Speech Accent Identification and Speech Recognition Enhancement by Speaker Accent Adaptation

by

Mohammad M. Tanabian, B.Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science in Information Systems Science

Carleton University, Ottawa, Ontario, Canada
Department of Systems and Computer Engineering

March 2005

The undersigned hereby recommends to
The Faculty of Graduate Studies and Research
acceptance of the thesis,

Speech Accent Identification and Speech Recognition Enhancement by Speaker Accent Adaptation

submitted by

Mohammad M Tanabian

In partial fulfillment of the requirements
for the degree of Master of Information Systems Science

Professor Rafik Goubran, Thesis Supervisor

Chairman, Department of Systems and Computer Engineering

Carleton University

ABSTRACT

This thesis examines the impact of accent present in speech on the performance of speaker independent Automatic Speech Recognition (*ASR*) systems.

The research is focused on validating the assumption that the presence of accent in speech can increase the error rate and that a speaker independent *ASR* engine, trained by a variety of accents, performs poorer than an engine that is trained for a particular accent, when tested by the same accent. It also proposes a method to lower the recognition error rate and measure the improvement by first determining the accent of the utterance and then applying the appropriate ASR engine from a bank of engines trained for different accents. The second part of this research work proposes a method for identifying accents by tracking the temporal evolution of speaker's vocal tract. An *LPC* feature space is proposed to model the speaker's vocal tract.

ACKNOWLEDGEMENTS

First I would like to thank my thesis supervisor Dr. Rafik Goubran for his tremendous support and invaluable guidance and ideas. Without his supportive critiques, encouragement and patience this work would not be possible. I would like to thank my friend Dr. Bahram Zahirazami for his constant supply of ideas, reviews and comments for whom I have a lot of respect and I value his opinions and criticisms highly. I would like to thank my friend Donna Hogan for assisting me in navigating less understood sides of Microsoft Word and helping me with formatting this document. I would like to thank my wife Mojdeh who supported me with her love and stood by my side and sacrificed on our family time to allow me to finish this work.

DEDICATION

To my mother and my father who gave me the passion for learning, early on in my life

Table of Contents

List of Figures

List of Tables

CHAPTER 1 Introduction

This chapter is summary of this thesis document and outlines its subsequent sections. It describes the research motivations of the work and explains the problem that this research is attempting to address. It also outlines the thesis contributions and at the end explains the upcoming chapters.

1.1 Motivation

Automatic Speech Recognition (*ASR*) systems are being used in a variety of products. They have started to appear in toys, speech enabled software packages, automated attendants for call centers – e.g. Bell Canada's Emily-- and automobiles. The commercial ASR systems are built to be speaker independent and are usually trained, mainly by speech samples of native accent speakers and are developed for one language and, for one accent. Although they work relatively well with native accent speakers, they perform rather poorer when the speaker presents a non-native accent. One reason for this is that non-native speakers show different strengths in reading and pronunciation abilities. Particularly, they show a large amount of variability in pronouncing sounds that are not part of their native language. For example, in many *Arabic* dialects, the phonemes /g/, /p/, /ʒ/ and /tʃ/ don't exist. With speech recognition going mainstream, and the increase in the number of people migrating from one

country to another, accent is becoming a serious consideration in developing state-of-the-art *ASRs*.

At the same time, in countries with more than one official language, such as Canada, this becomes an even more important issue, since a big portion of the population has a different accent when speaking in the other official language that is not their mother tongue. Non-native accent can also be found in tourist centers or in automatic international phone call services.

Accent identification belongs to the broader problem of language identification (*LI*), which is also one of the new challenges in automatic speech recognition systems. The idea is that, if we can identify the speaker's language, they can be routed to the appropriate service center. In the case of telephone call center, this means routing the call to the appropriate customer service representative. If the customer's call is being handled automatically by a computer, identifying the customer's language, will allow that call to be routed to the appropriate *ASR* engine. On a more practical note, since English is the most widely used second language, these services often are provided in English. In this case, an *ASR* system that could detect and adapt to the accent of the speaker, would perform better than a system that is built for native accent.

Another example can be found in the automobile industry. Speech enabled command and control systems (*C&C*) are available on most high end automobiles and are becoming an option on a wider range of make and models. Car makers sell their products globally and dealing with localization of their products is an important factor in staying competitive.

Accent adaptation in *C&C* systems in automobiles also poses a strong potential in better positioning of new cars in different markets.

1.2 Problem Statement

Speech recognition experts deal with a wide range of problems. These issues include speaker variation, speech ambiguity, variation within individual speakers, noise and interference and speaker accents. These issues are discussed in more detail in Chapter 2. The focus of this thesis is to investigate the effects of accented speech on the performance of speech recognition systems and on how to address it. Many available commercial *ASR* systems do not perform well when used by speakers that have a different accent than the one the system was trained with. Some of the available systems are trained with a cocktail like collection of accents to be more tolerant to accented speech. The problem with these systems is that at the expense of generality, they deliver higher average error rates. We propose a solution based on first, identifying the accent of the speaker and then selecting the appropriate recognition engine from a bank of trained engines. The appropriate engine is the one that is trained for the identified accent.

1.3 Thesis Contributions

We developed a lab environment to study the effects of accent on the performance of an *HMM* based recognizer with a widely used *TIMIT* database. We have validated an assumption that we made at the beginning of our research work that when dealing with accented speech, a system that can identify the accent and adapt to it, will perform better when compared to a system based on native accents only. We also developed and tested a system to identify accents from an utterance. We proposed an accent classification approach based on tracking speaker's vocal tract variation during vowel utterances. We used LPC feature based trajectory as a model to track temporal evolution of the vocal tract. Our complete proposed solution is composed of an accent classifier as a pre-processing stage with a bank of engines that are each trained with a particular accent.

1.4 Thesis Organization

This thesis examines an adaptation mechanism based on the native language of the user to reduce the error rate of the recognition systems.

For the purpose of this thesis, the language of the recognition engine is chosen to be American English (English). This choice has a significant impact on the availability of tools, databases and other related works than can expedite the research.

This thesis is organized into six chapters. In chapter 2, the fundamentals that are required to understand speech processing, speech recognition, accuracy enhancement and the proposed solution for accent identification are discussed.

Chapter 3 provides an overview of the experimental setup. It explains the tools that we used, the *TIMIT* database and the pre-processing that we did to prepare the data for the simulation experiments.

Chapter 4 discusses the effects of accent on *ASR* systems and the result of the experiments that we conducted. These are the core concepts of this thesis. This chapter explains the block diagrams and the findings of the experiments.

Chapter 5 discusses the accent identification part of our research work. In this chapter details of the database, feature selection and the neural networks classifier are explained.

Chapter 6 closes the thesis and provides concluding discussions and suggestions for future related work.

CHAPTER 2 Background

This chapter provides a review of speech processing and speech recognition. In the first part, speech chain, speech production organs and speech modeling are explained. Speech recognition is described in the second part of the chapter and a list of current issues in the field of speech recognition is discussed.

2.1 Speech Recognition

Automatic Speech Recognition -- or *ASR* for short -- is a sub domain of Man-Machine Interface field that deals with enabling machines to communicate with human users using verbal conversation. Speech synthesis systems can verbalize words, allowing computers to utter words and sentences.

It was in the 1950s that researchers were dreaming of a "*phonetic typewriter*". With the advancing knowledge of speech signal analysis and processing and knowing about formants, transitions and acoustics of consonants, they believed that the process of phonetic transcription could be done by machines and subsequently phonemic and eventually *graphemic* (i.e. written) would follow as a matter of course.

ASR has been of interest to not only scientists and engineers but has inspired such science fiction series as the computer aboard the *Enterprise spaceship* in Gene Rodenbery' *Start Trek* TV series, the computer *HAL* in Stanley Kubrick's movie *2001-A Space Odyssey* and the robot *R2D2* in George Lucas's *Star Wars* movie.

There have been enormous amount of research publications and scientific literature on the subject of speech recognition. But despite the value of having a machine that is able to understand spoken words, and the significant effort that has gone into building one, we are far from the ideal goal of a system that can recognize spoken phrases by any talker in any language, in any environment and condition.

Speech processing in general and speech recognition in particular are considered relatively new topics that are a cross-convergence of several disciplines including the followings [1]:

- **Electrical Engineering and Digital Signal Processing:** This discipline deals with the process of extracting useful information from the speech signal which is used in the recognition mechanism.
- **Computer Science:** Computer scientists are busy developing effective algorithms to model and perform different tasks in speech recognition software and hardware. As an example pattern recognition algorithms are essential in any *ASR* system. Once the special features are extracted from the speech signal, a set of algorithms and methods are applied to it to cluster them and compare them to reference patterns in order to identify the words, sentences, etc.
- **Physics and Acoustics:** To design an effective *ASR* system, understanding the physical nature of sounds produced by speakers, the mechanics of physiological

organs that contribute in this process and the process of speech perception are fundamental.

- **Linguistics:** Linguistics is the science of studying the relationship between sounds (phonology), words (syntax), meaning of words (semantics) and perception concluded based on meaning (pragmatics). This discipline also embodies the concept of grammar and language parsing. As ASR systems are becoming more complex, language models and semantic information are playing important roles in delivering higher performance.
- **Physiology:** This discipline deals with the study of human body organs that are responsible for speech production, reception and understanding. As a result of studying the physiology of the human body, useful tools and concepts have been developed. For example, artificial neural networks that are used in pattern recognition are modeled based on the human brain's neurons and their mass parallel connectivity.
- **Psychology and ergonomy:** This is the science of studying human related factors and their influence in design and implementation of speech driven user interfaces. This is becoming more important as ASR systems are being implemented in voice enabled command and control mechanisms and for conversational scenarios.

2.2 Speech related applications

A wide range of industries are attempting to benefit from artificially generated speech and automatic speech recognition to increase their productivity. Among them, telecommunication service providers and call center operators are pioneers in using speech technology. For

instance, Bell Canada, over the last few years, has introduced a new voice enabled service to handle customer service calls [2]. The service is offered by a virtual attendant called *Emily*. Customers can, through conversation, tell *Emily* the nature of their request, and then the system routes their call to the right department.

The marine weather radio channels services across North America are using Text-To-Speech (*TTS*) systems to broadcast weather information. The weather information is gathered and processed by a computer system, then read by the *TTS* system and then broadcasted.

In recent years, car makers have also become important users of speech technology. Recent models of automobiles offer a speech interface, or *C&C*, to allow the driver to control different functions such as dialing a phone number, playing music or directing the navigation system by voice commands.

2.3 Speech Processing

2.3.1 The speech chain

The speech chain includes all the stages where speech is generated by the speaker and transmitted through air, and received and comprehended by the listener [3]. As shown in Figure 2-1, the process begins with conceptualizing the message that is to be uttered by the talker. Then, the speaker, using his/her speech production organs, generates the sounds required to convey the message in mind.

Along with the linguistic message related sounds, extra information including the identity of the talker, his/her mood and gender as well as the talker's accent is conveyed through the

generated voice. These extra pieces of information are usually included unconsciously and most often unintentionally and inevitably. Once the voices relating to message are generated by exhaling air that has gone through a talker's speech production organs, it travels through the air in the form of sound waves. At the listener's end, these sound waves create varying air pressure, which are detected by the listener's ear. This varying pressure is turned into nervy signals and then transmitted to the listener's brain, where it is decoded. Here is where the speech chain is completed. The conceptual and perceptual parts of the speech chain are not yet well understood, but there has been a vast amount of research on artificial production, transmission, audition and recognition of speech. Digital computers and in general digital signal processing have been the main drivers of speech processing in recent decades.

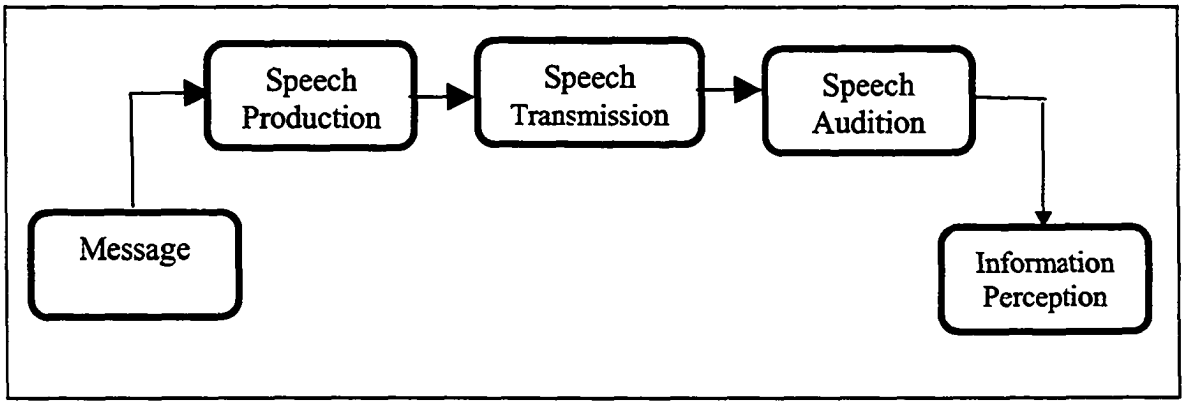

Figure 2-1: The Speech Chain

2.3.2 Speech production

Figure 2-2 shows a schematic diagram of the vocal organs [5]. Speech production is the result of coordinated movements of many anatomical structures. Pressurized air from the lungs, below the *larynx*, provides the energy for speech signal excitation. The *vocal cords*

inside the *larynx* can be positioned in a way that pressurized air can flow through the *glottis* (the space between the vocal folds) with or without causing the vocal folds to vibrate.

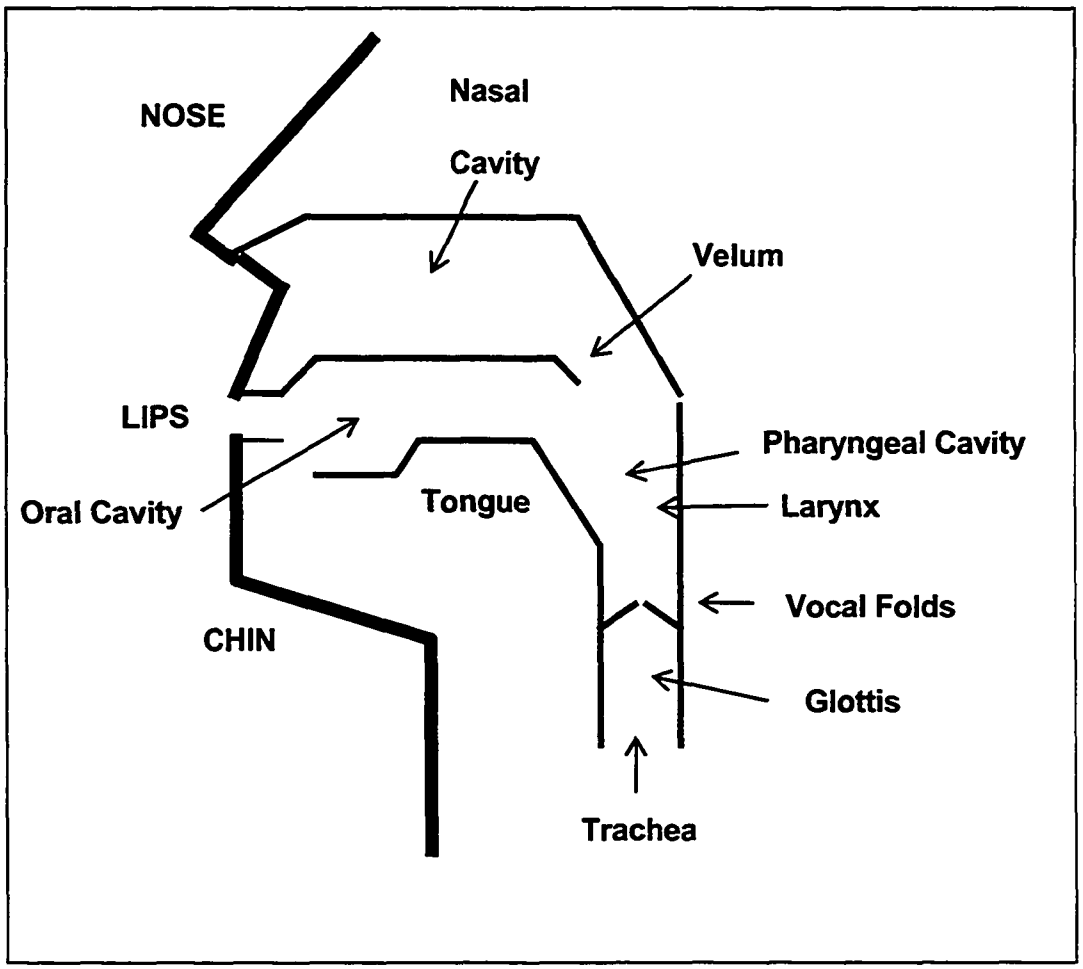

Figure 2-2: The speech production organs

Air is forced up from the lungs, and while passing through the glottis and other narrowings in the *vocal tract*, either periodic or aperiodic excitation is produced. The other parts of the vocal tract including *tongue*, *jaw*, *lips*, *velum* and *nasal cavities* act as resonants that modify the spectrum of this excitation before it is emitted from the lips as sound waves. To understand speech processing and speech recognition, the following concepts are defined:

- **Speech Spectrum:** The human auditory system is capable of receiving and comprehending audio signals with a frequency in the range of 16 *Hz* to 16 *KHz*. As humans age, the upper limit drops and can fall as low as 10 *KHz*. Among the young, it can reach as high as 20 *KHz*. At the lower end of the spectrum, the perceived sound becomes a pulse train and at the high end, it fades into silence [4].

- **Spoken language:** Every language is composed of small units of sounds, which together form words which make up the language. These small units are called phonemes [3]. The number of unique sounds is greater than the number of individual alphabet letters in most languages hence phoneticians have had to introduce a new system of identifying phonemes. Table 2-1 and Table 2 show the phonemes that comprise the American English language. The most widely accepted notation displaying phonemes is the International Phonetic Alphabet or IPA which is given in the first column of these two tables. Since IPA symbols are based on Roman characters, they cannot be produced by most computer displays and printers, therefore a substitute has been developed using ASCII characters. ASCII notations are shown in the second column of the tables for each phoneme.

- **Vowels and consonants:** Conventionally, speech sounds are categorized into vowels and consonants. In some literature, they are also referred to as vocoids and contoids [1]. Consonants are simpler to define in terms of anatomical production.

IPA	ASCII	ARPABET	EXAMPLES
b	b	b	bad, lab
d	d	d	did, lady
f	f	f	find, if
g	g	g	give, flag
h	h	m	how, hello
j	j	y	yes, yellow
k	k	k	cat, back
l	l	l	leg, little
m	m	m	man, lemon
n	n	n	no, ten
ŋ	N	ng	sing, finger
p	p	p	pet, map
r	r	r	red, try
s	s	s	sun, miss
ʃ	S	sh	she, crash
t	t	t	tea, getting
tʃ	tS	ch	check, church
θ	th	th	think, both
ð	TH	dh	this, mother
v	v	v	voice, five
w	w	w	wet, window
z	z	z	zoo, lazy
ʒ	Z	zh	pleasure, vision
dʒ	dZ	jh	just, large

Table 2-1: Consonant phonemes

IPA	ASCII	ARPABET	EXAMPLES
ʌ	^	uh	cup, luck
ɑ:	a:	aa	arm, father
æ	@	ae	cat, black
e	e	eh	met, bed
ə	..	ea	away, cinema
ɜ:ʳ	e:(r)	er	turn, learn
ɪ	i	iy	hit, sitting
i:	i:	ih	see, heat
ɒ	o	aa	hot, rock
ɔ:	o:	ao	call, four
u	u	uw	put, could
u:	u:	uh	blue, food
aɪ	ai	ay	five, eye
aʊ	au	aw	now, out
oʊ	Ou	ow	go, home
eəʳ	e..(r)	axr	where, air
eɪ	ei	ey	say, eight
ɪəʳ	i..(r)	ia	near, here
ɔɪ	oi	oy	boy, join
ʊəʳ	u..(r)	ux	pure, tourist

Table 2-2 Vowel Phonemes

Consonants are often defined by the point and manner of articulation. Point of articulation is mainly the location of the principal activity in the vocal tract and the participating organs. For example for bilabial sounds, the point of articulation is between the lips. Table 2-3 Principal points of articulation summarizes the different points of articulations.

