi^{T} = \sum_{t=1}^{T} L(t)\Sigma^{-1}o(t)W\xi\xi^{T} \tag{B.5}$$

Here ξ is the extended mean vector and Σ is the covariance matrix. If W is shared by R states, then the general form expands to this:

$$\sum_{t=1}^{T}\sum_{r=1}^{R} L_r(t)\Sigma_r^{-1}o(t)\xi_r^{T} = \sum_{t=1}^{T}\sum_{r=1}^{R} L_r(t)\Sigma_r^{-1}o(t)W\xi_r\xi_r^{T} \tag{B.6}$$

Where $L_r(t)$, is the occupation likelihood, defined as described in (3.28)

$$L_r(t) = P(q_r(t)\,|\,O_t, M) \tag{B.7}$$

and $q_r(t)$ indicates the Gaussin component at time *t*, O_T={o(1),…,o(T)} is the adaptaion data. The equation (B.6) is rewritten, thus:

$$\sum_{t=1}^{T}\sum_{r=1}^{R} L_r(t)\Sigma_r^{-1}o(t)\xi_r^{T} = \sum_{t=1}^{T}\sum_{r=1}^{R} V_r W D_r \tag{B.8}$$

Where

$$V_r = \sum_{t=1}^{T} L_r(t)\Sigma_r^{-1} \tag{B.9}$$

and

$$D_r = \sum_{t=1}^{T} \xi_r\, \xi_r^{T} \tag{B.10}$$

Let's define Z: the right hand side of (B.8) to be a $n\times(n+1)$ matrix; then

$$Z = \sum_{t=1}^{T}\sum_{r=1}^{R} L_r(t)\Sigma_r^{-1}o(t)_r\xi_r^{T} \tag{B.11}$$

Also let the elements of Z, V, W, and D, be z_{ij}, v_{ij}, w_{ij}, and d_{ij} respectively; then the formula (B.11) can be rewritten this way

$$z_{ij} = \sum_{pt=1}^{n}\sum_{q=1}^{n+1} w_{pq}\sum_{r=1}^{R} v_{ip}^{(r)} d_{qj}^{(r)} \tag{B.12}$$

Since D_r is symmetrical, equation (B.12) can be rewritten, thus:

$$z_{ij} = \sum_{q=1}^{n+1} w_{iq} \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} \tag{B.13}$$

setting

$$g_{jq}^{(i)} = \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} \tag{B.14}$$

yields

$$z_{ij} = \sum_{q=1}^{n+1} w_{iq} g_{jq}^{(i)} \tag{B.15}$$

where Z and G can be computed from the observation vectors and model parameters. So we will have this:

$$w_i^t = (G^{(i)})^{-1} z_i^t \tag{B.16}$$

where w_i and z_i are the i^{th} rows of W and Z respectively.

The use of a regression class tree to generate classes dynamically does not introduce a problem into the above formulation; instead, if the parent node R has children {R1,..., Rc} then

$$Z = \sum_{c=1}^{C} Z_{R_c} \tag{B.18}$$

and

$$G = \sum_{c=1}^{C} G_{R_c} \tag{B.18}$$

The same rules with some modifications apply to finding a variance transformation matrix.

$$\Sigma = B^T H B \tag{B.19}$$

where H is the linear transformation to be estimated and B is the inverse of the Choleski factor of Σ^{-1}, so

$$\Sigma^{-1} = CC^T \tag{B.20}$$

and

$$B = C^{-1} \tag{B.21}$$

By employing the same auxiliary function as defined in (B.4)

$$Q(M_{old}, M_{new}) = \sum_{s \in S} \sum_{t=1}^{T} q(O, s \mid M_{old}) \log b_{s_t}(o_t) \tag{B.22}$$

After expanding $\log b_j(o_t)$ and differentiating $Q(M_{old} \mid M_{new})$ with respect to H and equating it to zero [32] we will have this:

$$H = \frac{\sum_{r=1}^{R} C_r^T \left[L_r(t)(o(t) - \mu_r)(o(t) - \mu_r)^T \right] C_r}{L_r(t)} \tag{B.23}$$