Name	Description
Bilabial	Between lips
Labiodental	Between lower lip and upper teeth
Apicodental	Tip of tongue on teeth
Apicogigival	Tip of tongue on gums
Apicoveolar	Tip of tongue on alveolar ridge
Apicodomal	Tip of tongue on alveolar ridge ("Retroflex")
Laminoalveolar	Blade of tongue on alveolar ridge ("Palatalized")
Laminodomal	Blade of tongue on hard palate
Centrodomal	Middle of tongue on back of hard palate ("Palatal")
Dorsovelar	Back of tongue on velum ("Velar")
Pharyngeal	Root of tongue constricting oral pharynx
Glottal	Between vocal cords

Table 2-3 Principal points of articulation

Manner of articulation is principally the degree of constriction at the point of articulation and the way of release into the upcoming sound. Table 2-4: Principal manners of articulations shows the main categories in manner of articulation and their descriptions.

Name	Description
Plosive	Vocal tract shuts off at point of articulation, nasal passages shut off at velum. Plosives have clean, sharp release. Also called stops
Aspirated	Vocal tract initially shut, as with plosives; release marked by puff of air before next speech sound
Affricate	Initial closure of vocal tract followed by gradual release producing turbulence
Fricative	Vocal tract partially open at point of articulation, velum shut. Turbulent noise created at point of articulation. Also called spirants or sibilants
Lateral	Vocal tract closed at point of articulation but open at sides
Semivowel	Vocal tract partly open at point of articulation without turbulence. (These are consonantal vocoids; glides [w] and [j] are in this category
Nasal	Vocal tract closed at point of articulation; velum open
Trill	Oscillatory opening and closure at point of articulation

Table 2-4: Principal manners of articulations

Vowels are produced by exciting a non-constricted vocal tract with a periodic excitation produced by vocal cord vibration. In contrast to consonants, vowels are much less defined. This is mainly because the tongue never touches another organ when vowels are uttered. Rather, the vowels are defined in terms of position or the distance of the tongue from other parts of the mouth.

We can associate vowels with a combination of the following positions:

1. Tongue high or low
2. Tongue front or back
3. Lips rounded or un-rounded
4. Nasalized or un-nasalized

Figure 2-3 shows the vowel triangle and the relative locations of common vowels on a chart of first (F1) and second (F2) formant frequencies [7].

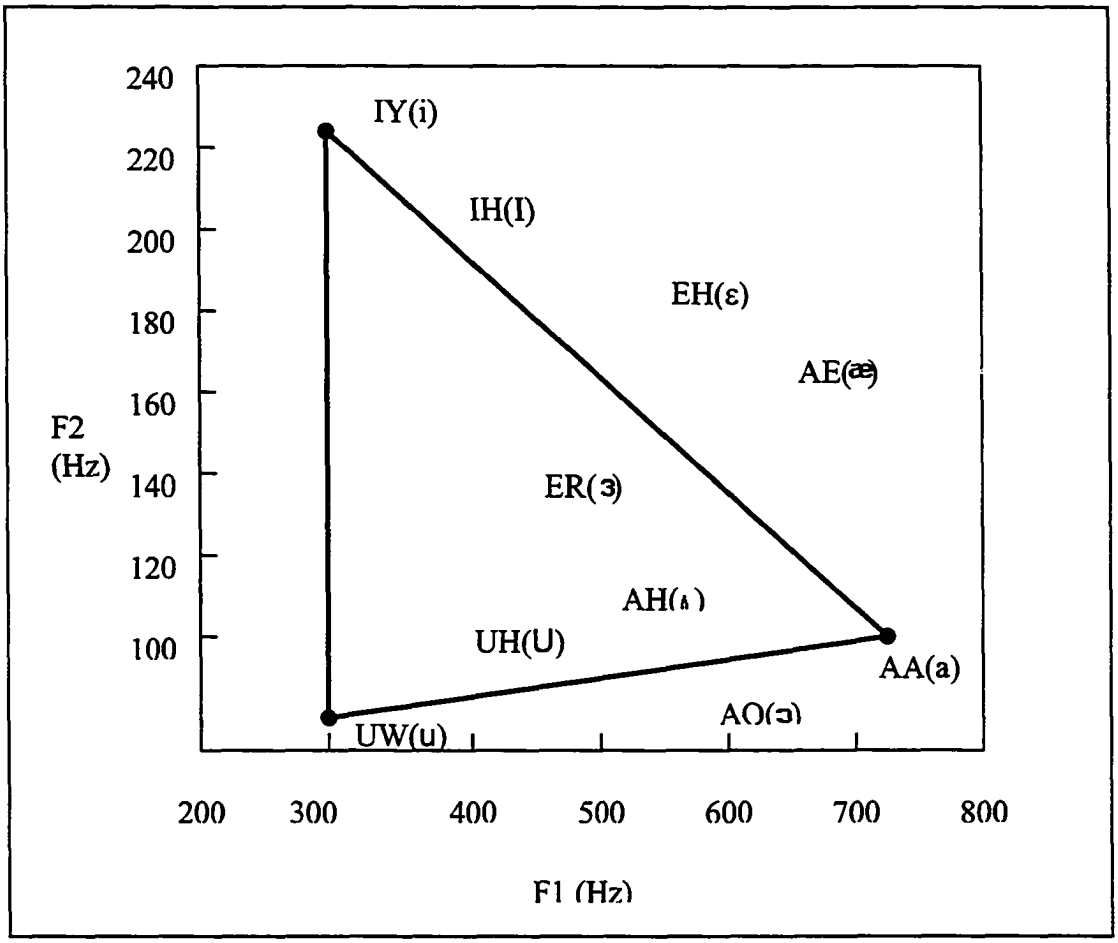

Figure 2-3: The Vowel Triangle with centroid position of the common vowels. F1 and F2 are first and second formant frequencies

Acoustic phonetics: Acoustically, the vocal tract is a non-uniform cross-section, which is approximately 17 cm long in adult males [1]. It is usually open on one end and nearly closed on the other. At its approximate midpoint, the nasal cavity which is about 13 cm long branches off, with a valve at the branch (*velum*), as shown in Figure 2-4.

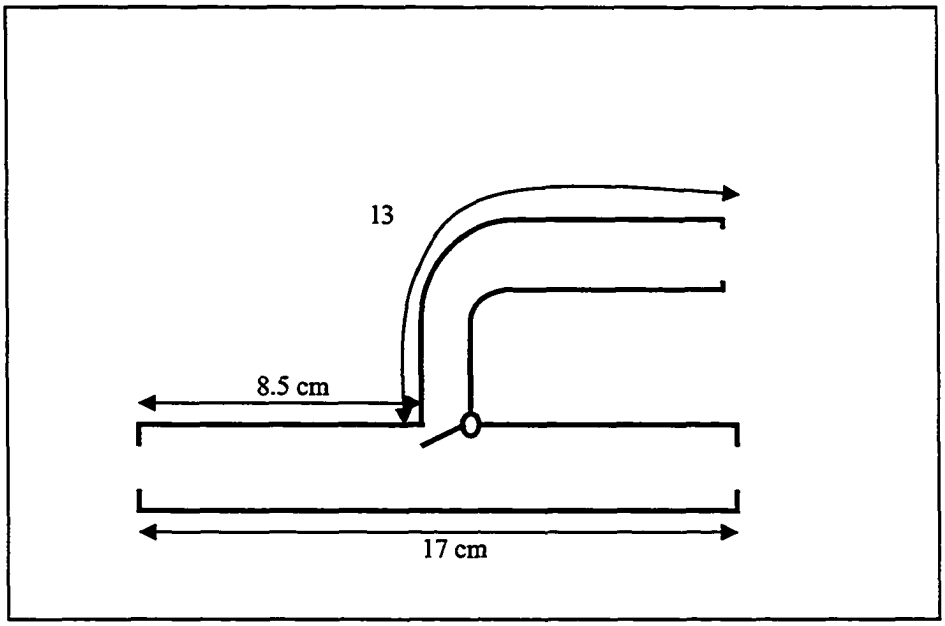

Figure 2-4: A simplified illustration of vocal tract and nasal cavity
The junction is the velum.

This modeled vocal tract is a distributed structure and therefore has many natural frequencies. Natural frequencies are those at which the transfer function of the tube has

maximum amplitude. With this simplified model, the location of these frequencies on the spectrum would be determined by the relationship shown in (2-1):

$$f_n = \frac{(2n-1)c}{4l}, \quad n = 1,2,3,... \qquad (2\text{-}1)$$

In the air, $c=350\ m/s$; and for a tube of length $l=17\ cm$, the natural frequencies occur at odd multiples of approximately $500\ Hz$ and n is the order of the natural frequencies. Since the real vocal tract is not uniform, the resonances are not equally spaced. These resonances are known as formants and are the most important acoustical characteristic of the vocal tract. Generally, three to five formants are sufficient to model speech, but in a speech synthesizer the extra resonances add to the accuracy and to the naturalness of the synthesized speech. These resonant frequencies correspond to the poles of the transfer function of the vocal tract. The glottal pulse train has many harmonics, and when passed through the vocal tract with several resonances, the output has information about the speaker's vocal tract structure. Table 2-5 Frequency ranges for the first three formants in adult males and females

Formants	**Adult males**	**Adult females**	**Bandwidth, Hz**
F1	200-800	250-1000	**40-70**
F2	600-2800	700-3300	**50-90**
F3	1300-3400	1500-4000	**60-80**

Table 2-5 Frequency ranges for the first three formants (F1, F2 and F3)

This is one of the ways a listener can distinguish between different speakers. Figure 2-5 shows the range of the first three formants in adult males and females [1].

First and second formants are principally dependent on the location and height of the tongue. Figure 2-5 shows frequency ranges for the first three formants, F1, F2 and F3 [5].

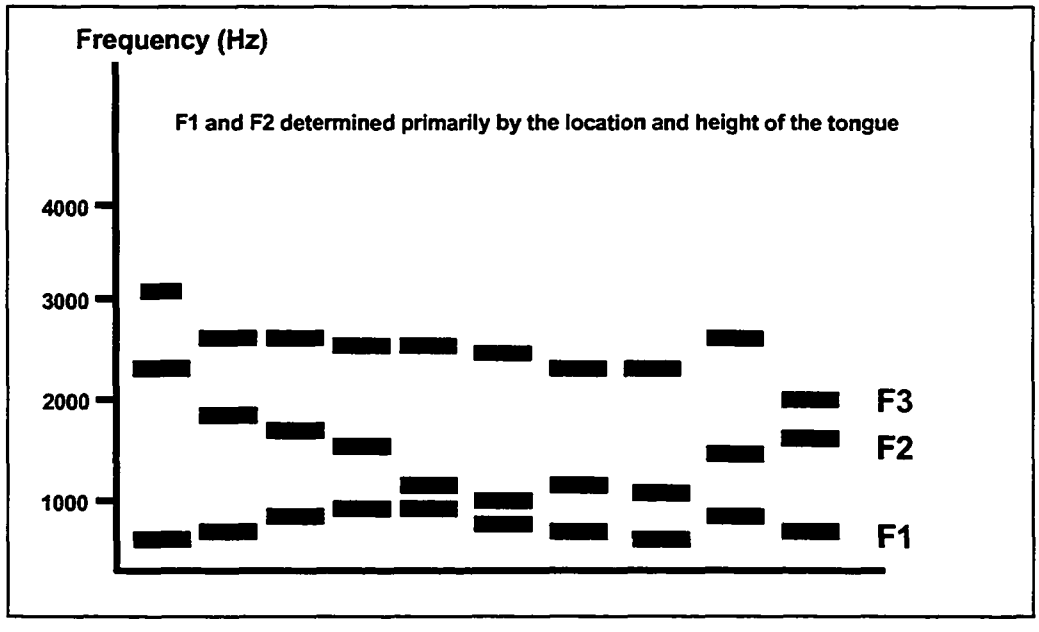

Figure 2-5: Frequency range for the first three formant frequencies F1, F2 and F3

A well known experiment on formants was done by Peterson and Barney who recorded vowels from 33 men, 28 women and 15 children then analyzed their formant frequencies. They plotted the first formant (lower frequency) along the X axis and the second formant along the Y axis as it is shown in

Figure 2-6. Then they plotted the range of each vowel on the diagram. It is important to notice the overlap of vowels in this diagram.

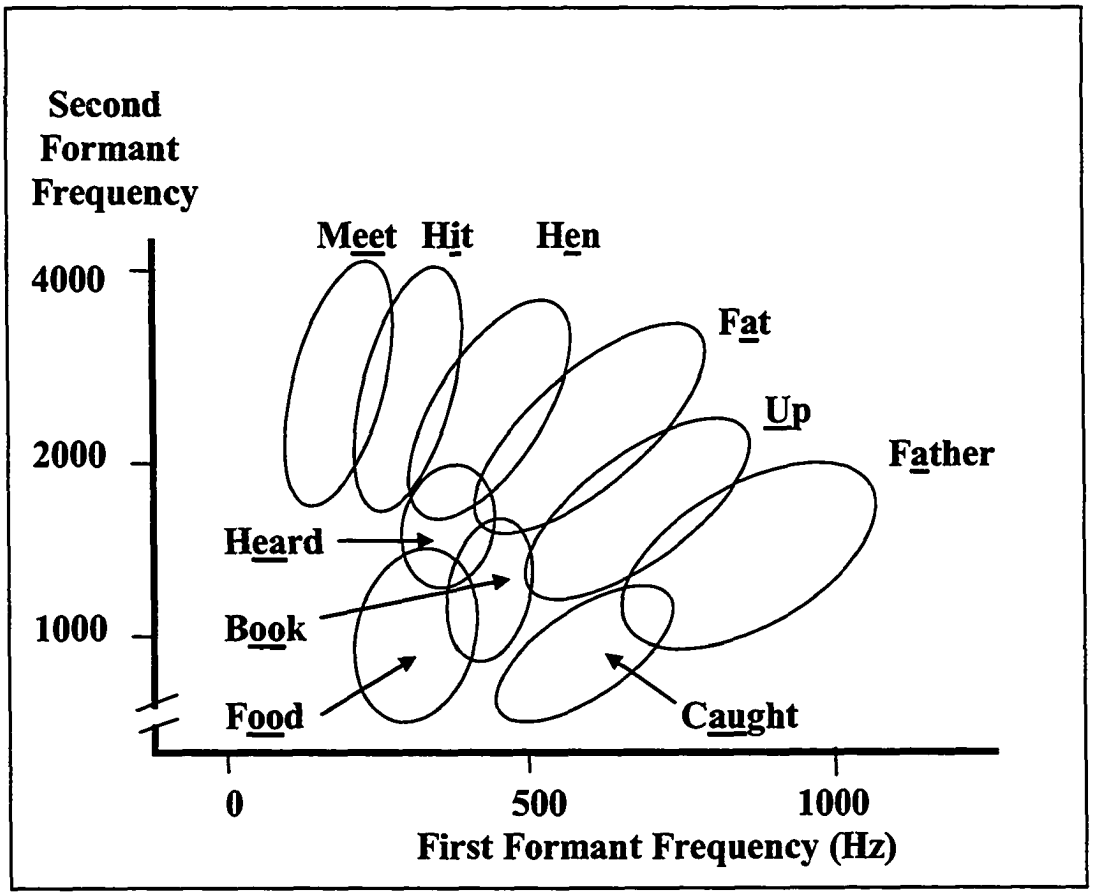

Figure 2-6: Typical F1-F2 loci for English vowels (After Peterson and Barney, 1952)

2.3.3 Parametric models of speech

In order to transmit or recognize speech, one possibility is to solely rely on a raw time-domain representation of the speech in the form of a digitized signal. Since speech signals are quasi stationary in short durations (usually less than 30 ms), there are models that can capture the nature of the signals. Among theses models, predictive methods have proven to be computationally efficient and practically superior to other methods.

- **Linear Predictive Coding (*LPC*)**

Predicting the future values of a signal form from its past values is not something new. In fact, prediction and forecasting techniques have been developed in many areas of operational research (OR) and in control theory to allow for adaptive inventory management systems, production systems and anti-aircraft missiles. [6]. Given a linear system such as the one in Figure 2-7, the goal of the prediction is to estimate the future output of the system from a linear combination of its past input and outputs as it is given in (2-2).

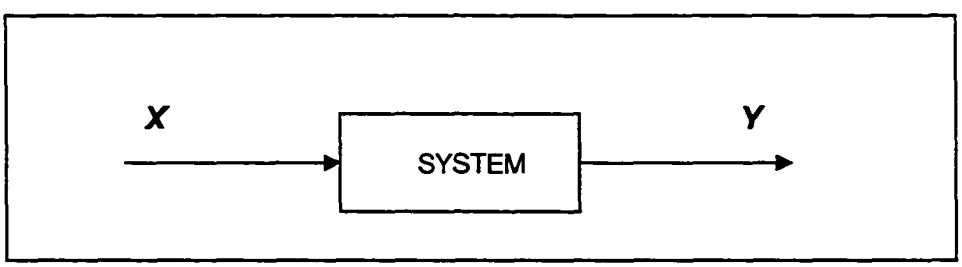

Figure 2-7: Linear system with input X and output Y

$$y(n)= \sum_{j=0}^{q} b(j)x(n-j)-\sum_{i=1}^{p} a(i)y(n-i) \qquad (2\text{-}2)$$

Equation (2-2) shows the output of the system from the linear predictor of its past inputs and past outputs. The factors *a(i)* and *b(j)* are called predictor factors. Many speech modeling systems are describable, at least approximately, by a constant coefficient difference equation such as (2-3): [1]. This system approximation modeling is referred to as linear predictive coding or *LPC*.

$$\sum_{i=0}^{p} a(i)y(n-i)=\sum_{j=1}^{q} b(j)x(n-j) \qquad (2\text{-}3)$$

We can derive an approximate system function from the predictor coefficients. When applying ***Z*** transform, if *H(z) = Y(z)/X(z)*, where *H(z)* is a ratio of polynomials *N(z)/D(z)*, then we can write *N(Z)* and *D(Z)* as in (2-4).

$$N(z)=\sum_{j=1}^{q} b(j)z^{-i}$$

$$D(z)=\sum_{i=1}^{p} a(i)z^{-i} \qquad (2\text{-}4)$$

Here, the predictor coefficients provide immediate access to poles and zeros of *H(z)*, provided that *p* and *q* are chosen properly. Because there can be zeros in both *N(z)* and *D(z)*, this is a mixed pole-zero model.

There are two important derivatives of this general model that are widely used in speech processing:

1. The first one is an *all-poll* model in which, the numerator *N(z)* is constant. In statistical term, this type of model is known as *autoregressive* or *AR* For this type of model $b(j) = 0$ for $j>0$; and the predictor uses only output samples.
2. The second one is an *all-zero* model in which the denominator *D(z)* is equal to unity and the predictor uses only the samples from the input. In statistical term, this model is known as *moving average* or *MA*.

With the same terminology, the general mixed system is called *autoregressive moving-average* or *ARMA*. One of the limitations of the *LPC* coefficients for modeling a speech frame is that it does not accurately model the coupling of the nasal tract and the vocal tract. Although modifying the model to an *ARMA* model can correct this, for speech processing purposes, the all-pole model is used more widely because it is easier to compute.

There are many different methods to compute the model's coefficients. Two well knows methods are Schur's recursion method and Burg's method to calculate the Reflection Coefficients or *RCs*. We used Burg's method to calculate the coefficients as a preliminary stage to construct the feature vectors for training and testing the *HMM* bank based recognizer. Chapters 4, 5 and 6 discuss the experimental settings in more details.

- **Cepstrum**

The cepstrum of a signal is the Fourier transform of the logarithm of its power spectrum [1]. In other word, cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude of the signal. Cepstral coefficients have been shown to be a more robust and reliable feature set for speech recognition compared to *LPC* coefficients [7].

Cepstral coefficients are based on the theory of *homomorphic* system analysis [8]. This analysis is used in signal processing of separate signals that are combined nonlinearly [9]. A speech signal is a non-linear combination of a low-frequency element from the vocal tract and a high frequency element due to excitation. In other word, the spectrum of the speech signal may be given by [1]:

$$X(f) = G(f)H(f) \qquad (2\text{-}5)$$

Where: $G(f)$ is the glottal excitation spectrum
$H(f)$ is the vocal tract transfer function

When we take the logarithm of the two sides, the multiplication becomes addition:

$$T(f) = 2\ln|X(f)| = 2[\ln|G(f)| + \ln|H(f)|] \qquad (2\text{-}6)$$

A sketch of $T(f)$ is shown in Figure 2-8-a. As illustrated, $T(f)$ has two components which are the slow varying of the spectrum envelop and the higher frequency component. These two

components can be separated by filtering or by taking a second Fourier transfer as it is given in (2-7) and shown in Figure 2-8-b.

$$C(q)= F\{T(f)\}=2F[\ln|G(f)|+\ln|H(f)|] \qquad (2\text{-}7)$$

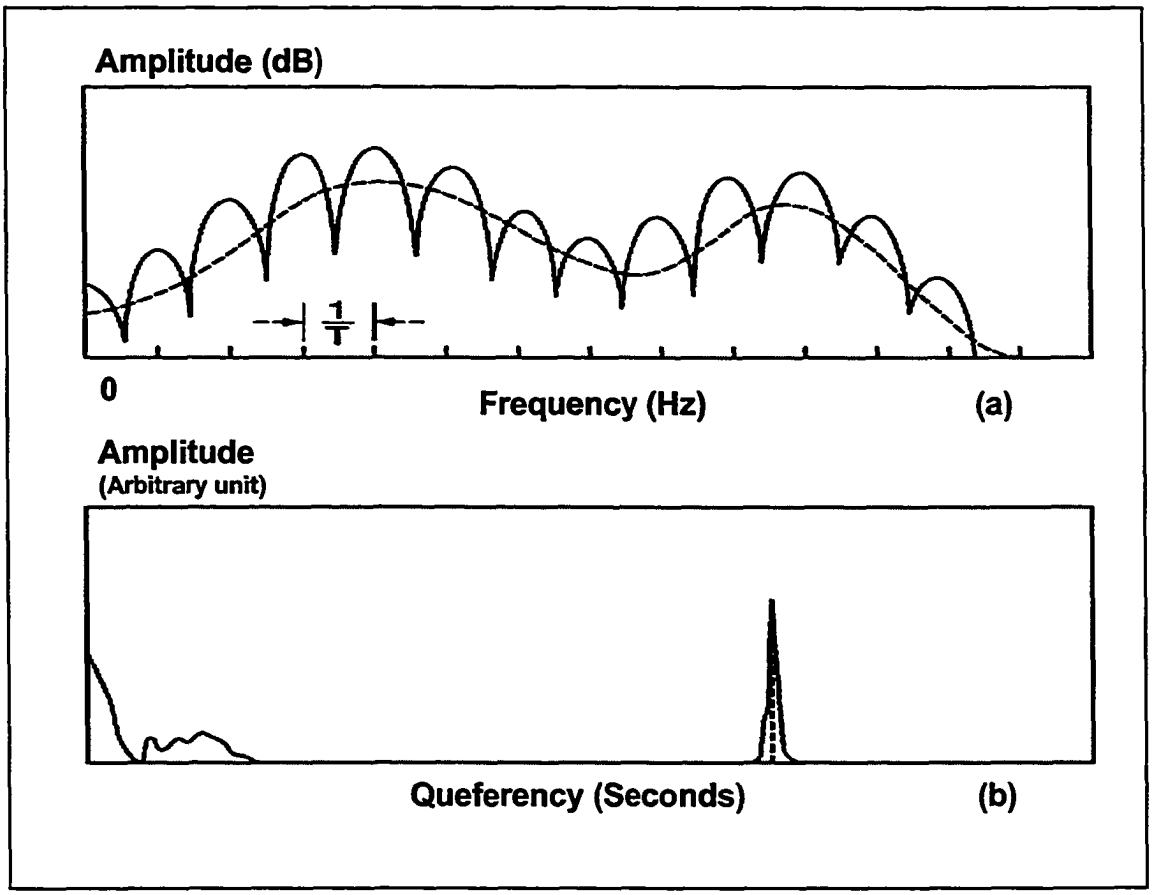

Figure 2-8: Derivation of Cepstrum (a) Logged Power Spectrum and (b) the Fourier transform of (a)

The term cepstrum was first used by J.R. Tukey, in contrast to spectrum. The naming reflects the fact that this transform does not take the signal back to the time-domain, but to a domain

that he called *queferency*. He coined several other terms in reference to this transform such as *liftering*, *rahmonic*, *repiod*, *darius*, *gamnitude* and *saphe*. In our experiments, we used cepstral and Δ-*Cepstral* as the feature vector for training the recognizer. We converted *LPC* coefficients to cepstral coefficients.

2.3.4 Speech transmission

Speech in its natural form transmits through the air. Speakers and microphones are electro mechanical devices that can be used to transmit speech artificially through electronic signals in analog forms. In analog transmission of a speech signal, the required channel bandwidth is proportional to the bandwidth of the voice signal. Although human audible voice can go as high as *20 KHz*, most energy required for intelligible speech is contained between frequency bands of 200 *Hz* to 4 *KHz*.

Analog transmission of speech is becoming less practical as most of the current speech processing devices transmit speech in the form of digital signals. In its simplest form, digitizing speech is accomplished by the coding of its waveform. Encoders that use this form of speech coding are known as waveform encoders.

An example of a speech coding technique is pulse code modulation (*PCM*) which samples speech 8000 times per second. This means that for every second of speech, there are 8000 numbers that carry the information that are embedded in the time-domain representation of the speech signal. In standard *PCM*, each number takes two bytes and as a result, the required bandwidth for transmission is 64 Kbps[1]. *PCM* limits the highest frequency element in the

[1] Kilo bits per second

signal to 4 *KHz* and hence uses a sampling frequency of 8000 samples per second[2]. Many modern encoding techniques can achieve a comparable quality of transmission with much lower bandwidth requirements. Modern speech encoders only require channel bandwidth from 4 to 16 *KHz* to code and transmit the speech. For example, source encoders usually need much less channel bandwidth at the expense of higher coding and processing requirements. Figure 2-9 shows the bandwidth requirement of different coding techniques [10].

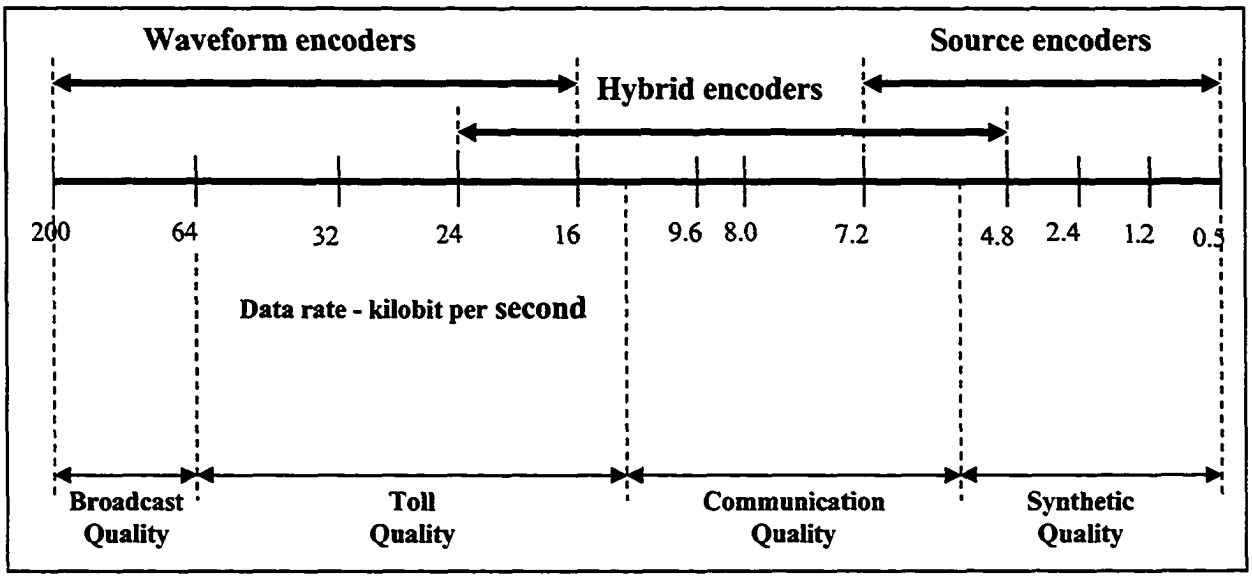

Figure 2-9: Spectrum of speech coding transmission rate and their quality levels

As illustrated in Figure 2-9, source encoders can encode speech with bandwidths as low as 2 *KHz* or even lower. At this bandwidth, although the message is intelligible and clear, the speech sounds artificial and synthetic. Communication quality speech is intelligible but still presents some quality distortion. Toll quality is the standard that is used for wire-line public

[2] This is to satisfy the Nyquist limit and avoid aliasing

switched telephone networks. Broadcast quality refers to wide (7 *KHz*) signals that have been encoded with no perceptible noise.

Unlike waveform encoders, which encode the speech information without any knowledge and assumption regarding how speech is generated, source encoders rely on either modeling the speech production process or on correlation between consecutive samples of the speech signal. Figure 2-10 shows a comparative block diagram of an *LPC* based source encoder and a wave encoder.

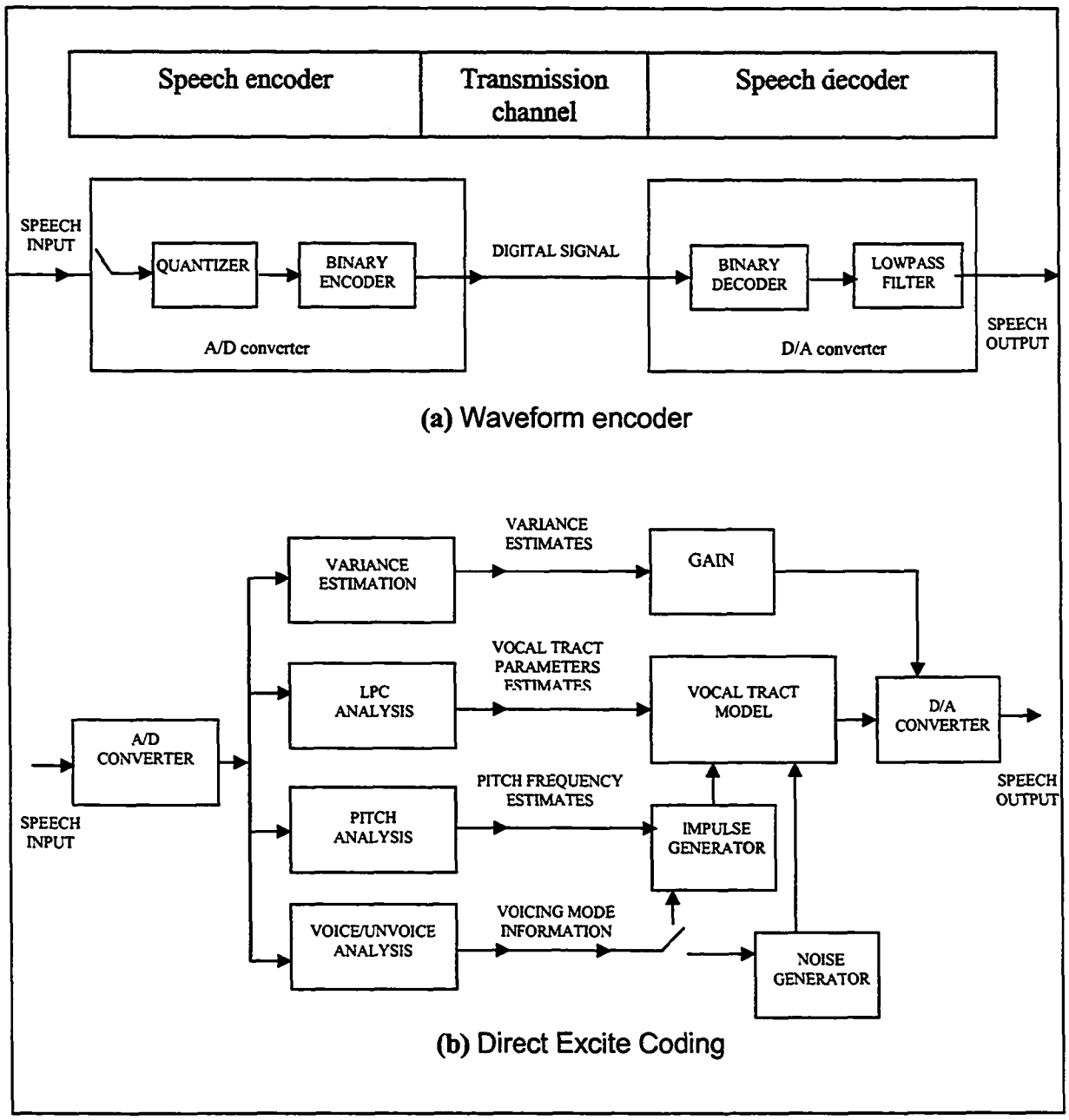

Figure 2-10: Speech encoders: (a) Waveform encoder and (b) source *LPC* encoder

2.3.5 Speech audition and perception

In the human body, the ear is the auditory organ that transforms the mechanical pressure into neural signals that can be interpreted by the brain to decode the voice message or the sound. The ear has three main sections: the outer ear, the middle ear and the inner ear.

Figure 2-11 [11] shows the different sections of the human hearing system.

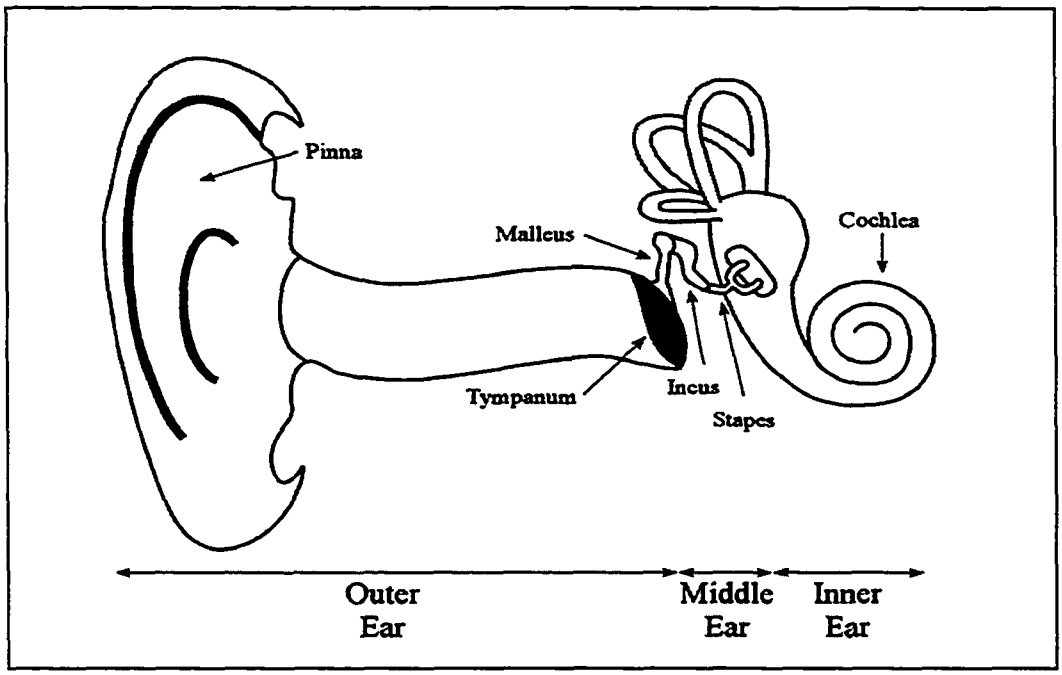

Figure 2-11: Human ear

The outer ear consists of the visible parts of the organ or *pinna*, and the *meatus* which is the auditory canal leading to the eardrum or *tympanum* and is responsible for guiding the air pressure to the middle section of the ear. The outer ear acts also as a resonant cavity which emphasizes on frequencies between 3-5 *KHz* [3]. The middle ear section, consisting of three tiny bones, *Malleus*, *Incus* and *Stapes*, transfers mechanical vibrations generated by the

eardrum to the inner ear. The inner ear consists of the *cochlea* which is filled with fluid and membranes which vibrate in response to the vibrations they receive from the middle ear. Membranes transform the mechanical vibrations of the middle ear, to spatio-temporal disturbances. The most important membrane is the *basilar membrane* which functionally acts as a bank of band-pass filters applied to eardrum excitation [12]. It is also shown that for frequencies between 500 to 800 *Hz*, the vibrating frequency of this membrane is linearly proportional to the frequency of excitation. For excitation frequencies above 800 Hz, the membrane vibrating frequency is logarithmically proportional. Therefore, the *basilar membrane* may be modeled as a bank of filters that are linearly spaced for frequencies below 800 *Hz* and logarithmically spaced for frequencies above 800 *Hz*. The mechanical vibration alongside the basilar membrane causes the flow of ionic current through the inner hair cells which generate impulses within the auditory nerve system which are conveyed to the brain.

Microphones, artificially imitate the function of the human ear. They transform mechanical air pressure that results from traveling speech signals through air, to varying electrical signals. Magnetic microphones transform air movement into movement of a coil in a magnetic field, thereby creating alternating current proportional to the air pressure. *Piezo-electric* microphones use a piece of *piezo-electric* material to change the air pressure to an electrical voltage proportional to the incoming air pressure. Waveform and source encoders are the tools used to extract digital parameters from the raw electrical signals produced by microphones.

The last stage in the speech chain is perception. Artificial methods of understanding speech are discussed in the next section, when Automatic Speech Recognition is introduced.

2.4 Automatic Speech Recognition

If we could imagine computers that are able to communicate with humans in a natural way through verbal communication, it could revolutionize the way we use a computer. Computers, and other devices controlled by computers, could be used in very different environments than are being used now, and by ordinary people who are not adept at using today's computers.

Speech input and output systems have been of interest for research and engineering work for several decades. The progress has been more fruitful in the speech output systems and today, there are successful commercialized Text-To-Speech or *TTS* systems in many applications. These include telephone call centers, toys, and consumer electronic devices such as telephone sets, clocks and even watches. Building a speech enabled input system for computers seems to be a much more challenging and difficult problem to solve.

Speech input systems can be divided into two categories: *speech understanding systems* and *speech recognition systems.* Between the two, speech recognition is the easier task to tackle. There aren't many available speech understanding systems and most likely there won't be an efficient and effective one for many years to come. Speech recognition systems on the other hand, have come a long way over the last few decades and small and medium vocabulary systems have started to be introduced commercially for consumer use.

The problem of automatic speech recognition by machine can be defined as the process of converting speech signals, or the feature that are extracted from speech signals into units of speech such as words, sentences etc.

Research in speech recognition goes back to almost four decades ago. Davis, Bidulph and Balsjek, three Bell Laboratories scientists, built a single user isolated digit recognition system in 1952 [13]. Later on, during the1960s and 1970s and part of the1980s, researchers from across the world made several important advances in the field of speech recognition. In the 1970s, isolated word recognition became a viable and usable technology. Within this decade, pattern recognition techniques were used for speech recognition and Japanese research showed how dynamic programming methods could significantly improve the search techniques in speech pattern matching. During the 1980s, the speech recognition research shifted from template based approaches to statistical modeling methods, especially the *HMM* based approaches [10]. In the 1980 and 1990s, several more sophisticated and larger vocabulary *ASR* research systems were built and more applications were experimented. Among the better known ones are *SPHINX* from Carnegie Mellon University, *BYBLOS* from *BBN* and other systems in *MIT* and *AT&T Bell Labs*. In controlled, low noise and low speaker variability environments, today's *ASR* systems can be used for many more complex applications than the isolated word recognition of the 1970s.

Figure 2-12 shows a spectrum of applications for *ASR* systems in the past, present and the future [14]. Despite all the advances over the last several decades, large vocabulary speech recognition that is tolerant to real life noise levels and speaker variability still remains a research topic in academia rather than an available commercial solution. Speaker accent variability is one of the issues that is becoming an important consideration in many *ASR* applications and will be discussed in detail in Chapter 4.