Example of the EM Algorithm

Assume that the following defines a single state in a recognition system using the two-dimensional acoustic space with diagonal covariance

$$\mu_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix}$$

$$\mu_0 \begin{bmatrix} 4.1 \\ 3.4 \end{bmatrix}, \Sigma_0 = \begin{bmatrix} 0.02 & -0.02 \\ -0.02 & 0.02 \end{bmatrix}$$

Now let us assume that we have two frames of adaptation data, thus

$$o_1 = \begin{bmatrix} 4 \\ 3.5 \end{bmatrix}, o_1 = \begin{bmatrix} 4.2 \\ 3.3 \end{bmatrix}$$

Recalling (B.16), we will solve the set of functions:

$$w_i^t = (G^{(i)})^{-1} z_i^t$$

$$Z = \sum_{t=1}^{T} \sum_{r=1}^{R} L_r(t) \Sigma_r^{-1} o(t)_r \xi_r^T$$

where w_i and z_i are the i^{th} rows of W and Z respectively.

If we set offset equal to 1, then the extended mean vector will be

$$\mu = [1 \quad 2 \quad 3]$$

and if we assume $L_1(1) = 0.3$, and $L_1(2)$=0.8 then

$$Z = 0.3 \begin{bmatrix} 0.25 & 0 \\ 0 & 0.111 \end{bmatrix} \begin{bmatrix} 4 \\ 3.5 \end{bmatrix} [1 \quad 2 \quad 3] + 0.8 \begin{bmatrix} 0.25 & 0 \\ 0 & 0.111 \end{bmatrix} \begin{bmatrix} 4.2 \\ 3.3 \end{bmatrix} [1 \quad 2 \quad 3]$$

Then

$$Z = \begin{bmatrix} 1.14 & 2.28 & 3.42 \\ 0.4096 & 0.8192 & 1.2288 \end{bmatrix}$$

For a diagonal covariance, we define the elements of G_i, thus:

$$g_{jq}^{(i)} = \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)}, \qquad q \in (1,\ldots,n+1)$$

$$V_r = \sum_{t=1}^{T} L_r(t) \Sigma_r^{-1} = 0.3 \begin{bmatrix} 0.25 & 0 \\ 0 & 0.111 \end{bmatrix} + 0.8 \begin{bmatrix} 0.25 & 0 \\ 0 & 0.111 \end{bmatrix}$$

$$D_r = \sum_{t=1}^{T} \xi_r \xi_r^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1 \quad 2 \quad 3] = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

$$V_r = \begin{bmatrix} 0.275 & 0 \\ 0 & 0.1221 \end{bmatrix}$$

At this point, we have what we need to solve G_1:

$$G_1 = 0.275\begin{bmatrix}1 & 2 & 3\\ 2 & 4 & 6\\ 3 & 6 & 9\end{bmatrix} = \begin{bmatrix}0.275 & 0.550 & 0.825\\ 0.550 & 1.100 & 1.650\\ 0.825 & 1.650 & 2.475\end{bmatrix}$$

$$G_2 = 1.1221\begin{bmatrix}1 & 2 & 3\\ 2 & 4 & 6\\ 3 & 6 & 9\end{bmatrix} = \begin{bmatrix}0.122 & 0.244 & 0.366\\ 0.244 & 0.488 & 0.733\\ 0.366 & 0.733 & 1.099\end{bmatrix}$$

$$(G_1)^{-1} = \begin{bmatrix}1.855E-2 & 3.711E-2 & 5.566E-2\\ 3.711E-2 & 7.421E-2 & 1.113E-1\\ 5.566E-2 & 1.113E-1 & 1.669E-1\end{bmatrix}$$

$$(G_2)^{-1} = \begin{bmatrix}4.179E-2 & 8.357E-2 & 1.254E-1\\ 8.357E-2 & 1.671E-1 & 2.507E-1\\ 1.254E-2 & 2.507E-1 & 3.761E-1\end{bmatrix}$$