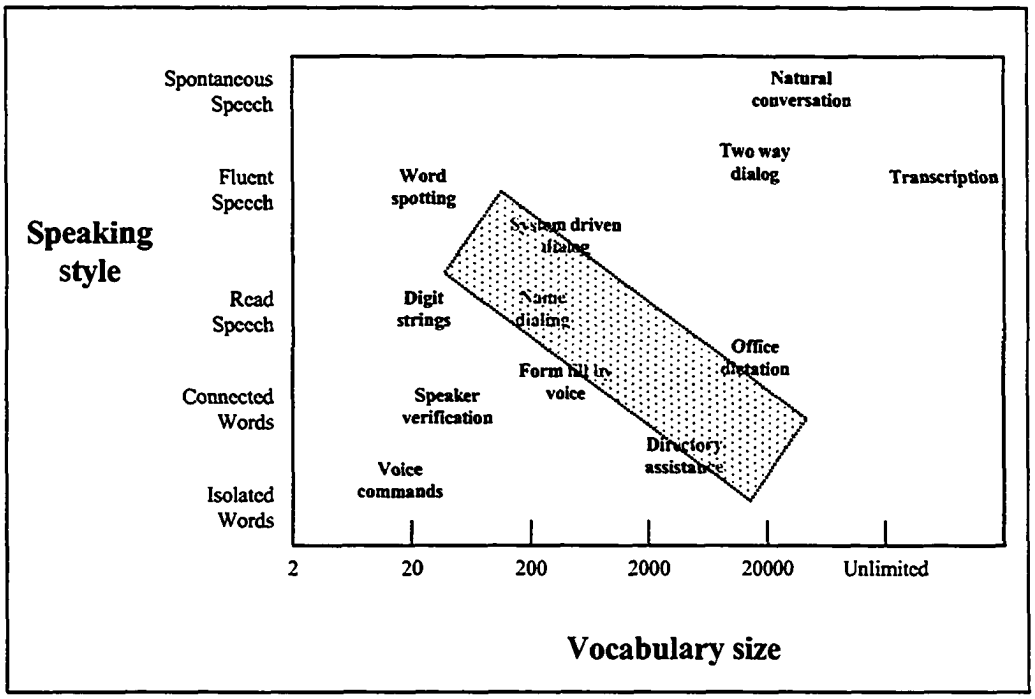

Figure 2-12: Dimensions of Automatic Speech Recognition applications and current capabilities (dotted area)

ASR can be viewed as a pattern recognition task, requiring mapping between each possible input waveform and its corresponding text [15]. Before a pattern recognition algorithm can be applied to the utterance to associate it to one of the classes, several steps have to be taken to prepare the speech signal. Figure 2-13: General ASR block diagram shows the schematics of a speech recognition system. Speech signal observation has to be digitized and then an appropriate feature set has to be extracted from consecutive 20-30 *ms* frames for which the speech signal is assumed to be stationary. Many features are derived from the linear prediction methods that were discussed previously.

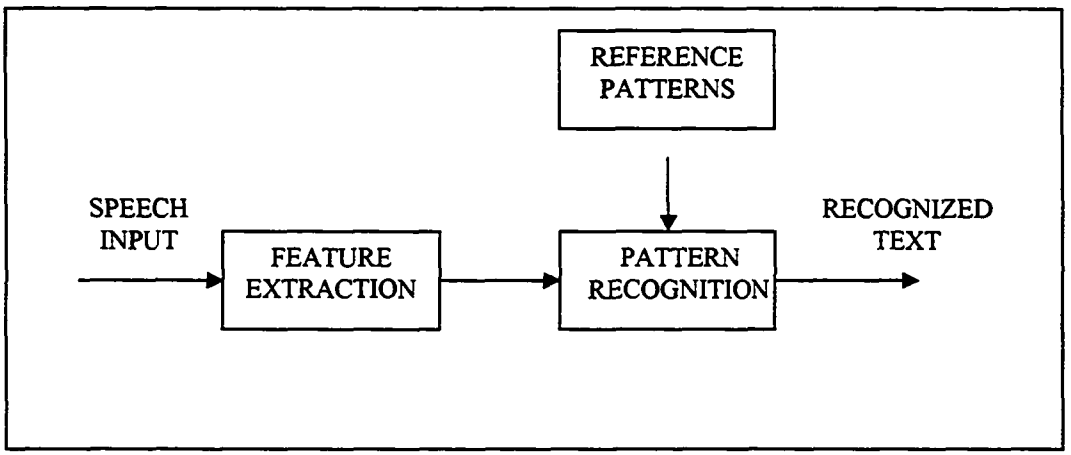

Figure 2-13: General ASR block diagram

2.4.1 Bayes decision rule

If we assume that pattern classes are statistically independent, and the occurrence of one pattern is independent of the occurrences of other patterns, a pattern recognition technique can be built based on Bayes decision theory. This theory deals with random observations of a group of patterns consisting of M classes (C_1-C_M) of events where the goal is to identify which class the observation belongs [16] to. We can assume that *a priori* probabilities of $P(C_1)$ - $P(CM_1)$ are known. This is not a limiting condition since from the training set we can identify these probabilities. In fact, if N is the total number of training observations, then $P(C_i) \cong N_i/N$ for $i=1,2,..M$. The other statistical quantities we assume to be known are the class-conditional density functions $p(x|\ C_i)$, $i=1,2,..M$ describing the distribution of the feature vectors in each of the classes. If these are not known, they can also be estimated from the available training data. From the Bayes conditional rule, we can write the conditional probability as it is given in (2-8).

$$P(C_i|x) = \frac{p(x|C_i)P(C_i)}{P(x)} \quad (2\text{-}8)$$

Where $p(x)$ is the *pdf*[3] of the x and then we can have:

$$p(x) = \sum_{i=1}^{I} p(x|C_i)P(C_i) \quad (2\text{-}9)$$

According to Bayes classification rule, we can then classify the observation x into class k if:

$$P(C_k|x) \geq P(C_j|x) \quad for\ all\ j \neq k \quad (2\text{-}10)$$

For context-dependent classification, where successive feature vectors are not independent, there are other approaches to build the pattern recognition component of an ASR system such as Hidden Markov Model.

2.4.2 Temporal Template Matching

As was the case for many early word and phoneme recognition systems, temporal template matching uses temporal references extracted over the duration of the word or phoneme, for pattern matching. Recognition is then carried out by matching the unknown utterance to the stored templates and determining the one that demonstrate the closest resemblance. When the templates are chosen to represent words, the errors due to segmentation or classification of smaller units of speech such as phonemes are avoided, but when the number of words increases beyond a few hundred words, storing and handling the reference templates become

[3] Probability Distribution Function

impractical [17]. Good examples of this type of recognizer are isolated word and digit recognition systems that were built in the *1970s* and *1980s*. In this type of recognizer, pauses between words simplify recognition because they make it easier to determine endpoints, i.e., the start and endpoint of each word. Several different feature vectors are successfully used in temporal template matching speech recognition. These include *zero-crossing rate*, spectral details such as DFT spectrum, *formants frequencies* (especially the first three), *LPC* parameters and *filter-bank* outputs [1]. In building the templates, short frames of 20-30 *ms* are used to produce a single feature vector to minimize the effect of non-stationary behavior of speech signal. The sequence of these vectors form the templates. The reference template is obtained during the training of the system, where a significant amount of averaging is used to form the reference template from the extracted feature vectors [8]. In the test stage, in many cases, *Euclidean* distance such as (2-11) is used to measure similarity between reference and test templates.

$$d(x,y)=\sqrt{\sum_{i=1}^{N}(x_i-y_i)^2} \qquad (2\text{-}11)$$

The main issue with this type of recognizer is the intra-speaker variability of speech signals in speaker dependent and inter-speaker variability of speech in speaker independent recognition. Duration of vowels or consonant phonemes, and therefore words, in conversational speech is usually variable and rarely fixed. Since the length of the template in (2-7) is proportional to the utterance, and the vectors used to calculate the similarity distance must have the same length of N, it is difficult to compare two vectors with different lengths. This issue forced researchers to use some form of time normalization techniques to make the

two vectors similar in length. The phonemes in the *TIMIT* database that we used for the experimental part of this research display vividly this random variation. For example, the vowel /i/, when examined for all 630 speakers in the database, shows a mean duration of 89.9 ms and standard deviation of 96.4 ms. The maximum registered length for this phoneme is 428.2 ms and minimum length is 23.3 ms. Another example is the phoneme /d/ which shows a mean of 23.0 ms, a standard deviation of 26.1 ms, a maximum duration of 120 ms and a minimum duration of 2.4 ms. When observing all the phonemes, it becomes apparent that vowels and nasals show a higher degree of variability than do most fricatives [8]. This makes it impractical to use a linear compression method to deal with utterance length variability and therefore a non-linear time alignment method is required. Dynamic Time Warping [18] is the method that has been successfully used for time normalization of utterances. In this process, the time axis of the test utterance is non-linearly distorted, or warped, to bring its features in line with the reference template. Figure 2-14 [1] shows a typical re-mapping function.

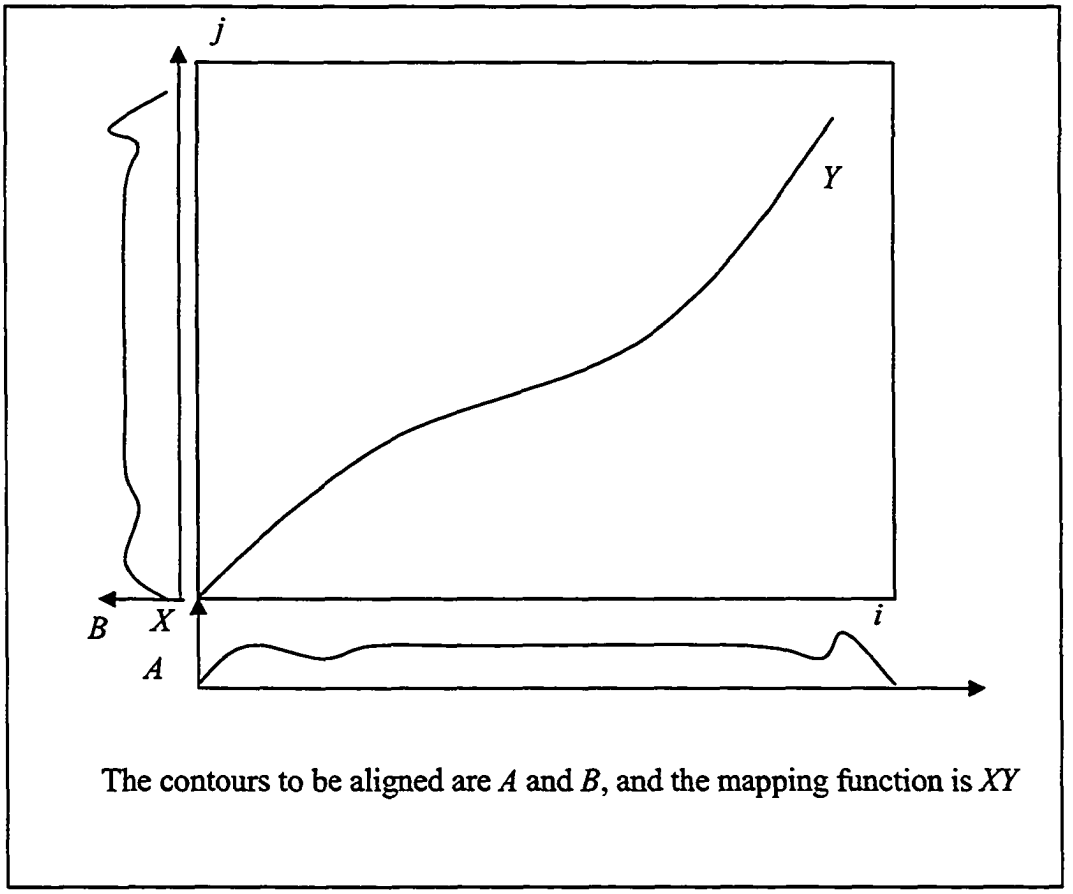

Figure 2-14: Dynamic Time Warping.

2.4.3 Neural Networks

The performance of today's speech recognition systems are far below that of humans [17]. Artificial neural networks, or *ANN* for short, are an attempt to emulate the way the human brain works for pattern recognition. They offer a potential for building massive parallelism based pattern recognition systems that may someday deliver better and more capable *ASRs*. Much of the progress in the design and theory of neural networks has been in the area of signal processing. Generally speaking, an *ANN* is a signal processing system composed of a large number of simple processing units called neurons. Neurons are arranged in layers and are connected to each other with links. The weights associated with links act as signal multipliers. A very interesting property of neural networks is that they can adapt, by changing their connection weights, to a particular condition [19]. Given adequate training data, *ANNs* can generate complex decision boundaries that can solve an arbitrary complex classification task [20].

The Neuron Model

All neural networks that have been proposed over the years have been based on a common building block known as the neuron and have been modelled after biological neurons. The most widely used neuron model is based on McCulloch and Pitts work as it is illustrated in Figure 2-15 with a comparison to a biological neuron. It consists of a computational block with several input connections $x(t)$ and one output connection $y(t)$. The computational block consists of two separate functions, a net function of $u(x, w)$ and an activation function of $\varphi(\zeta)$. The net function aggregates all the inputs and their corresponding weights and

delivers them to the activation function which in turn produces the output of $y(t)$. In the 1950s Rosenblatt [21] suggested a special case of a neuron called a perceptron. A perceptron has a linear net function and a threshold activation function as in (2-12).

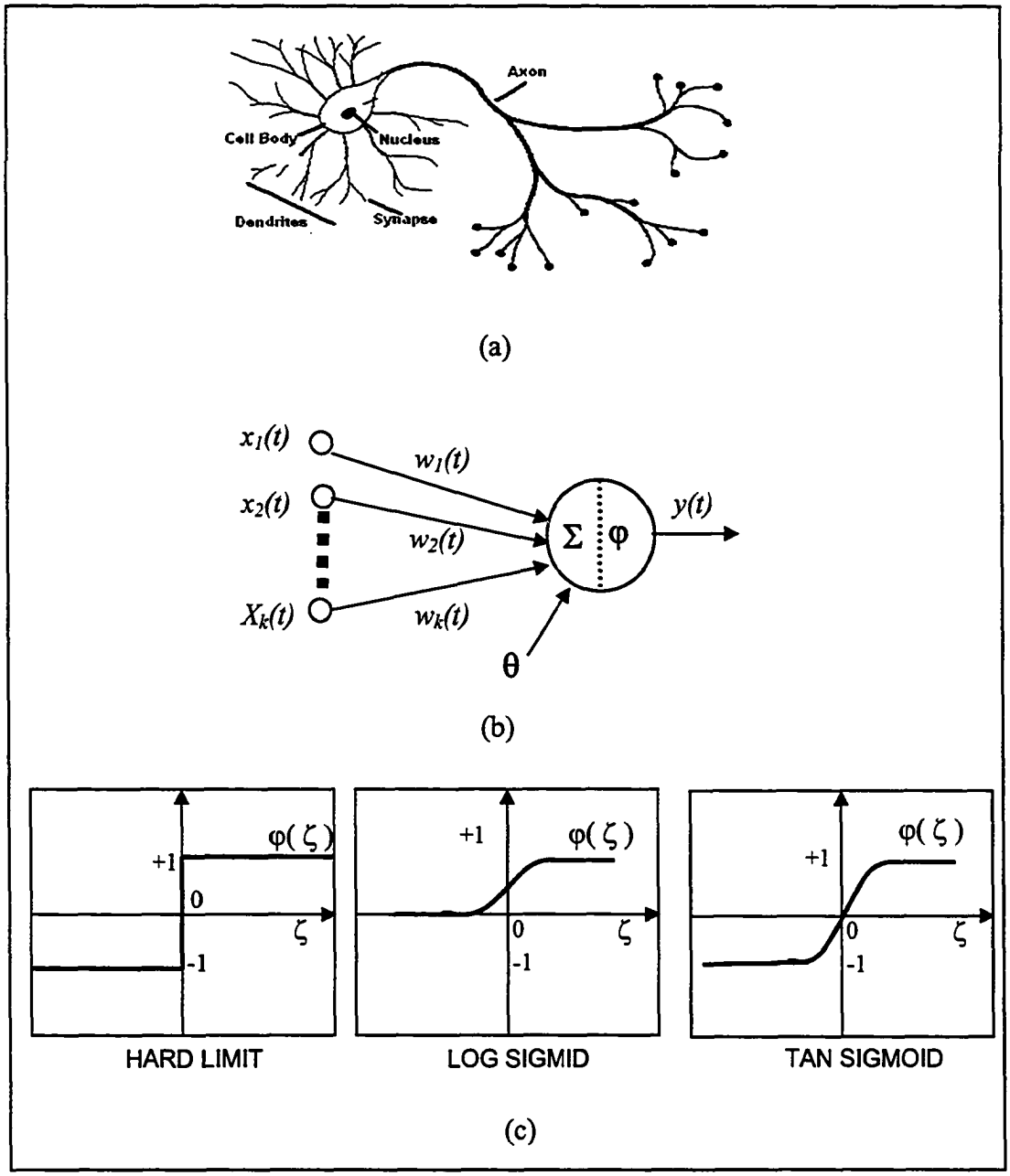

Figure 2-15: (a) a biological neuron, (b) McCulloch-Pitts model for artificial neuron and (c) different activation functions

$$u = \sum_{i=1}^{K} x_j(t) w_j(t) + \theta \qquad (2\text{-}12)$$

θ is called bias and is used to model the threshold. Through iterative training, the perceptron can learn how to adapt its decision boundaries to cluster the input vectors. For example, in the case of training a perceptron to classify vectors into two categories, the following steps can be used [22]:

1. Initialize the weight and thresholds:

   Set $w_i(0)$ for $i = 1..K$ *and* θ to small random variables. Here $w(t)$ is the weight from input i at time t and θ is the threshold in the output node.

2. Present new input and desired output:

   Present new continuous input $x_0, x_0 \ldots x_{K-1}$ along with the desired output $d(t)$

3. Calculate actual output according to (2-13):

$$y(t) = \varphi \left(\sum_{i=0}^{K-1} w_j(t) x_j(t) + \theta \right) \qquad (2\text{-}13)$$

$\varphi(\zeta)$ is one of the non-linear activation functions in Figure 2-15: (a) a biological neuron, (b) McCulloch-Pitts model for artificial neuron and (c) different activation functions

4. Adapt weights according to (2-14):

$$w_i(t+1) = w_i(t)+\eta[d(t)-y(t)]x_i(t), \qquad 0\le j\le K-1 \qquad (2\text{-}14)$$

$$d(t) = \begin{cases} +1 \text{ if input from class } A \\ -1 \text{ if input from class } B \end{cases}$$

η is a positive fraction less than 1 that is used to control the learning rate. $y(t)$ is the desired output at time t.

5. Repeat by going to step 2 until the weights converge for the given input

It is important to note that the weights will be adjusted if an incorrect decision is made. Given a suitable training sequence, a perceptron is capable of classifying any linearly separable pairs of classes. Linearly separable classes are those whose probability density functions do not overlap and that can be separated by a hyperplane as shown in Figure 2-16.

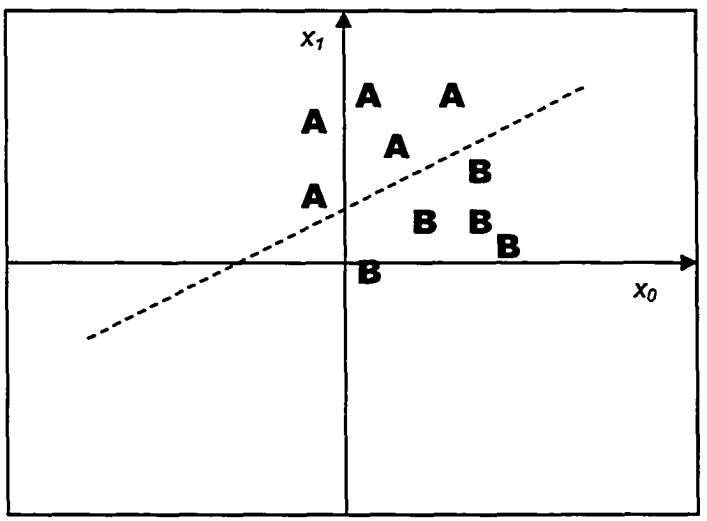

Figure 2-16: A single layer perceptron that classifies an input vector into two classes denoted *A* and *B*

MLP: Multi-Layer Perceptron

A multi-layer perceptron is a feed forward layered network of neurons. Each neuron has its own non-linear activation function that is often continuously differentiable [23]. Each *MLP* consists of an input layer, an output layer and at least one middle layer also known as a hidden layer.

Figure 2-17 compares an *MLP* with different numbers of hidden layers and its ability to generalize a periodic function.