By substituting G into (B.16) we get this:

$$w_1^T = \begin{bmatrix}1.855E-2 & 3.711E-2 & 5.566E-2\\ 3.711E-2 & 7.421E-2 & 1.113E-1\\ 5.566E-2 & 1.113E-1 & 1.669E-1\end{bmatrix}\begin{bmatrix}1.14\\ 2.28\\ 3.42\end{bmatrix} = \begin{bmatrix}0.2961\\ 0.5922\\ 0.8883\end{bmatrix}$$

$$w_2^T = \begin{bmatrix}4.179E-2 & 8.357E-2 & 1.254E-1\\ 8.357E-2 & 1.671E-1 & 2.507E-1\\ 1.254E-2 & 2.507E-1 & 3.761E-1\end{bmatrix}\begin{bmatrix}0.4096\\ 0.8192\\ 1.2288\end{bmatrix} = \begin{bmatrix}0.2396\\ 0.4792\\ 0.7188\end{bmatrix}$$

We can now compute the adapted means:

$$W_1 = \begin{bmatrix}0.2961 & 0.5922 & 0.8883\\ 0.2396 & 0.4792 & 0.7188\end{bmatrix}$$

$$\mu_1 = W_1\xi_1 = \begin{bmatrix}0.2961 & 0.5922 & 0.8883\\ 0.2396 & 0.4792 & 0.7188\end{bmatrix}\begin{bmatrix}1\\ 2\\ 3\end{bmatrix} = \begin{bmatrix}4.145\\ 3.355\end{bmatrix}$$

As indicated, the mean has moved closer to the observed mean. A similar process is used to calculate the adapted covariance matrix.

Appendix C

Tree-Based Clustering

A Tree-Based clustering is a binary tree in which a yes/no phonetic question is attached to each node [36], and according to the route a triphone traverses, it ends up in a leaf node. Then all the nodes in the leaf node are considered phonetically similar and they

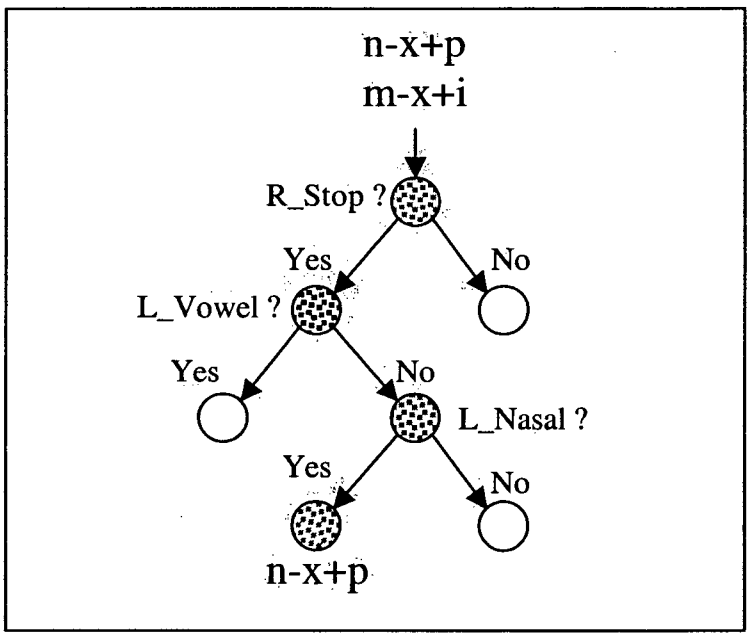

Figure C1 Decision tree-bases state tying

can be grouped to share parameters. For example, Figure C1 shows a case of tying the center states of a triphone of phoneme **X**. In this example, the triphone n-x+p will end up in the lower shaded node, because its right is "Stop" and its left is "Nasal".

To create a tree, the phonetic questions that categorize the phonetic context of a triphone state must be defined. The assumption behind the choice of phonetic questions is that phonemes that belong to the same phonetic class have a similar influence on the pronunciation of a phoneme. The set of questions defined for this project is copied from

the set of questions developed at Cambridge University. However, the questions are modified to fit the phoneme set employed in this project (phoneme set used by CMU). Refer to Appendix D for the list of questions used for this project.