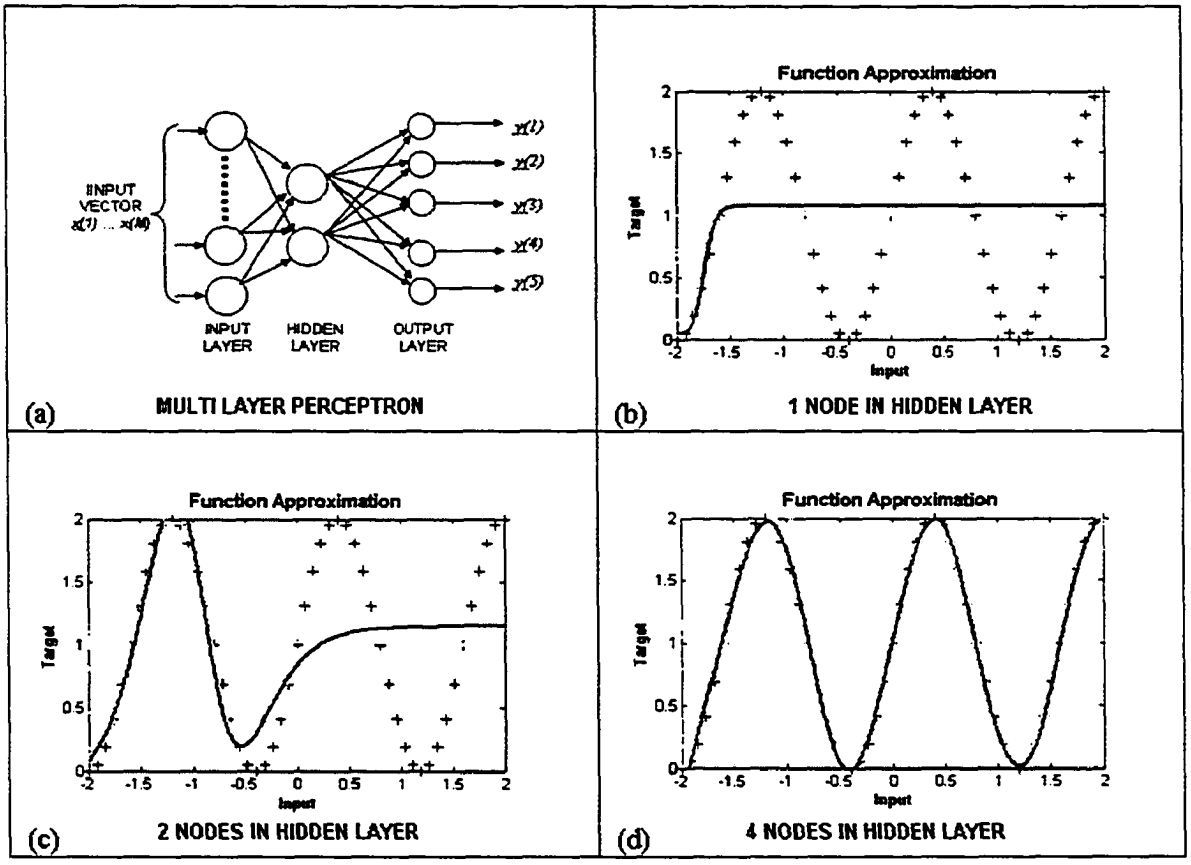

Figure 2-17: Comparing generalization ability of a network with different number of nodes in its hidden layer

Figure 2-17: (a) is a multi-layer perceptron with 2 nodes in its hidden layer. Part (b) shows function approximation for a network with one node in its hidden layer. Part (c) shows function approximation for the same network but with two nodes in its hidden layer and part (d) is the function approximation with 4 nodes in the hidden layer

Having multi layers, allows an *MLP* to robustly adapt to a nonlinear function.
Figure 2-17 (b), (c), and (d) show how increasing the number of nodes in the hidden layer, increases a network's adaptability to higher levels of non-linearity.
Although increasing the number of nodes in a hidden layer allows the network to adapt to non-linearity better, it should be noted that when designing a network, its size should be kept as small as possible [16]. If the size of a network, mainly the number of hidden layers and the number of nodes in the hidden layers, is too large, the network loses its capability to generalize and will have difficulty classifying the observations that have not been presented to the network before. This is called over-fitting and it happens because increasing the size of a network, causes it to adapt to the training set so well that it has difficulty classifying the test observations.
Figure 2-18:Over-fitting and its effect on network's ability to approximate a function shows how over-fitting can negatively affect a networks ability to approximate a function. Part (a) is the function to be approximated, part (b) is the approximation by a neural net with 2 hidden layer, part (c) is the function approximate ion by a neural net with 6 hidden layers and part (d) is the approximation by a neural net with 9 hidden.

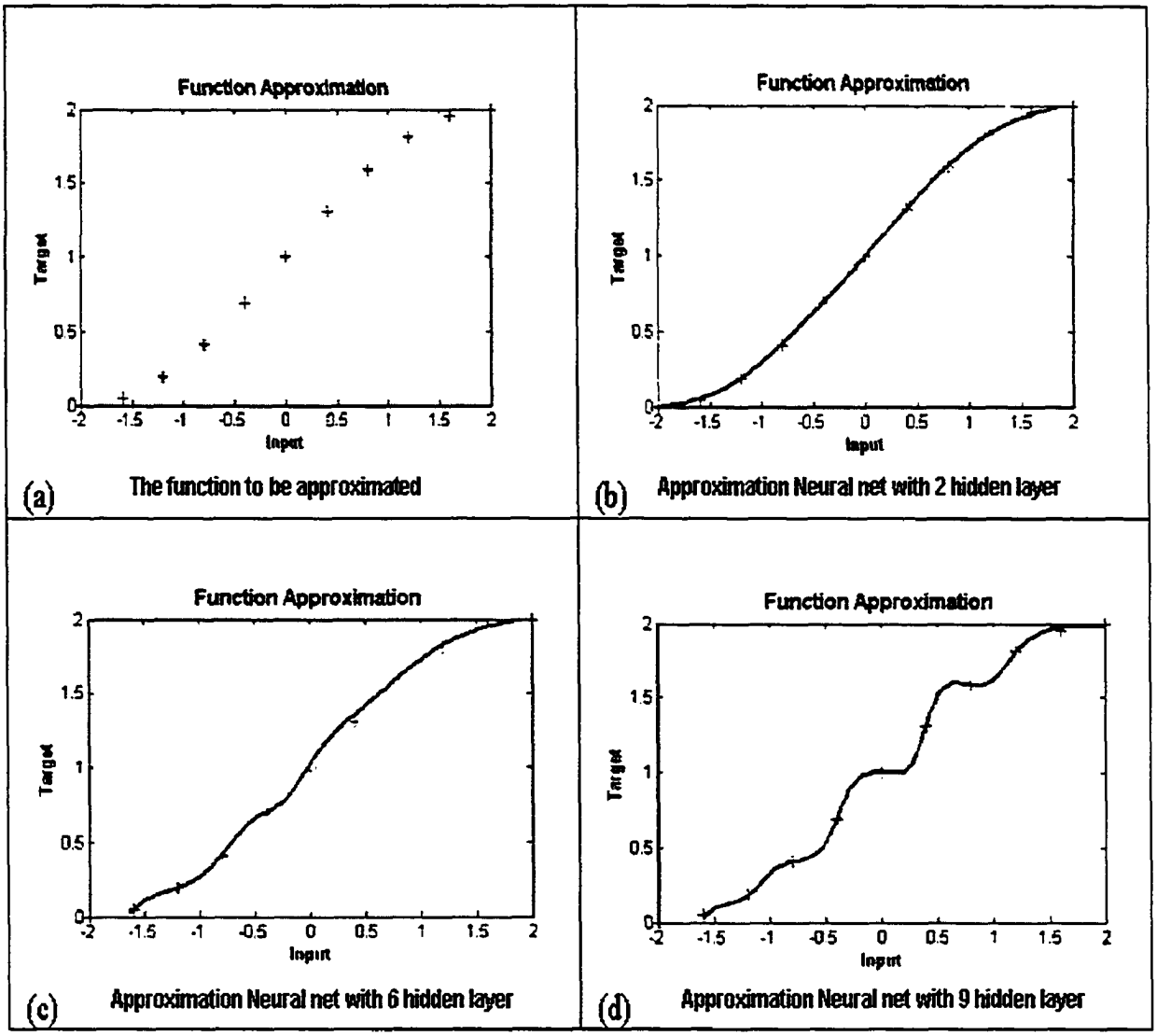

Figure 2-18:Over-fitting and its effect on network's ability to approximate a function

Over training can also cause the network to lose its ability to classify the test data correctly. This occurs when the network undergoes a large numbers of training iterations which, are also known as epochs.

The size of the network should be such that it is (a) large enough to learn what makes the feature vectors within each class "similar" and (b) small enough, with respect to the size of the input training set, so as not to be able to learn the underlying differences among the data of the same class [16]. Different methods have been proposed to determine the size of a network. The most widely used ones are:

1. *Analytical methods*: These methods use algebraic or statistical techniques to determine the number free parameters
2. *Pruning methods*: A large network is initially selected and then the number of free parameters is successively reduced.
3. *Constructive methods*: In contrast to pruning methods, here a small network is initially chosen and neurons are successively added, based on an appropriately selected training rule for each added neuron.

The rule of thumb to prevent over-fitting is that the number of free parameters in the network should be (significantly) less than the number of examples [24].

2.4.4 Hidden Markov Model

Different sources of uncertainty are present in speech and speech recognition such as intra and inter-speaker variations, contextual effects, and homophone words and thus speech signals can be modeled using a stochastic approach. [25]. The most widely known stochastic method for speech recognition is the Hidden Markov Model or *HMM*. *HMM* based approaches are used in most modern *ASRs* today. They are the alternative strategy in building a pattern recognition system and they make use of the stochastic model of speech production [1]. The hidden Markov models have been explored by many researchers but the initial research work by Rabiner [26], Bahl et al and Levinson are cited most frequently.

Hidden Markov models are finite-state automata. They can be a powerful tool when they are used as part of a finite state network to represent a language and its components such as phonemes, words and grammars, specifying the sequence of words in a language [14]. For large vocabulary speech recognition these networks are very large and an efficient search mechanism is essential in determining which path produces the observed speech signal with the maximum likelihood. In the early development of speech recognition systems, Dynamic Programming (*DP*) methods were used for dynamic time warping techniques in temporal template matching systems [1]. Dynamic programming is also used as a common search method in hidden Markov models and is known as the *Viterbi* algorithm [7].
Hidden Markov models are based on Markov processes. A Markov process is one that moves from one state to the next depending only on the previous *n* states; *n* is referred to as the

order of the process. In its simplest from, the choice of state is based purely on the previous state. In a hidden Markov process, there are two sets of states.

The first one is the observable states and the second is the hidden states. This tool allows recognizing a hidden pattern from its set of observable states. To understand the hidden Markov models in the context of speech recognition, it is intuitively easier to think of the model as the vocal tract, and of the hidden states, as different positions of the vocal tract. The output, or observable states, could be thought of certain sounds or acoustic events associated with each articulatory position [27]. For example, in the case a word recognition system, these may correspond to distinct phonemes. With each state, there is an associated probability density function $p_i(X)$ which denotes the probability of observing vector X in state i. To use the temporal order of events in speech, in *HHM* based speech recognition systems, only left-to-right state transitions are allowed [3]. Figure 2-19 shows a 5 state *HMM*.

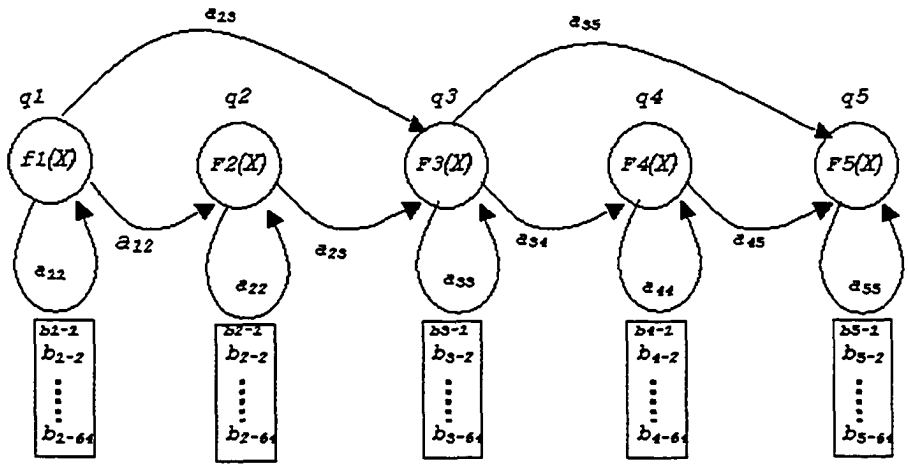

Figure 2-19: A 5 state Hidden Markov Model

To demonstrate how hidden Markov models can be used as a pattern recognition system, an example of an isolated digit recognizer is examined [27]. Figure 2-20 shows the block diagram of such a system. The speech signal is processed and *LPC* coefficients are calculated, then a vector quantizer (*VQ*) is used to reduce the dimension of feature vectors. It also acts as a transfer function from the continuous space of *LPC* coefficients to the discrete space spanned by the codes in the *VQ*'s codebook. This transfer consequently reduces the complexity of the *HMM* model and simplifies the training.

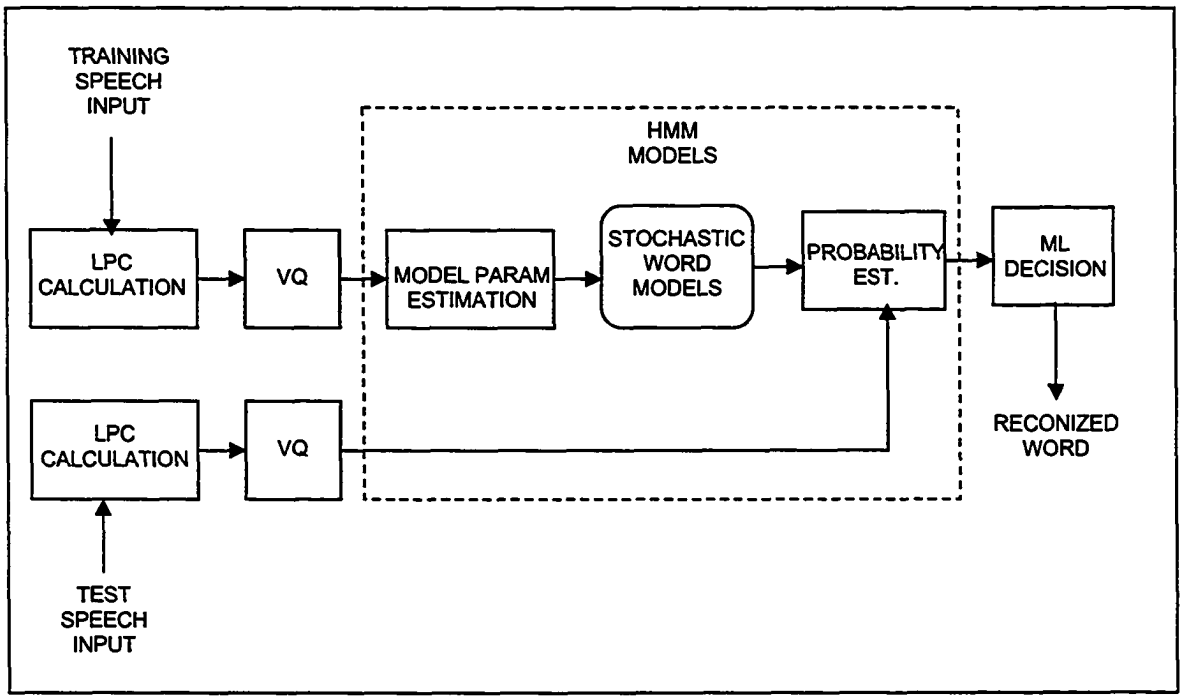

Figure 2-20: An HMM based isolated digit recognition block diagram

For the training, it is assumed that *100* instances of each digit spoken by different speakers are collected and passed through the *LPC* calculations and the vector quantization. It is also

assumed that the codebook size of the *VQ* is set to *64* and the model is a *5*-sate *HMM* as shown in Figure 2-19. We need to estimate the parameters of this model which are the *A* and *B* matrices. *A* is the transition probability matrix and *B* is the symbol probability matrix. The symbols are the vector quantizer's indices, v_k, $1 \le k \le 64$. *A* and *B* are defined as in (2-15), (2-16).

$$A=[a_{ij}]$$
$$\text{where} \quad a_{ij}=\Pr[q_j \text{ at } t+1 \mid q_i \text{ at } t] \quad \text{for} 1 \le i,j \le 5 \tag{2-15}$$

$$B=[b_{jk}]$$
$$\text{where} \quad b_{jk}=\Pr[v_k \text{ at } t \mid q_j \text{ at } t] \quad \text{for} 1 \le i,j \le 5 \text{ and } 1 \le k \le 64 \tag{2-16}$$

Once the vector quantization is performed on *LPC* coefficients, the data will be a series of observations for each word, ***O***, in which we have 100 utterances as in (2-17). Each $O^{(k)}$ then can be written as in (2-18).

$$O = O^{(1)} O^{(2)} \ldots\ldots O^{(100)} \tag{2-17}$$

$$\text{each } O^{(k)} \text{ is a sequence of } \{O^{(k)}\}_{t=1}^{T_k} \text{ of symbols} \tag{2-18}$$

The probability of P_k, that $O^{(k)}$ was generated by a model with *A* and *B* as parameters is given in (2-19).

$$P_k = \Pr[O_k \mid A,B] = e_1 \mathbf{B}_1^k \; A\mathbf{B}_2^k \; A\mathbf{B}_{t_k}^k \; e_5 \tag{2-19}$$

The constants e_1 and e_5 are the first and fifth unit vectors showing that the model starts at state q_1 and ends in state q_5. $\mathbf{B}_1^k$ is the diagonal matrix where the j^{th} non-zero entry is b_{jr} where $r = \boldsymbol{O}^{(k)}$. We should note that we assume the utterances are independent and the probability of the entire training sequence is given by (2-20)

$$P = \Pr[O \mid A,B] = \prod_{k=1}^{100} P_k \tag{2-20}$$

To train the model, we need to find a set of model parameters A and B so that the P in (2-20) is maximized for the training observations $\boldsymbol{O}$. From the *Baum-Welch* optimization algorithm in each iteration, we have new parameters $\overline{A},\overline{B}$ as in (2-21) and (2-22).

$$\bar{a}_{ij} = \frac{a_{ij}\dfrac{\partial P}{\partial a_{ij}}}{\sum_{k=1}^{5} a_{ik}\dfrac{\partial P}{\partial a_{ik}}} \quad 1 \le i,j \le 5 \tag{2-21}$$

$$\bar{b}_{jk} = \frac{b_{jk}\dfrac{\partial P}{\partial b_{jk}}}{\sum_{r=1}^{64} a_{jr}\dfrac{\partial P}{\partial a_{jr}}} \quad 1 \le j \le 5 \; and \; 1 \le k \le 64 \tag{2-22}$$

When training the models, $\Pr[O|\bar{A},\bar{B}] \geq \Pr[O|A,B]$ thus repeated training iterations most likely converge to a local maximum of *P*. Once the training is complete, we also have a set of parameters for each class denoted by (A_0, B_0), (A_1, B_1) ... (A_9, B_9). To present a set of unknown observations to the trained *HMM*, we have an observation sequence of ***O*** that is derived from an uttered digit. This can be any of $w_1, w_2, \ldots w_9$. We compute the likelihood probabilities of $P_i = \Pr[O|A_i,B_i]$ $for 0 \leq i \leq 9$. The unknown utterance is classified as class *i* **iff** $P_i > P_j$ for all $0 \leq i, j \leq 9$ and $i \neq j$.

2.4.5 Issues in Automatic Speech Recognition

As explained in Chapter 1 and the first part of Chapter 2, researchers in automatic speech recognition face several challenges. As a summary and a pretext to the next three chapters here is a list of major issues that currently exist in *ASR* systems:

1. **Speaker variation:** people tend to sound different when they speak. In every individual's voice there is a speaker dependent voice-print that differentiates him/her from other talkers, which is how we can recognize different people from their voice. In fact, this is the basis of the speaker recognition systems that can identify people based on a sample of their voice. Although this is useful for speaker identification, it makes it more challenging for speech recognition systems to identify the same word spoken by different speakers.

2. **Ambiguity:** there is not a unique direct mapping between acoustic parameters and phonemic parameters. This makes it very difficult to recognize words merely from acoustic knowledge gathered from the utterance. In the case of conflicts, the system needs to refer to language structure to resolve the ambiguity. For example, recognizing homophonic words – which are different words that are similar in how they sound – is very difficult without considering language knowledge in the recognition process.

3. **Variations in individual speech:** to make matter even more challenging, humans never say a word exactly in the same way twice. This is due to several reasons. First is carelessness; people only pay enough attention to the way they speak to convey their purpose. Some words are sometimes uttered with different syllables. Short words are often eliminated or reduced to grunts. The second factor is co-articulation; context affects the phonetic characteristics of spoken sounds. The third reason is temporal variation; one important source of variability in speech comes from different duration of spoken words or even time alignments of phonemes within words for different speakers.

4. **Noise and interference:** noise is well known to be one of the most challenging issues in speech recognition systems. Even humans sometimes can not overcome the noise degradation problem even though, on top of acoustic information, humans use context as well as visual clues to detect spoken words.