Tied-state triphone

In this step, similar acoustic states of the triphones are tied to reduce the number of parameters and ensure that all state distributions can be robustly estimated. As a result, triphone states whose emission probabilities are very similar are tied together. These tied states share the parameters evaluated by all observations assigned to the set. The tied model is much smaller that the untied model, so it can be implemented more efficiently, compared with the untied one.

Initially, all the selected models are grouped into the root node AB. Then this node is split using the phonetic question from the set of questions that yields the biggest likelihood of improvement $\Delta(A,B)$ for the child nodes A and B [20]:

$$\Delta(A,B) = \left| L(A)L(B) - L(AB) \right|$$

$$\Delta(A,B) = \left| \left(\frac{1}{2} (n_A \sum_{d=1}^{D} \log \left[\frac{\sigma_{d,AB}}{\sigma_{d,A}} \right]^2 + n_B \sum_{d=1}^{D} \log \left[\frac{\sigma_{d,AB}}{\sigma_{d,B}} \right]^2 \right) \right|$$

Where n_x is the number of observations for node x, D dimentionality of the feature vector and $\sigma_{d,x}$ the variance of component d of node x.

This process is repeated until the increase in log likelihood falls below the threshold specified for $\Delta(A,B)$. As a final stage, the decrease in log likelihood is

calculated for merging terminal nodes with different parents. Any pair of nodes for which this decrease is less than the threshold used to stop splitting are then merged.

In this project all the states of each triphone model are clustered as

```
("x","*-x+*","x+*","*-x").state[i]
```

where X is a phoneme and i is the state of the model. This process is implemented for each $X \in phoneme\_set$ *and* $i = [2 \quad 3 \quad 4]$.

The HTK provides a tool, HHed, for clustering purposes. I have used the question set in Appendix D with a threshold equal to 350.

Appendix D

Question set employed for clustering

The Questions provided are copied from the set of questions developed at University of Cambridge, but modified to fit the phoneme set we adapted in this project, the CMU phoneme set.

R_Silence	*+sil
R_Pause	*+sp
R_Stop	*+p,*+b,*+t,*+d,*+k,*+g
R_Nasal	*+m,*+n,*+ng
R_Fricative	*+s,*+sh,*+z,*+zh,*+f,*+v,*+ch,*+jh,*+th,*+dh
R_Liquid	*+l,*+r,*+w,*+y,*+hh
R_Vowel	*+eh,*+ih,*+ao,*+aa,*+uw,*+ah,*+er,*+ay,*+oy,*+ey,*+iy,*+ow
R_C-Front	*+p,*+b,*+m,*+f,*+v,*+w
R_C-Central	*+t,*+d,*+n,*+s,*+z,*+zh,*+sh,*+th,*+dh,*+l,*+r
R_C-Back	*+sh,*+ch,*+jh,*+y,*+k,*+g,*+ng,*+hh
R_V-Front	*+iy,*+ih,*+eh
R_V-Central	*+eh,*+aa,*+er,*+ao
R_V-Back	*+uw,*+aa,*+uh
R_Front	*+p,*+b,*+m,*+f,*+v,*+w,*+iy,*+ih,*+eh
R_Central	*+t,*+d,*+n,*+s,*+z,*+zh,*+sh,*+th,*+dh,*+l,*+r,*+eh,*+aa,*+er,*+ao
R_Back	*+sh,*+ch,*+jh,*+y,*+k,*+g,*+ng,*+hh,*+aa,*+uw,*+uh
R_Fortis	*+p,*+t,*+k,*+f,*+th,*+s,*+sh,*+ch
R_Lenis	*+b,*+d,*+g,*+v,*+dh,*+z,*+zh,*+sh,*+jh
R_UnFortLenis	*+m,*+n,*+ng,*+hh,*+l,*+r,*+y,*+w
R_Coronal	*+t,*+d,*+n,*+th,*+dh,*+s,*+z,*+zh,*+sh,*+ch,*+jh,*+l,*+r
R_NonCoronal	*+p,*+b,*+m,*+k,*+g,*+ng,*+f,*+v,*+hh,*+y,*+w
R_Anterior	*+p,*+b,*+m,*+t,*+d,*+n,*+f,*+v,*+th,*+dh,*+s,*+z,*+zh,*+l,*+w
R_NonAnterior	*+k,*+g,*+ng,*+sh,*+hh,*+ch,*+jh,*+r,*+y
R_Continuent	*+m,*+n,*+ng,*+f,*+v,*+th,*+dh,*+s,*+z,*+zh,*+sh,*+hh,*+l,*+r,*+y,*+w
R_NonContinuent	*+p,*+b,*+t,*+d,*+k,*+g,*+ch,*+jh
R_Strident	*+s,*+z,*+zh,*+sh,*+ch,*+jh
R_NonStrident	*+f,*+v,*+th,*+dh,*+hh
R_UnStrident	*+p,*+b,*+m,*+t,*+d,*+n,*+k,*+g,*+ng,*+l,*+r,*+y,*+w
R_Glide	*+hh,*+l,*+r,*+y,*+w
R_Syllabic	*+m,*+l,*+er
R_Unvoiced-Cons	*+p,*+t,*+k,*+s,*+sh,*+f,*+th,*+hh,*+ch
R_Voiced-Cons	*+jh,*+b,*+d,*+dh,*+g,*+y,*+l,*+m,*+n,*+ng,*+r,*+v,*+w,*+z,*+zh
R_Unvoiced-All	*+p,*+t,*+k,*+s,*+sh,*+f,*+th,*+hh,*+ch,*+sil,*+sp
R_Long	*+iy,*+aa,*+ow,*+ao,*+uw,*+m,*+l

R_Short	*+eh,*+ey,*+aa,*+ih,*+ay,*+oy,*+ah,*+uh
R_Dipthong	*+ey,*+ay,*+oy,*+aa,*+er,*+m,*+l
R_Front-Start	*+ey,*+aa,*+er
R_Fronting	*+ay,*+ey,*+oy
R_High	*+ih,*+uw,*+aa,*+iy,
R_Medium	*+ey,*+er,*+aa,*+eh,*+m,*+l
R_Low	*+eh,*+ay,*+aa,*+aw,*+ao,*+oy
R_Rounded	*+ao,*+uw,*+aa,*+oy,*+w
R_Unrounded	*+eh,*+ih,*+aa,*+er,*+ay,*+ey,*+iy,*+aw,*+ah,*+m,*+hh, *+l,*+r,*+y
R_NonAffricate	*+s,*+sh,*+z,*+zh,*+f,*+v,*+th,*+dh
R_Affricate	*+ch,*+jh
R_IVowel	*+ih,*+iy
R_EVowel	*+eh,*+ey
R_AVowel	*+eh,*+aa,*+er,*+ay,*+aw
R_OVowel	*+ao,*+oy,*+aa
R_UVowel	*+aa,*+m,*+l,*+uw
R_Voiced-Stop	*+b,*+d,*+g
R_Unvoiced-Stop	*+p,*+t,*+k
R_Front-Stop	*+p,*+b
R_Central-Stop	*+t,*+d
R_Back-Stop	*+k,*+g
R_Voiced-Fric	*+z,*+zh,*+sh,*+dh,*+ch,*+v
R_Unvoiced-Fric	*+s,*+sh,*+th,*+f,*+ch
R_Front-Fric	*+f,*+v
R_Central-Fric	*+s,*+z,*+zh,*+th,*+dh
R_Back-Fric	*+sh,*+ch,*+jh
R_aa	*+aa
R_ae	*+ae
R_ah	*+ah
R_ao	*+ao
R_aw	*+aw
R_ay	*+ay
R_b	*+b
R_ch	*+ch
R_d	*+d
R_dh	*+dh
R_eh	*+eh
R_er	*+er
R_ey	*+ey
R_f	*+f
R_g	*+g
R_hh	*+hh
R_ih	*+ih
R_iy	*+iy
R_jh	*+jh
R_k	*+k
R_l	*+l
R_m	*+m
R_n	*+n
R_ng	*+ng
R_ow	*+ow
R_oy	*+oy
R_p	*+p