5. **Accent and native language:** native language can affect the way we pronounce words in a second language a great deal. The phonetic differences between different languages, makes it difficult to speak a word in a second language the way native speakers

pronounce it. For example, since the phoneme */p/* does not exist in Arabic, Arab-English speakers tend to pronounce */p/* as */b/*. Even speakers of the same language with different accents pose a challenge to many *ASR* systems. For example, an *ASR* engines that is trained with a southern American accent, does not deliver the same performance when recognizing speakers from other regions of the US. To address this issue some systems have been built and fine tuned toward a specific speaker. These systems adapt to one particular speaker over time. This lowers the error rate but at the expense of long and tedious user training. A problem occurs when these types of *ASR* systems are used by other users, they do not perform well until they go through another lengthy training phase. These are called speaker dependent *ASR* systems. On the other hand speaker-independent *ASR* systems do not perform as well as speaker dependent ones, but offer a much better user experience since the system is ready to operate out of the box.

With a closer look, in practice there are many occasions in which *ASR* systems are used in environments, such as automobiles, house interiors and so on, that are shared by a group of people (e.g. a family) that represent common speaker clustering behaviors. For example, with the growing population of people that are migrating from one place to another, such a clustering can occur when the users are from a common ethnic group, having the same native language which is different from the language of the *ASR* system.

An *ASR* system that can identify the native language (or the accent) of these users, can conveniently and quickly adapt to them. This adaptation is much less expensive than a full function convergence of the recognition engine to a broad range of speakers and in our proposed solution can take place promptly.

The accent problem and its impact on *ASR* performance is the subject of this thesis and will be dealt with in details in the coming chapters.

2.5 Summary

This chapter provided a background overview of speech processing and speech recognition. This background covers many of the techniques and tools that we used in our experiment to validate the accent degradation assumption and to propose a solution to address it. It started with explaining the speech chain. The speech chain starts with speech production, followed by speech transmission and finally speech reception and perception. Human Speech production organs, their role and their function were discussed. The human auditory system was also explained. Machine based speech production, transmission, reception and processing were reviewed. Different methods of speech recognition including artificial neural networks and hidden markov models were discussed and at the end of the chapter a review of current issues in speech recognition systems were outlined.

CHAPTER 3 Experimental Setup

In this chapter, experimental setup and the tools that we used to carry out our research work are introduced. These include the *TIMIT* database that we used in our experiments, the pre-processing that we did to prepare the *TIMIT* database, the feature extraction, sound editing and analysis tools and other related setup considerations that we made.

3.1 The TIMIT database

We used the *TIMIT* database for both parts of our experiment including the *HMM* bank recognizers training and testing for studying the effects of accents in speech recognition and the solution we proposed to address it. We also used the *TIMIT* corpora in our accent identification experiment for training and testing the neural network based accent classifier. *TIMIT* is a the result of collaborative research by *Texas Instrument*, *MIT* and *DAPRA* which sponsored the activity. The corpus design was a common work of *TI*, *Stanford* and *MIT*. Its goal was to provide the database to researchers in the area of automatic speech recognition.

The *TIMIT* corpus database contains spoken sentences from 630 speakers from seven different dialect regions. An eighth group of sentences were collected from army brat that moved around.

Table 3-1 shows the dialect regions and

Table 3-2 shows the distribution of the corpuses among dialect regions.

There are three different sentence types in the *TIMIT* database. The first type is the dialect type or, as it has been referred to, the *SA* sentence. The corpus in the *SA* sentences is designed so as to emphasize dialectal differences. We used the first group in *SA* sentences or *SA1* in our research work. The second type of sentence in the database consists of compact or *SX* sentences. This type of sentence is designed to provide good coverage of pairs of phones. Each speaker reads five of these sentences and in total each text is spoken by 7 different speakers.

Dialect Region	Description
Dialect Region 1	**New England**
Dialect Region 2	**Northern**
Dialect Region 3	**North Midland**
Dialect Region 4	**South Midland**
Dialect Region 5	**Southern**
Dialect Region 6	**New York City**
Dialect Region 7	**Western**
Dialect Region 8	**Army Brat (moved around)**

Table 3-1: Eight different dialect regions in TIMIT database

Dialect Region	Male	Female	Total
1	31 (63%)	18 (27%)	49 (8%)
2	71 (70%)	31 (30%)	102 (16%)
3	79 (67%)	23 (23%)	102 (16%)
4	69 (69%)	31 (31%)	100 (16%)
5	62 (63%)	36 (37%)	98 (16%)
6	30 (65%)	16 (35%)	46 (7%)
7	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)
Total	438 (70%)	192 (30%)	630 (100%)

Table 3-2: Distribution of *TIMIT* database over the dialect regions

The third type of sentence is the phonetically diverse or *SI* sentence. This type is an extra group and is meant to add diversity to the type of sentences.

Table 3-3 shows the distribution of sentence types in the database.

Sentence Type	Sentences	Speakers	Total	Sentences/Speaker
Dialect (SA)	2	630	1260	2
Compact (SX)	450	7	3150	5
Diverse (SI)	1890	1	1890	3
Total		2342	6300	10

Table 3-3: Distribution of TIMIT sentence types

The database is divided into a training set and a test set. The corpora in the database are completely transcribed. The speech samples are stored in Wave format with a special 1024 byte heading. The *TIMIT* header format is a *NIST SPHERE* format and is different from the standard Microsoft wave file header. For each speech file, there are transcription files to identify the begin-sample and end-sample of words and phonemes. There are also text files that describe the caption of the speech files.

Table 3-4 explains *TIMIT's* different file formats.

File Type	Description
(.wav)	***SPHERE*-headered speech waveform file. The Sphere header is 1024 bytes**
(.txt)	**Associated orthographic transcription of the words the person said.**
(.wrd)	**Time-aligned word transcription.**
(.phn)	**Time-aligned phonetic transcription.**

Table 3-4: File types in *TIMIT* database and their description

As we will see, we developed a phoneme extraction utility to parse the <.phn> file to find the beginning and the end sample count of each phoneme in *SA1* sentences in <.wav> files in the *TIMIT* database. We later extracted samples associated with the phonemes that we used in our experiment in separate files. Later on we used these phoneme files to calculate the *LPC* feature vectors for training and testing the accent classifiers.

Here are examples of the <*.txt*>, <*.wrd*> and <*.phn*> files for an *SA1* sentence:

- *Orthography (.txt):*

```
0 61748 She had your dark suit in greasy wash water all year
```

- *Word label (.wrd):*

```
0      7470   h#        28360 30960 in
7470   11362  she       30960 36971 greasy
11362  16000  had       36971 42290 wash
15420  17503  your      43120 47480 water
17503  23360  dark      49021 52184 all
23360  28360  suit      52184 58840 year
```

- *Phonetic label (.phn):* *(Note: beginning and ending silence regions are marked with h#)*

```
0 7470      h#   21053 22200 r     32550 33253 r     46040 47480 axr
7470 9840   sh   22200 22740 kcl   33253 34660 iy    47480 49021 q
9840 11362  iy   22740 23360 k     34660 35890 z     49021 51348 ao
11362 12908 hv   23360 25315 s     35890 36971 iy    51348 52184 l
12908 14760 ae   25315 27643 ux    36971 38391 w     52184 54147 y
14760 15420 dcl  27643 28360 tcl   38391 40690 ao    54147 56654 ih
15420 16000 jh   28360 29272 q     40690 42290 sh    56654 58840 axr
16000 17503 axr  29272 29932 ih    42290 43120 epi   58840 61680 h#
17503 18540 dcl  29932 30960 n     43120 43906 w
18540 18950 d    30960 31870 gcl   43906 45480 ao
18950 21053 aa   31870 32550 g     45480 46040 dx
```

Another interesting set of statistics is shown in

Table 3-5. It lists *Mean duration*, *Standard deviation*, *Max duration* and *Min duration* for all the phonemes in the *TIMIT* database [28].

Phoneme	Mean Duration *(ms)*	STD *(ms)*	Max Dur. *(ms)*	Min Dur. *(ms)*
i	89.9	96.4	428.2	23.3
I	78.7	83.6	399.1	23.3
e (eI)	128.3	135.13	390.6	31.7
ɛ	90.6	96.2	281.1	21.4
ae	148.6	155.2	425	35
a	123	129.7	483.4	20
ɔ	123.6	131.2	396.6	24.4
o (oU)	126.7	133	438.6	34.9
U	76.5	82.1	253.7	18.3
u	107.8	117.8	334.2	24.4
ʌ	89.3	95.9	340.1	18.8
3	118.3	124.6	366.4	30
∠	48.5	51.6	173.6	46.3
al	145	154.7	424.4	38.5
ɔI	162.9	167.7	318.5	62.6
*a*U	163.6	171.6	415	62.7
ju	107.8	117.8	334.2	24.4
j	66.7	75.7	231.3	11.4
w	67.4	74.6	216.4	9.5
l	60.5	64.6	229.8	11.9
r	61	65.9	180.2	2.3
m	61.2	66	229.75	4.3
n	54.3	58.8	212.3	7.8
ŋ	62.6	67	240.25	13.5
f	103	108.6	305	13.3
v	60	63.9	190	12
ʊ	90.3	97.7	310	12.6
σ	37.9	42.6	128.7	4.5
s	113.2	119.4	381.25	23.4
z	84.1	89.7	276.4	18.2
ʃ	116.1	119.8	299.8	24.1
ź	83	88.3	263.6	22.2
h	65.6	72.2	250	13.8
p	44.2	49.3	150	6.3
b	17.5	18.9	98.6	2.9
t	48	53.7	152.5	4.6
d	23	26.1	120	2.4
k	51.4	57.3	193.8	6.1
g	30.4	33.1	106.1	7.5
č	86.3	90.6	345.6	20
ž	58.7	64.1	231	14.9

Table 3-5: Phoneme duration distribution in *TIMIT* database

Table 3-5, vowels and nasals illustrate higher duration variability than most fricatives.

To pre-process the *TIMIT* database for our experiment, we developed a set of tools to extract words and phonemes from the sentence files and store them in separate files to be used for feature extraction. In particular, we developed two tools called *w_xtract* and *p_xtrac*[4]*t*. The preprocessing *w_xtract* tool extracts the desired word and saves it into a separate <*.wav*> file for each speaker and from the specified sentence. It uses the <*.wrd*> file to determine the *begin* and the *end* sample counts for each word. It also copies the 1024 byte *NIST* Sphere Wave header from the original sentence file (*SA1* here) to the beginning of each word's file. To extract phonemes and populate the required database for the accent identification experiment, we developed the "*p_xtract*" tool. It is similar to "*w_xtract*" and extracts the desired phoneme from a sentence saving it into a separate Wave file. The desired phoneme is identified by its relative index in the sentence file (*SA1.wav*). The utility reads begin and end sample counts of each phoneme from <*.phn*> and similar to "*w_xtract*", copies the NIST Wave header to the beginning of each phoneme file.

3.2 Signal processing and *HMM* toolset

We used a set of signal processing and *HMM* building tools to prepare samples and calculate the feature vector and build the *HMM* model for the words in the *SA1* sentence of the *TIMIT* database. We used these tools to study the effect of accent in ASR performance and to

[4] See Appendix A for source listing of w_xtract and p_xtract

evaluate the performance of our proposed solution. A more elaborate discussion of this experiment is included in Chapter 4. These tools included low pass filtering, down-sampling, *LPC* and *Cepstral* calculations and *HMM* training and testing. We obtained these tools from the authors of [29] which was written based on an isolated digit recognizer experiment. We made modification to these tools to make them appropriate for our purpose.

3.3 MATLAB and MATLAB Neural Networks tool-box

We used *MATLAB* neural network tool box [30] and its general programming scripting facility for the accent identification experiment. In Chapter 5, we explain this experiment in more details. *MATLAB* provides an extensive and flexible set of tools for building and analyzing neural networks. Figure 3-1 Steps to create, train and test a neural networkshows three steps and corresponding commands to create, train and test a neural network in *MATLAB* neural network toolbox.

STEP 1:. CREAT THE NEURAL NETWORK OBJECT AND INTIALIZE IT (Use command *newff* or use *nntool*)
STEP 2:. TRAIN THE NETWORK (Use command *train* or use *nntool*)
STEP 3:. TEST THE NETWORK WITH TESTING DATA AND COMPARE THE RESULTS (Use command *sim* or use *nntool*)

Figure 3-1 Steps to create, train and test a neural network in *MATLAB*

3.4 PRAAT

PRAAT [31] is a speech analysis tool that was built by Paul Boersma and David Weenink from the Institute of Phonetic Science in the Netherlands. *PRAAT* is easy to use and very flexible. We used *PRAAT* for a variety of purposes such as editing, displaying and analyzing speech files and calculating pitch and formant frequencies. *PRAAT* provides a set of useful tools for speech analysis, speech labelling and segmentation, neural networks, speech synthesis, speech manipulation and more. It also provides a macro programming language for automating experiments.

CHAPTER 4 Effects of accent on speech recognition systems

The presence of an accent in speech can significantly degrade the recognition accuracy of automatic speech recognition systems. Many *ASR* systems are trained to handle speaker variability within speakers of the native language of the system with a standard (neutral) accent.

Native language plays a crucial role in the ability of speakers to conform to accent neutral ways of pronouncing sounds. Acoustic models in ASRs get confused by the effect of accent on acoustic features and language models get confused by poor grammar and uncommon word selection [32]. Even speakers with the same mother tongue, but with heavy regional accents, lower the recognition accuracy of an *ASR* that is trained for either the standard (neutral) accent or a cocktail of accents [33]. The issue becomes more challenging for large vocabulary continuous speech recognition systems (*LVCSR*). A conversational scenario in [32] demonstrates how accent can affect the performance of an *LVCSR* system. In this scenario, a native speaker is compared to a Japanese speaker speaking in English, both trying to get information from a speech enabled customer help line of an airline as its is shown in

Figure 4-1: Comparing an *ASR*'s performance in dealing with (1) native and (2) non-native speakers (with accent).

At the time of this scenario, the Japanese speaker had been living in the U.S. for two years. The native English speaker, with only one misrecognition experience in flight number, is able to get the information that she needs from the system.

In the case of the speaker with the Japanese native language, the user attempts several times to convey what he needs to the system and in the end gets frustrated and leaves the system unsuccessfully. This scenario shows that in real world applications, speaker variability requires serious consideration if *ASR* systems are to be used ubiquitously.

The scenario illustrated in Figure 4-1 shows other issues than accent that are not the focus of this research work. For example, native and non-speakers say flight numbers in different ways.

System: Welcome to Pittsburgh Airlines' automated flight information service. Do you know the flight number?
User: Five-twelve.
System: Would *you* like arrival or departure information?
User: Arrival.
System: Flight 524 arriving on Tuesday, Feb 27th. Is that correct?
User: No.
System: What would yon like to change, the flight number or the date?
User: The flight, number.
System: Please say the flight number.
User: Five one two.
System: Flight 512 arriving on Wednesday, Feb 28th. Is that correct?
User: Yes.
System: Flight. 512 arrives in more than one city. Would you like Boston Logan or Charleston, SC?
...

1

System: Welcome to Pittsburgh Airlines' automated flight information service. Do you know the flight number?
User: Yes.
System: Please say the flight number.
User: five hundred and twent- five hundred and twelve.
System: Would you like arrival or departure information?
User: Arrival.
System: Flight 283 arriving on Tuesday, Feb 27th. Is that correct?
User: ? No flight, five hundred and twelve please
System: What would you like to change, the flight number or the date?
User: ? So... flight number is five hundred and twenty no five hundred and twelve
System: Please say the flight number.
User: Flight, number is five hundred and twelve.
System: Flight 94 arriving on Wednesday, Feb 28th. Is that correct?
User: No.
System: What would you like to change, 1,he flight, number or the date?
User: Flight, number, please.
System: Please say the flight number.
User: Five hundred and twenty.
System: Flight 520 arriving on Wednesday, Feb 28th. Is that correct?
User: Yes. System: Flight 520 arrives in more than one city. Would you like Chicago O'Ha.re or Kansas City?
User: no, no. Not, twenty, twelve.
System: Please repeat. or rephrase that.
User: ... ?
System: Please repeat or rephrase that.
User: *<click>*

2

Figure 4-1: Comparing an *ASR*'s performance in dealing with (1) native and (2) non-native speakers (with accent)

This chapter explains the experiments that we conducted for evaluating and analyzing the core hypothesis of this thesis. This hypothesis is the scenario that was demonstrated in the airline customer support line. We validate that an *ASR* engine, when dealing with a set of speech samples from speakers with different distinct accents, will give higher recognition accuracy when trained and evaluated by a single accent compared to a cocktail set of several accents. The comparison is exhibited in Figure 4-2: ASR with (a) Cocktail training set versus (b) single accent training set.

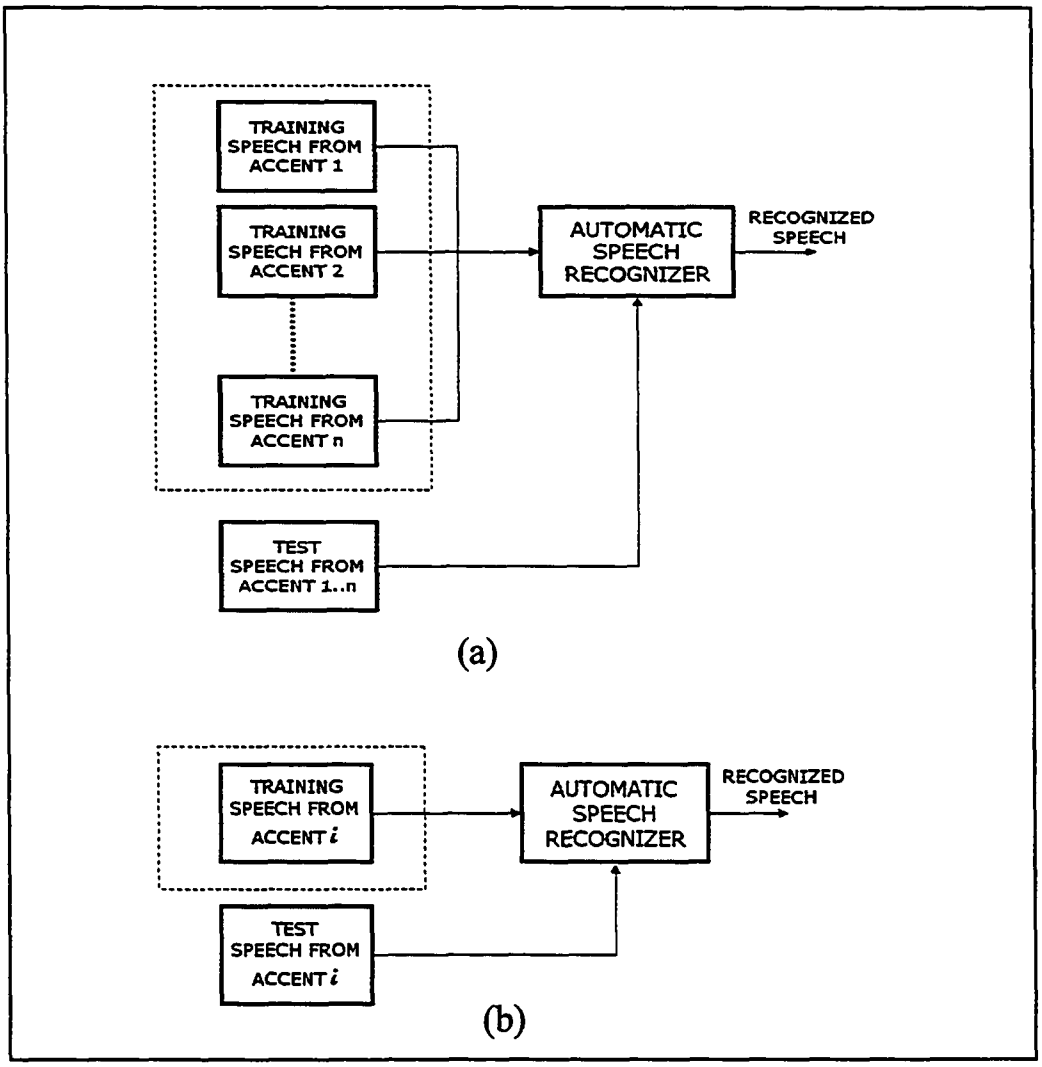

Figure 4-2: ASR with (a) Cocktail training set versus (b) single accent training set

Once we show that this hypothesis holds, we propose a two stage speaker accent adaptation approach that enhances the accuracy of the *ASR* engine when recognizing speech from speakers with an accent. The proposed solution first identifies the speech sample's accent and then routes the feature vector extracted from the utterance to an appropriate speech recognition engine from a bank engines that are each trained for a particular accent.

4.1 State-of-The-Art in Speaker Accent Adaptation

There are situations that an ASR system can not adequately perform for speakers that exhibit a non-standard accent. By adapting the ASR to these speakers, we can achieve a reasonable recognition performance. These approaches generally examine speech from the speaker; identify the difference between the speaker and the standard – average – way of speaking or the accent of the speaker, and through an adaptation scheme attempt to adapt to the speaker.