R_r	*+r
R_s	*+s
R_sh	*+sh
R_t	*+t
R_th	*+th
R_uh	*+uh
R_uw	*+uw
R_v	*+v
R_w	*+w
R_y	*+y
R_z	*+z
R_zh	*+zh
L_NonBoundary	*-*
L_Silence	sil-*
L_Pause	sp-*
L_Stop	p-*,b-*,t-*,d-*,k-*,g-*
L_Nasal	m-*,n-*,ng-*
L_Fricative	s-*,sh-*,z-*,zh-*,f-*,v-*,ch-*,jh-*,th-*,dh-*
L_Liquid	l-*,r-*,w-*,y-*,hh-*
L_Vowel	eh-*,ih-*,ao-*,aa-*,uw-*,ah-*,ax-*,er-*,ay-*,oy-*,ey-*,iy-*,ow-*
L_C-Front	p-*,b-*,m-*,f-*,v-*,w-*
L_C-Central	t-*,d-*,n-*,s-*,z-*,zh-*,sh-*,th-*,dh-*,l-*,r-*
L_C-Back	sh-*,ch-*,jh-*,y-*,k-*,g-*,ng-*,hh-*
L_V-Front	iy-*,ih-*,eh-*
L_V-Central	eh-*,aa-*,er-*,ao-*
L_V-Back	uw-*,aa-*,uh-*
L_Front	p-*,b-*,m-*,f-*,v-*,w-*,iy-*,ih-*,eh-*
L_Central	t-*,d-*,n-*,s-*,z-*,zh-*,sh-*,th-*,dh-*,l-*,r-*,eh-*,aa-*,er-*,ao-*
L_Back	sh-*,ch-*,jh-*,y-*,k-*,g-*,ng-*,hh-*,aa-*,uw-*,uh-*
L_Fortis	p-*,t-*,k-*,f-*,th-*,s-*,sh-*,ch-*
L_Lenis	b-*,d-*,g-*,v-*,dh-*,z-*,zh-*,sh-*,jh-*
L_UnFortLenis	m-*,n-*,ng-*,hh-*,l-*,r-*,y-*,w-*
L_Coronal	t-*,d-*,n-*,th-*,dh-*,s-*,z-*,zh-*,sh-*,ch-*,jh-*,l-*,r-*
L_NonCoronal	p-*,b-*,m-*,k-*,g-*,ng-*,f-*,v-*,hh-*,y-*,w-*
L_Anterior	p-*,b-*,m-*,t-*,d-*,n-*,f-*,v-*,th-*,dh-*,s-*,z-*,zh-*,l-*,w-*
L_NonAnterior	k-*,g-*,ng-*,sh-*,hh-*,ch-*,jh-*,r-*,y-*
L_Continuent	m-*,n-*,ng-*,f-*,v-*,th-*,dh-*,s-*,z-*,zh-*,sh-*,hh-*,l-*,r-*,y-*,w-*
L_NonContinuent	p-*,b-*,t-*,d-*,k-*,g-*,ch-*,jh-*
L_Strident	s-*,z-*,zh-*,sh-*,ch-*,jh-*
L_NonStrident	f-*,v-*,th-*,dh-*,hh-*
L_UnStrident	p-*,b-*,m-*,t-*,d-*,n-*,k-*,g-*,ng-*,l-*,r-*,y-*,w-*
L_Glide	hh-*,l-*,r-*,y-*,w-*
L_Syllabic	m-*,l-*,er-*
L_Unvoiced-Cons	p-*,t-*,k-*,s-*,sh-*,f-*,th-*,hh-*,ch-*
L_Voiced-Cons	jh-*,b-*,d-*,dh-*,g-*,y-*,l-*,m-*,n-*,ng-*,r-*,v-*,w-*,z-*,zh-*
L_Unvoiced-All	p-*,t-*,k-*,s-*,sh-*,f-*,th-*,hh-*,ch-*,sil-*,sp-*
L_Long	iy-*,aa-*,ow-*,ao-*,uw-*,m-*,l-*
L_Short	eh-*,ey-*,aa-*,ih-*,ay-*,oy-*,ah-*,uh-*

L_Dipthong	ey-*,ay-*,oy-*,aa-*,er-*,m-*,l-*
L_Front-Start	ey-*,aa-*,er-*
L_Fronting	ay-*,ey-*,oy-*
L_High	ih-*,uw-*,aa-*,iy-*
L_Medium	ey-*,er-*,aa-*,eh-*,m-*,l-*
L_Low	eh-*,ay-*,aa-*,aw-*,ao-*,oy-*
L_Rounded	ao-*,uw-*,aa-*,oy-*,w-*
L_Unrounded	eh-*,ih-*,aa-*,er-*,ay-*,ey-*,iy-*,aw-*,ah-*,m-*,hh-*,l-*,r-*,y-*
L_NonAffricate	s-*,sh-*,z-*,zh-*,f-*,v-*,th-*,dh-*
L_Affricate	ch-*,jh-*
L_IVowel	ih-*,iy-*
L_EVowel	eh-*,ey-*
L_AVowel	eh-*,aa-*,er-*,ay-*,aw-*
L_OVowel	ao-*,oy-*,aa-*
L_UVowel	aa-*,m-*,l-*,uw-*
L_Voiced-Stop	b-*,d-*,g-*
L_Unvoiced-Stop	p-*,t-*,k-*
L_Front-Stop	p-*,b-*
L_Central-Stop	t-*,d-*
L_Back-Stop	k-*,g-*
L_Voiced-Fric	z-*,zh-*,sh-*,dh-*,ch-*,v-*
L_Unvoiced-Fric	s-*,sh-*,th-*,f-*,ch-*
L_Front-Fric	f-*,v-*
L_Central-Fric	s-*,z-*,zh-*,th-*,dh-*
L_Back-Fric	sh-*,ch-*,jh-*
L_aa	aa-*
L_ae	ae-*
L_ah	ah-*
L_ao	ao-*
L_aw	aw-*
L_ay	ay-*
L_b	b-*
L_ch	ch-*
L_d	d-*
L_dh	dh-*
L_eh	eh-*
L_er	er-*
L_ey	ey-*
L_f	f-*
L_g	g-*
L_hh	hh-*
L_ih	ih-*
L_iy	iy-*
L_jh	jh-*
L_k	k-*
L_l	l-*
L_m	m-*
L_n	n-*
L_ng	ng-*
L_ow	ow-*
L_oy	oy-*
L_p	p-*
L_r	r-*

L_s	s-*
L_sh	sh-*
L_t	t-*
L_th	th-*
L_uh	uh-*
L_uw	uw-*
L_v	v-*
L_w	w-*
L_y	y-*
L_z	z-*
L_zh	zh-*

Table D1 The set of questions are used for clustering.

References

[1] T. Dutoit, "An Introduction to TTS Synthesis," Kluwer Academic Publishers, 1994

[2] F.J. Owens, "Signal Processing of Speech," Mcmillan New Electronics.

[3] K. Torkkola, "Astochastic Models and Artificial Neural Networks for Sutomatic Speech Recognition," Istitut Dalle Molle d'Intelligence Artifielle Perceptive (IDIAP), C.P. 609, CH-1920 Martigny, Switzerland.

[4] D.B. Roe, J.G. Wilpon, "Voice Communication Between Humans And Machine," National Academy of Sciences, Washington D.C. 1994

[5] R.L. Scheaffer, "Introduction to Probability and its Applications," Second Edition, Duxbury Press, 1995.

[6] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 35, no. 3, pp.400-01, March 1987

[7] H. Ney, R. Kneser, U. Essen, " On the estimation of small probabilities by leaving one out," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAM1-17, no.12, pp. 1202-12, December 1995.

[8] G. Chollet, "Automatic Speech and Speaker Recognition: Overview, Current Issues and Perspectives," Istitut Dalle Molle d'Intelligence Artifielle Perceptive (IDIAP), C.P. 609, CH-1920 Martigny, Switzerland, and CNRS, URA 820, 46 rue Barrault, F-75634 PARIS CEDEX, France.