The training data that is used to identify the speaker's accent is known as accent adaptation data. If for a speaker or a group of speakers a large amount of adaptation data is available, the *ASR* engine can be adapted off-line before use. This adaptation approach is known as *batch adaptation* [34]. If the number of groups of target users of the *ASR* engine is small, *batch adaptation* can be an effective and fast method of speaker accent adaptation. Once the engine can generalize the different speaker accents for which it is trained, it can recognize an accent and then choose the right codebook, model or feature transformation function to enhance recognition accuracy. The approach that we propose is a *batch adaptation method.*

If there is not enough *a priori* adaptation data available, then the *ASR* model has to adapt over time to the speaker, and therefore its accuracy performance may increase gradually. For each new speaker introduced to the model, the same process has to take place. This adaptation approach is known as *incremental or on-line adaptation* and is harder to implement in real-time speech recognition.

Another way that speaker accent adaptation methods are categorized is by distinguishing between *feature-based* and *model-based* approaches. In the *feature-bases* approach, the *ASR* model attempts to compensate for speaker variance from a standard accent. This approach is also known as the speaker normalization approach since it tries to force aligns a speaker's speech vectors to the standard accent. If *Y* is the speech with the accent and X is the standard accent speech, *Fv* in (4-1) is the transformation function of the adaptation, and *v* is the parameters of this transformation function. The parameters are chosen in a way to maximize the likelihood of the adaptation data.

$$X = F_v(Y) \tag{4-1}$$

The simplest form of this kind of speaker normalization is called Cepstral Mean Normalization (*CMN*) [35]. It models a cepstral bias vector for each class of accents as the difference between the speaker with the accent and a standard accent speaker. The bias is then used to normalize the feature vector before recognition.

Another feature-based adaptation approach is vocal tract length normalization (*VTLN*) [36]. *VTLN* normalizes the input speech signal by a linear frequency scaling factor or frequency warping. This frequency warping compensates for variations in formant frequencies across

different accents. In general, *feature-based* approaches have been found to have smaller improvements on the recognition accuracy than *model-based* approaches [37].

In *model-based* approaches instead of the feature vectors, the *ASR* model parameters are adapted to the speaker's accent. The adaptation is a mapping function that maps the original (standard) accent model of λ_x to the transformed model of λ_y which is supposed to better match the observed utterance as it is given is (4-2).

$$\lambda_x = G_\eta(\lambda_x) \tag{4-2}$$

η is the parameter set that needs to be estimated such that it maximizes the likelihood of the adaptation data. Maximum likelihood linear regression (*MLLR*) is a commonly known method of *model-based* adaptation [G8]. *MMLR* is a transformation approach that estimates a set of linear transformation matrices to modify the parameters of the *HMM* based *ASR* model.

We propose a third approach that based on prior knowledge of accent classes, for the purpose of speaker adaptation, a model for each accent class is trained. This requires the target user groups of the *ASR* system to be known *a priori*. The adaptation function in this case is a selector function as in (4-3) that based on the identified accent a_i of the input utterance, chooses the appropriate model e_i that is trained for a_i. The appropriate model e_i is trained in such a way to minimize recognition error for accent a_i.

$$e_i = \delta = \{ \textit{the engine the is trained with accent } a_i \} \tag{4-3}$$

4.2 The experiment and the results

The proposed approach for speaker accent adaptation is based on implementing a selection function as the adaptation mechanism. The assumption is that based on the input utterance, the speaker is assigned to an accent class of a_i. The selection function of δ routes the utterance to the appropriate engine of e_i.

Figure 4-3 shows the block diagram of the accent adaptation mechanism.

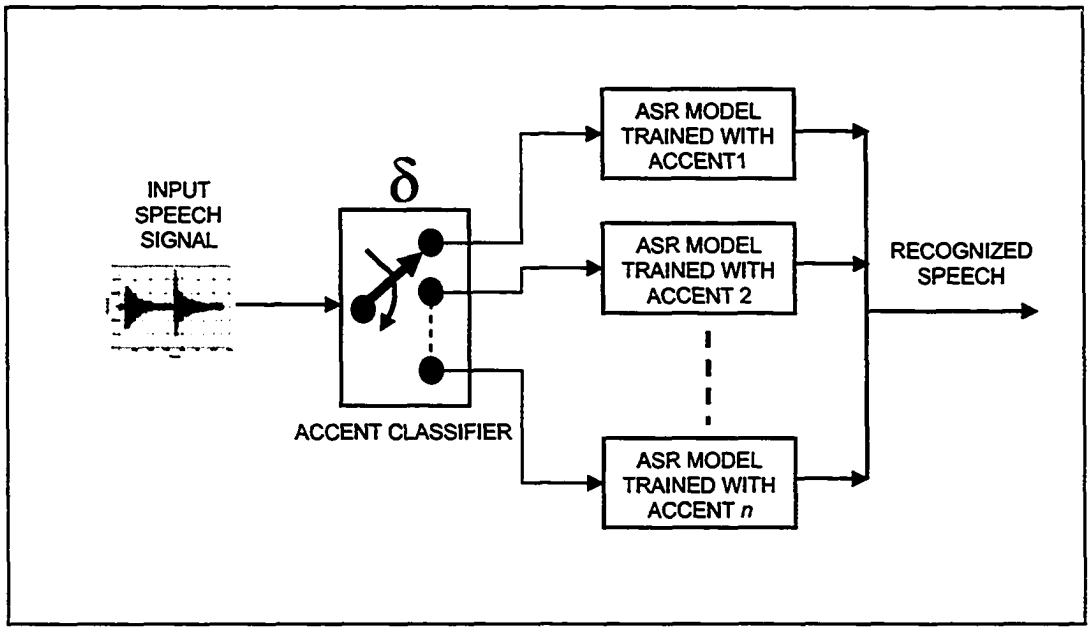

Figure 4-3: The selection function of δ as the adaptation function in the proposed approach

We show that an ASR model similar to the proposed approach significantly lowers the recognition error when compared to a single model that is trained with a training data comprising of all classes of accents. The ASR engine is built based on the hidden Markov model.

For our experiment, we used all 630 instances of the *SA1*sentence for dialect regions 1-7 from *TIMIT* database as training and testing tokens. We followed the same train and test grouping structure as in *TIMIT*. This resulted into 462 training utterances and 168 test utterances. The *SA1* sentence type is designed to stress dialectal differentiation aspects of the words and is read by all the speakers. We used a parser to extract each word out of the *SA1* sentence files, and store them separately in different *WAV* files. We disassembled the *SA1* type sentences for all speakers into separate words to make training and recognizing stages simpler to implement. Figure 4-4: *SA1* sentence for speaker FCFJ0 (female) in training set from *TIMIT* shows *SA1* sentence time domain and frequency domain spectrums.

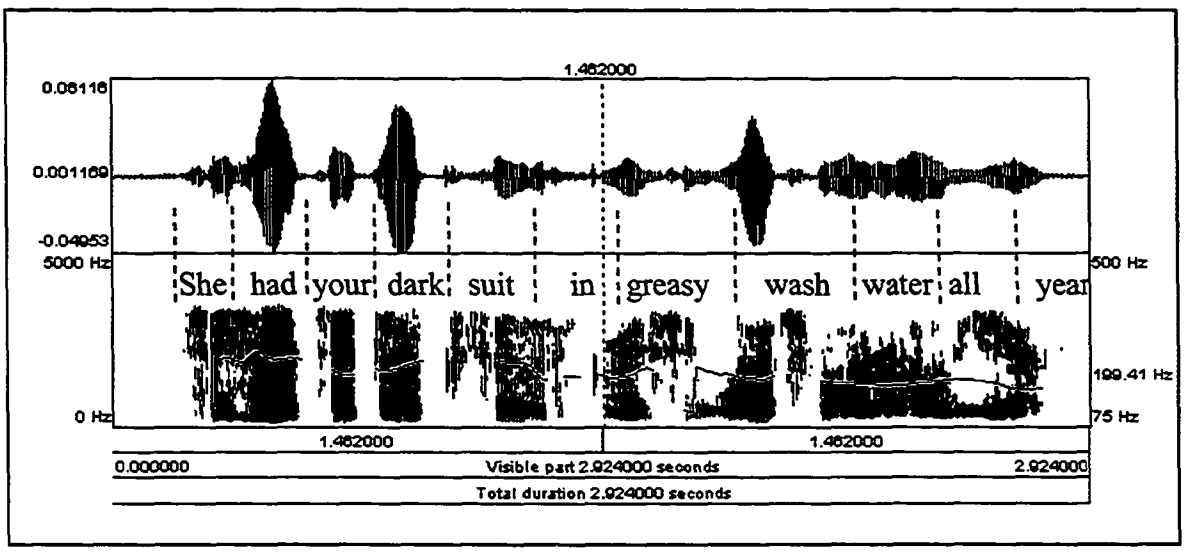

Figure 4-4: *SA1* sentence for speaker FCFJ0 (female) in training set from *TIMIT*

See Chapter 3 for more details on *TIMIT* database, its sentence types and distribution of gender and dialect regions.

4.2.1 Pre-processing of the Speech Data

Before we could use the *TIMIT* database, we performed several pre-processing stages to transform digital speech recorded in Wave format into feature vectors that we needed to train and test the *HMM* models. Figure 4-5 shows the pre-processing stages that we used.

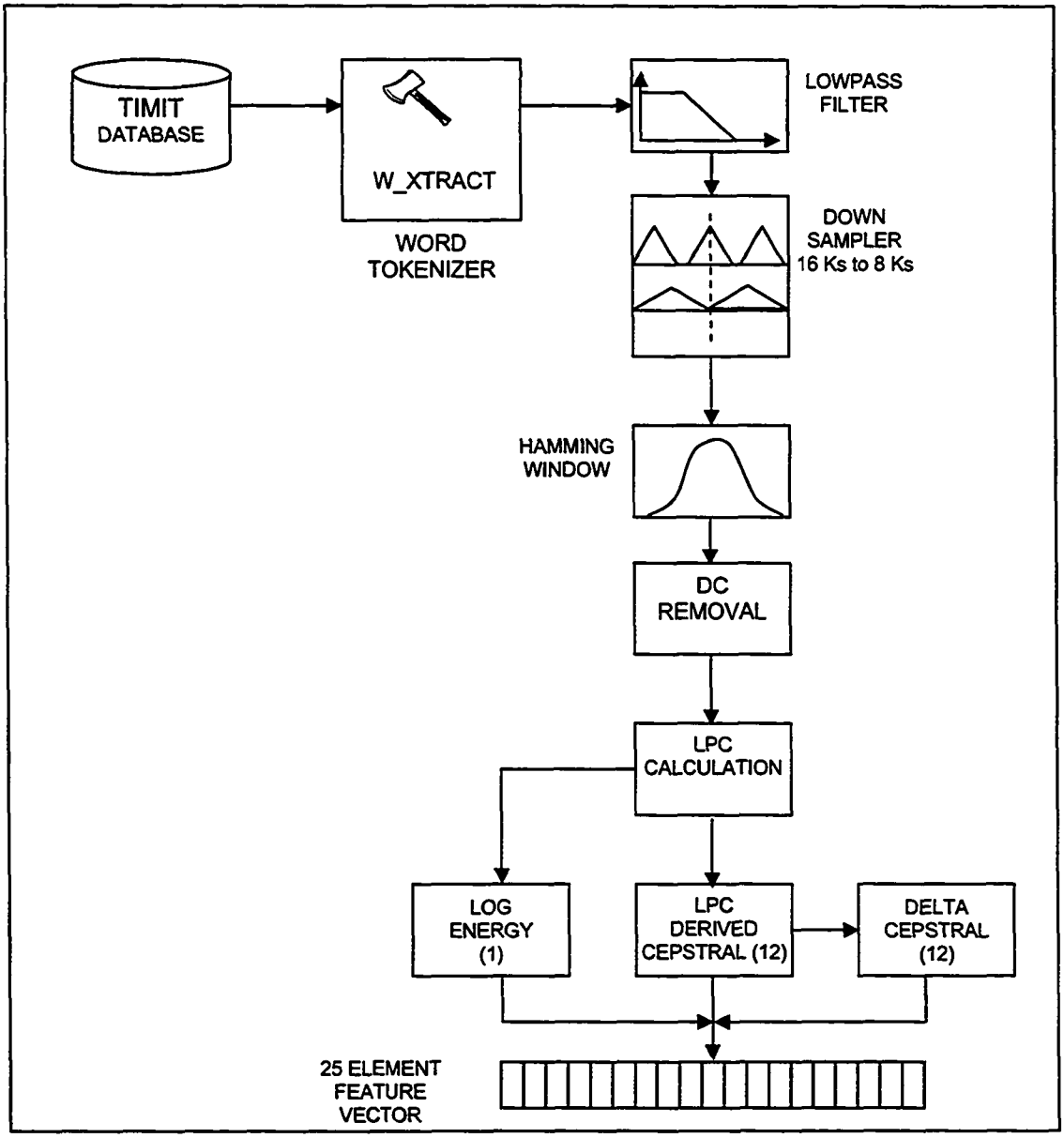

Figure 4-5: Pre-processing steps to prepare feature vectors from *TIMIT* database

The Low Pass Filter, the Down-sampler and the Hamming window

The low pass filter was used to filter out frequency elements above 3.4 *KHz*. This was necessary to avoid aliasing and satisfies the *Nyquist* criterion once the down sampling is done reducing from 16,000 *samples/sec* to 8,000 *samples/sec*. The filter was designed as a *Chebyshev IIR* filter. The ripple parameter was set to -1, the filter was a 5^{th} order filter, and sampling rate was set to 16000 samples-per-second (this is the sampling rate of *TIMIT* Wave file). Figure 4-6 shows the filter's frequency response.

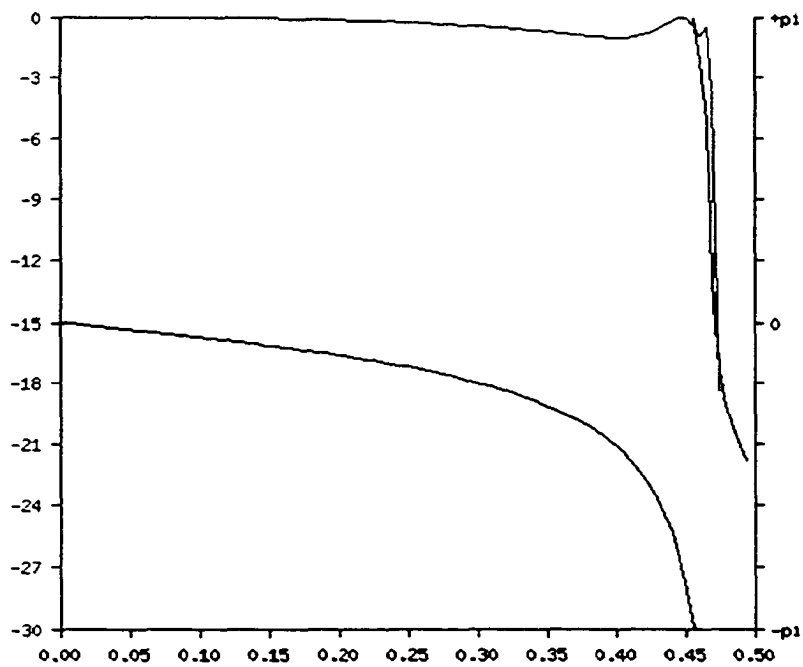

Figure 4-6: The frequency response of anti-aliasing Low-Pass Filter

The input to the down-sampler is the standard Wave format from *TIMIT* filtered by the *LPF*. It first removes the header, which is the first 1200 bytes of the file, and then simply skips

every other sample in the filtered file. A Hamming window with the window size of 25*ms* and a window shift of 12.5 *ms* was applied to the signal.

The feature extractor

This tool took the down-sampled low-pass filtered speech data, removed the *DC* component, and calculated 8 coefficients *LPC*; from the *LPC* coefficients it then calculated Cepstral coefficients, Δ–Cepstral coefficients and the log energy as the feature vectors that were used for training and testing the *HMMs*. The length of the cepstral vector and delta cepstral vector were each 12 and with the energy the length of the feature vector was 25.

This tool for each frame first calculates the LPC and then from the LPC, it calculates cepstral vector and energy.

4.2.2 The recognition module and the results

The recognition module was an implementation of *HMM* along with training and testing algorithms. A 5-state *HMM* [26] was trained for each word for each accent, resulting in 80 different models. The box representing an accent classifier in figure 2, was replaced by manually training all models, and testing them with a test portion of the corpuses of the *TIMIT* database, and recording the results. Figure 4-7 illustrates the schematics of the *HMM* used for the recognizer block.

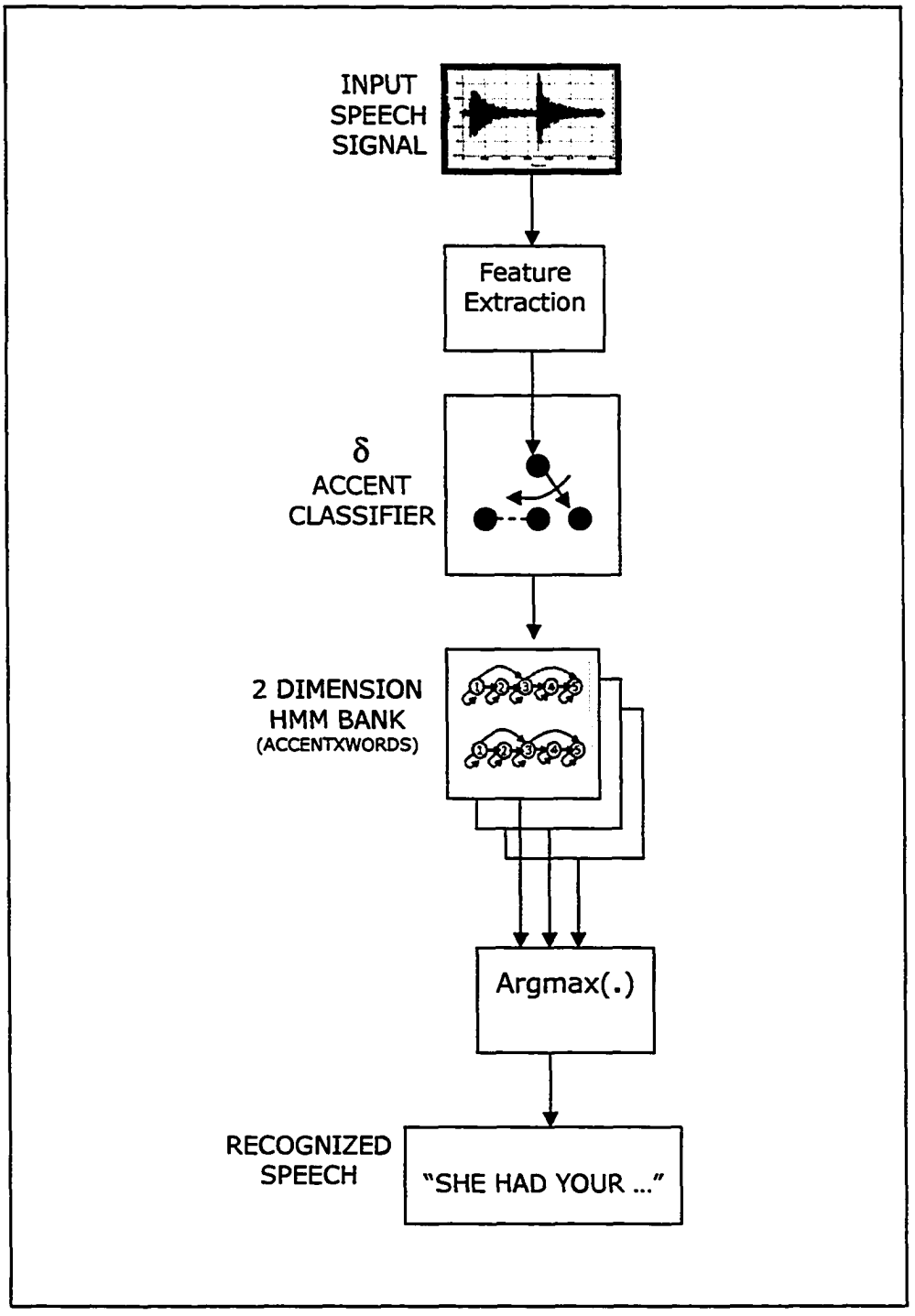

Figure 4-7: The block diagram of the system including the HMM bank recognizer

Table 4-1 shows the result of the simulation. As it is shown in the table, the second column of numbers are error rates for the case of testing an engine that is trained with all 7 *TIMIT* dialect regions (accents), which is subsequently tested by each region. The average error rate when calculated across the regions is 9.45% for the case that the engine is trained by all accents.