[9] B. Gold, "Speech and Audio Signal Processing," Massachusetts Institute of Technology, Lincoln Laboratory. Nelson Morgan, University of California at Berkeley, International Computer Science Institute

[10] P. Bhaskararao, "Sub-phonemic Segment Inventories for Concatenative Speech Synthesis," Deccan College, PUNE 6, India and ILCAA, Tokyo University of Foreign Studies, TOKYO, Japan.

[11] D.H. Klatt, "Review of text-to-speech conversion for English," JASA, 82:3, 737-93,. 1987.

[12] "Project of SPHINX," http://www.speech.cs.cmu.edu, Carnegie Mellon University.

[13] L.C. Oliveira, "Text-to-Speech Synthesis with Dynamic Control of Source Parameters," INESC/IST, Portugal, 1997.

[14] M.E. Beckman, "Speech Models and Speech Synthesis," Ohio State university, Department of Linguistics, USA, 1996.

[15] S. Furui, "Cepstral Analysis Techniques for Automatic Speaker Verification," IEEE Tran. On ASSP, 29, No.2, pp.254-272, 1981.

[16] L.R. Rabiner, R.W. Shafer, "Digital Processing of Speech Signal," Prentice Hall, 1978

[17] C. Becchetti, F.U. Bordoni, L.P. Ricotti, "Speech Recognition, Theory and C++ Implementation," Rome, Italy

[18] A. Oppenheim, "Signal and Systems," Englewood Cliffs, N.J. : Prentice-Hall, c1983.

[19] L.B. Jackson, " Digital Filters and Signal Processing," Third Edition, Kluwer Academic Publisher, 1996.

[20] "HTK Tool kit," Cambridge University, 2000

[21] N.O.S. Jayant, P. Noll, "Digital Coding of Waveforms," Prentice Hall, 1984.

[22] J. Makhoul, R. Schwartz, "State of the Art in Continuous in Speech Recognition," National Academy of Science, Washington, D.C.1994.

[23] F.S. Samaria, "Face recognition using Hidden Markov Model," Engineering Department, Cambridge University.

[24] F.R. Chen, L.D. Wilcox, D.S. Bloomberg, "Word spotting in scanned images using Hidden Markov Models," proc. ICASSp, 5, p.1 (1993)

[25] B. Hannaford, "Hidden Markov Model analysis of manufacturing process information," IROS 1991, Osaka, Japan (Nov 1991)

[26] J. Henderson, S. Salzberg, K. Fasman, "Finding genes in human DNA with Hidden Markov Model," Journal of Computational biology, 4, No. 2. Pp. 127-142 (1997)

[27] T. Kanungo, "Hidden Markov Models," Center for Automation Research, University of Maryland, www.cfar.umd.edu/~kanungo

[28] L.R. Rabinder, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," IEEE, Vol. 77, No. 2, February 1988.

[29] A.D. Conkie, S. Isard, "Optimal Coupling of Diphones," Center for Speech Technology Research, University of Edinburgh, Scotland, 1977.

[30] G. Richard, C.R. d' Alessandro, "Modification of the Aperiodic Component of Speech Signals for Synthesis," Center for Computer Aids for Industrial Productivity, Rutgers University, USA and LIMSI-CNRS, Orsay, France.

[31] V. Santen, Sproat, Olive, Hirschberg "Progress in Speech Synthesis," Springer, Verlag New York, 1997.

[32] J. Picone, " Special topics in Speech Recognition," Department of Electrical Engineering, Mississippi State University, 2000.

[33] C. J. Leggetter, P. C. Woodland, "Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression," Proceedings of the ARPA Spoken Language Technology Workshop, Barton Creek, 1995.

[34] M. Gales, P.C. Woodland, "Variance Compensation Within the MLLR Framework," Technical Report CUED/F-INFENT/TR242, Cambridge University Engineering Department, February 1996.

[35] P. Zhan, M. Westphal, M. Finke, Alex Waibel, "Speaker Normalization and Speaker Adaptation - A Combination for Conversational Speech Recognition," Proceedings of Eurospeech 1997, Rhodes, Greece, September 1997.

[36] L. Yi, P. Fung, " Decision Tree-based Triphone are Robust and Practical for Mandarin Speech Recognition," Human Language Technology Center, University of Science and Technology , Hong Kong.