Dialect Region and number of talkers in each region	HMM training based on all Accents	HMM training based on one Accent	Improvement %
1 (49)	15.15%	12.12%	20.00%
2 (102)	5.56%	3.84%	30.94%
3 (102)	6.41%	5.55%	13.42%
4 (100)	11.8%	9.02%	23.56%
5 (98)	5.95%	5.55%	6.72%
6 (46)	12.12%	6.06%	50.00%
7 (100)	9.17%	7.79%	15.05%
Weighted Average Error rate	**8.71%**	**6.79%**	**22.02%**

Table 4-1: The error rate table: Comparison between training HMM with all accents and only the accent being tested

The third column shows the engine that is exclusively trained for the region being tested represented in each row. In the right-most column, the percentage of error rate improvement is calculated according to (4-4).

$$\%\ IMPROVEMENT = \left(\frac{err_1 - err_2}{err_1}\right) * 100 \qquad (4\text{-}4)$$

err_1 is the recognition error rate for each accent when the engine is trained with all accents and err_2 is the recognition error rate when the engine is trained with one accent.

The results are consistent across all seven regions. As shown in the Table 4-1, the engines that are trained only with one accent perform better than the case when the engine is trained with all accents. It is also important to note that the result for each dialect region is the aggregate of the results of all the words for all the regions. Another observation that we made was, in several cases where *HMMs* were trained for a particular region it would perform better for other accents than the accent it was trained for. For example, the engine that was trained for the accent from region 1 produced an error rate of 15.5%. When the same trained *HMM* was tested against speech tokens from region 2, the error rate was 5.56%. One reason for this could be that some accents by nature are more difficult to recognize than others. We could see this clearly since when we tested accent 1 against *HMMs* trained for all regions individually or all together, it produced a relatively higher error rate than accents from other regions. At the same time, our results were consistent with the fact that when we trained the *HMM* with a cocktail of accents, the performance always degraded in contrast to the case where the *HMM* was trained with the accent that was being tested against. Figure 4-8: The graphical comparison for error rates between training HMM with all accents and only the accent being tested shows a chart that summarizes the error rates in Table 4-1.

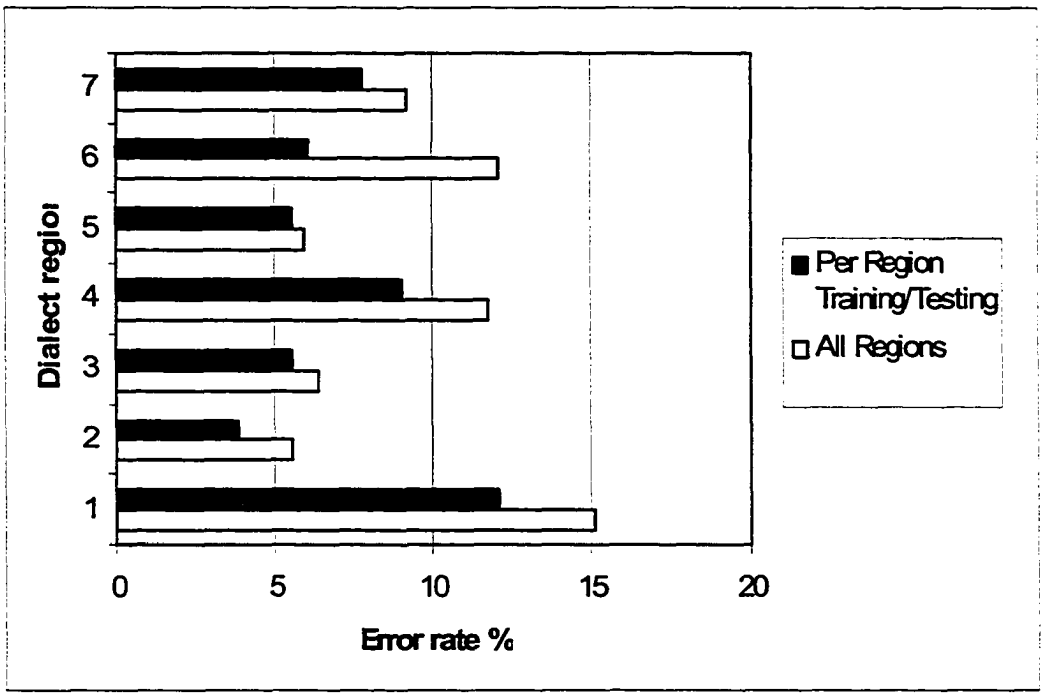

Figure 4-8: The graphical comparison for error rates between training HMM with all accents and only the accent being tested

4.3 Summary

As the robustness of speech recognition methods is increasing, speech enabled systems and environments are starting to gain popularity in day to day applications. From auto-attendants in call centers to automobiles, speech interface is proving to be more economical and effective. Speech enabled products are being used in different geographical regions around the world. Speaker accent is posing new challenges in designing and building automatic speech recognition systems.

We have presented a way to lower the recognition error rate in Automatic Speech Recognition systems by adapting the recognition stage to the accent of the utterance.

Many *ASR* engines that are available today are trained for the native accent only. They deliver a higher error rate when the speaker has a non-native accent. To address accent variations of speakers, some ASRs use a training database from an aggregate of a collection of accents. This approach usually lowers the performance of the system for native accent speakers and at the same time the overall recognition performance across all accents may not be satisfactory.

In order to improve the recognition performance and lower the error rate, we propose a two-stage process:

1) The first stage is to recognize the accent of the utterance. To conduct the simulation, we replaced the accent recognition stage with manual training of a bank of *HMMs* with accents from dialect regions 1 to 7 from the *TIMIT* database. Chapter 5 explains our experiment of using a neural network to identify and classify accents

2) In the second stage, from the bank of engines that are each trained with different accents – including a native accent – we choose the appropriate one to recognize the utterance.

We show that by using a bank of recognizers, on average, we lower the error rate by 24%. Further work can be carried out on localizing this method on foreign accents such as Chinese, German, French, Arabic and Persian. This requires building accent centered corpus databases similar to *TIMIT*.

CHAPTER 5 Accent identification

Accent identification is a new challenging problem closely related to the field of multilingiality [38] in automatic speech recognition. Accent identification is an important consideration since the knowledge gained from accent classification and identification could improve overall recognition performance [39].

In the previous chapter we presented the results of our experiment which show that by identifying an utterance's accent and adapting the speech recognition model to it, we can significantly lower the recognition error rate. In the experiment, we replaced the accent identification module by manually switching among all the different dialect regions from the *TIMIT* database. In this chapter we implement an accent identification module based on back-propagation artificial neural networks and demonstrate the results.

Although in this thesis we use the words accent and dialect interchangeably, there are subtle differences that differentiate an accent from a dialect. *Dialect* is defined as "a regional or social variety of a language distinguished by pronunciation, grammar, or vocabulary [40]". *Accent* on the other hand, is defined as "the relative prominence of a particular syllable of a word by greater intensity or by variation or modulation of pitch or tone [41]". Simply put, dialect is a variation among native speakers of the same language where accent is variation of speakers with different native languages when compared to the standard accent in a language. The accent exhibited in second language pronunciation would mostly depend on several speaker related factors such as the speaker's first (native) language, the age at which the speaker started to learn the second language and the amount of interaction that speaker has with native speakers of his/her second language [32].

In this chapter we propose a method of accent identification by tracking vocal tract model trajectory, based on studying the temporal evolution of *LPC* coefficients at the phonetic level as shown in a simplified two-dimensional illustration in
Figure 5-1. The length of each vector in the figure reflects the time duration of the phoneme. We show that this method can be a viable basis for accent identification and we study the appropriateness of some of the vowel phonemes in identifying American English dialects in the *TIMIT* database.

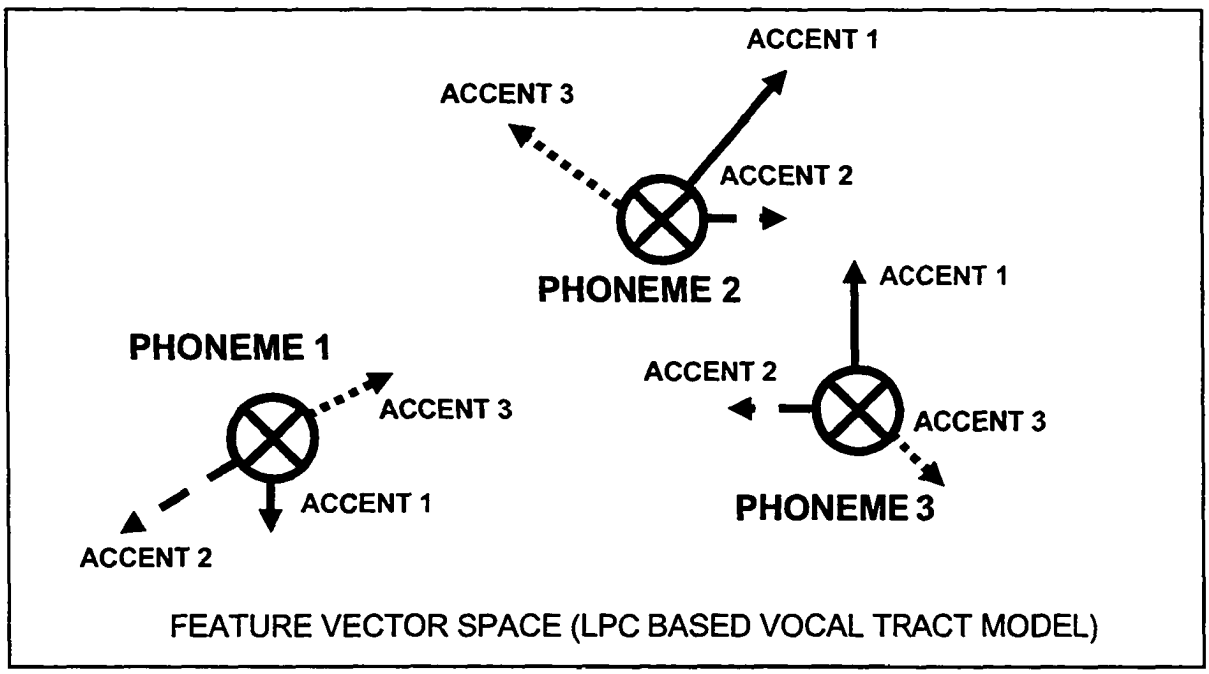

Figure 5-1: Illustration of accent modeling based on temporal evolution of feature vector space at phonetic level

5.1 State-of-the-Art in Accent Identification

Automatic accent identification has become a serious consideration and also a challenge in modern automatic speech recognition, processing and analysis systems [42].

Identifying accent can not only assist in lowering the recognition error rate, it can be a useful tool in many other areas such as forensic speech analysis. Speech patterns are proving to be increasingly valuable in criminal investigation. Accent and dialect analysis can provide information about an individual's homeland. Since September 11, 2001, the forensic role of speech analysis and accent identification has taken a more important role. Voice prints are being used to analyze the authenticity of taped speech and accent and dialect identification are being used as a guide in speaker profiling [42].

A number of research works have proposed and studied different ways of identifying accent from a speech utterance sample. Relying on prosodic features, sub-word acoustic analysis and word-based features are among the different approaches in accent identification.

Several studies propose an approach based on prosody as a discriminative feature in foreign accent identification [43] [44] [45]. Prosodic features such as normalized fundamental frequency (F_0) range and syllable rate are considered as discriminative feature space for accent identification [45]. In [41], based on the work done on *NATO N-4* corpus, it was shown that normalized F_0 exhibits a potential discriminative value for classifying *English Canadian*, French Canadian and *UK* accents. This study showed that normalized F_0 frequency, with its average level set to zero on the horizontal axis, varies between -0.39 to 0.36 for *English Canadian*, -0.36 to 0.44 for *French Canadian* and -0.38 to 0.47 for *UK*

accents. Similar experiment was done on syllable rate and sentence duration and showed that these prosodic features illustrate variability among the three accents in the research. The result showed that the average syllable rate was 7.5 for *Canadian English*, 6.5 for *French English* and 5.8 for *UK* accents. It also showed that the sentence duration varies distinctively among these three accents.

Spectral acoustic space and sub-word modeling is another approach to accent identification. A recent study [41] proposes a multi-lingual acoustic model for identifying accent. This model relies on detecting and exploiting phonemes that a non-native speaker borrows from his or her native language when speaking in a foreign language. The result of one study for accent classification based on sub-word phonetic features that was done on the *TIMIT* corpus database [41], showed about 30% identification accuracy. When a more educated selection of phonemes was used, the identification accuracy increased to 42%. A second study [46] showed that using averaged formant frequencies and phoneme duration as the feature vector can produce accent identification accuracy of about 50% for different regional Dutch accents.

The third approach to identifying accent is word based modelling and classification. This approach relies on dialectal variations in uttering words. Although the number of words that span a language is quite large, normally in the range of tens of thousands, in practice only a subset of these words are used frequently in day-to-day conversations. In fact it is shown that out of about 26,000 words that constitute *American English* [47], a mere one hundred words account for 66% of words used in daily American English [47]. This approach uses the variation space for each of these most frequently used words to classify accents.

5.2 The experiment and the results

The purpose of our experiment was to implement the accent classification function of δ that we introduced in the previous chapter.

We propose a sub-word acoustic feature based approach to accent classification [48]. This criterion relies on the assumption that speakers from different dialectal regions and with foreign accents exhibit different vocal tract variations when they utter an utterance at the sub-word level. In this chapter, we validate this assumption and we show the results of classifying *TIMIT* dialects using a back propagation learning neural network classifier. Figure 5-2 shows the block diagram of this accent classifier.

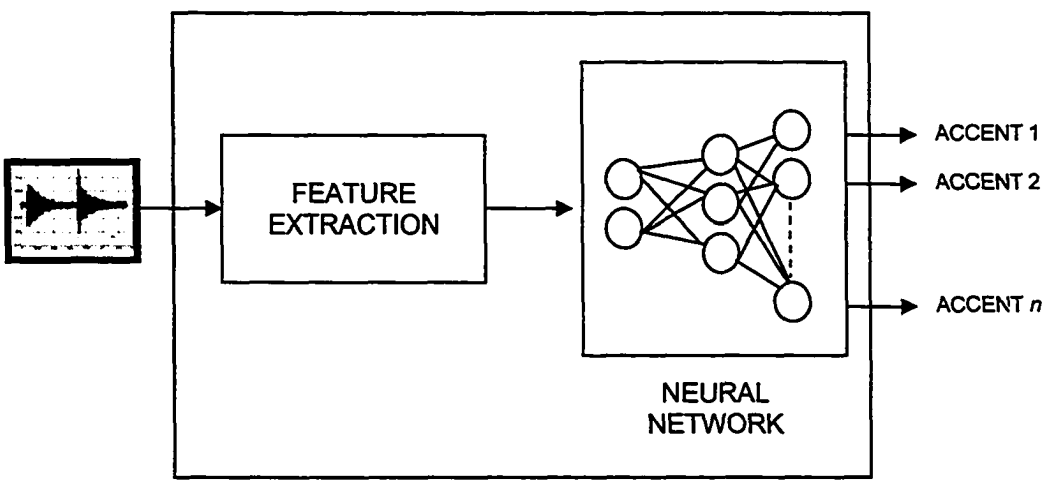

Figure 5-2: Block diagram of the proposed accent identification module

In this experiment, we used vowels to extract the feature vectors for training and testing our classifier. To capture the temporal evolution of the vocal tract, we calculated three sets of 8 *LPC* reflection coefficients at the beginning, in the middle and at the end of each phoneme as

the estimation parameters. In order to allow the neural network to learn this temporal evolution, we concatenated these three vectors as part of the input to the network.

LPC was originally calculated for frames of 30 ms and depending on the duration of the phoneme, frames, may or may have overlapped as is shown in Figure 5-3. The feature vector was then augmented with the codes indicating the phoneme giving a total length of the 25. Since the *LPC* reflection coefficients are calculated in the range of -1 to 1, to assist the neural network to converge faster, we normalized the phoneme code between 0 and 1. We chose the frame length of 30 ms with the assumption that this length will give a good starting point. In later stages of our investigation, we varied the frame length and the degree they were dispersed from each other to study how these changes would affect the accent recognition performance. We will present the results of these experiments in following parts of this chapter.

We will see that the frame length of 30 ms is not the optimal length for this experimental setting and other frame lengths will produce better results. We will also see that expanding and contracting the distance between adjacent frames has significant impact on the recognition performance. This is a strong indication that the degree to which we capture the temporal variation of the feature vector space is important in producing lower accent recognition rate.

To train and test the classifier in the implementation, we used the 168 instances of the *SA1* sentence from the *TIMIT* corpora. From [41], we know that the vowels */aa/*, */ix/* and */er/* exhibit the maximum dialect discriminative properties.

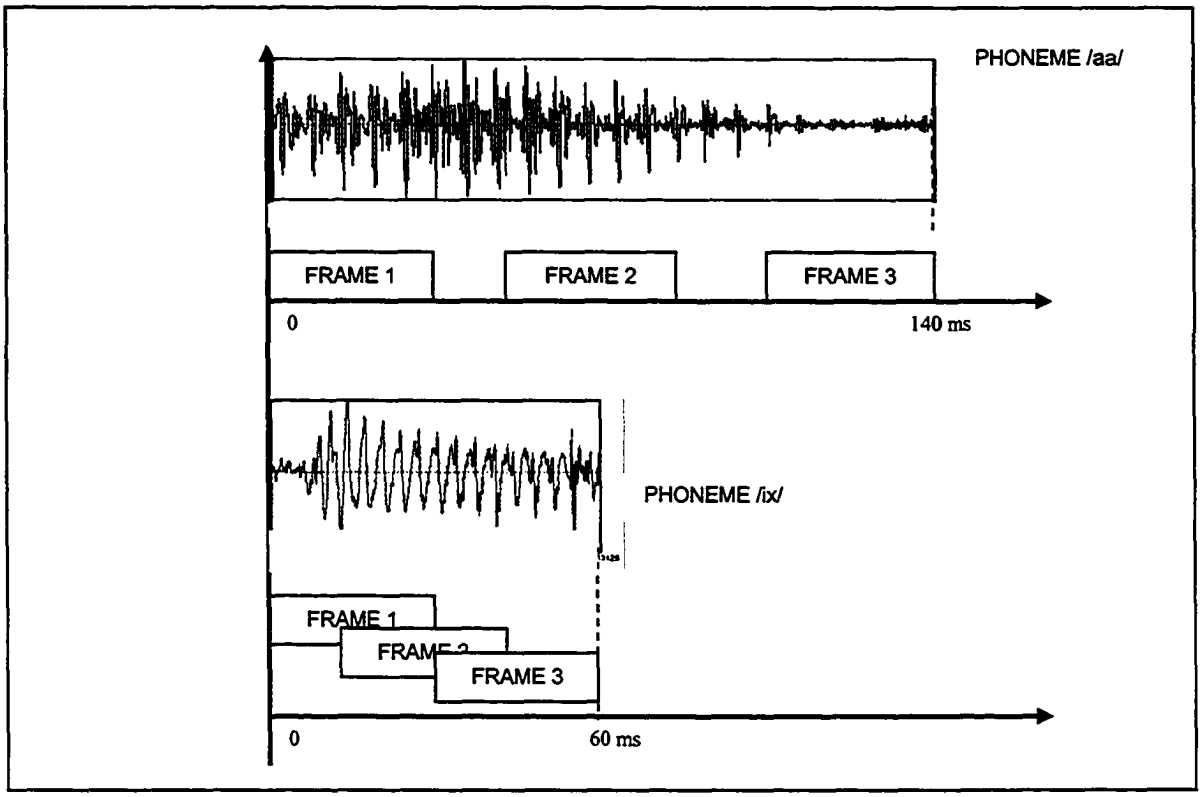

Figure 5-3: Framing of phonemes. Depending on the duration of the phonemes, frames mayor may not overlap

We developed a phoneme extraction tool called *p_xtract* to parse the phoneme transcription files in *TIMIT*, and extracted the phonemes that we desired. In total we prepared 572 instances of */aa/*, 133 instances of */ix/* and 93 instances of */er/* from both male and female speakers. Depending of the availability of female and male speaker utterances, we allocated between 10-20% of the available tokens for each phoneme to testing and the rest to training.

We used the neural network toolbox in *MATLAB* to implement the classifier. This tool box is a powerful array of tools to implement neural networks with a variety of properties. To build a benchmark, we started with a network with one hidden layer. We used 25 nodes in the

input layer, 220 nodes with tangent sigmoid transfer function which ranges from -1 to 1, in the hidden layer and 8 nodes in the output layer with log-sigmoid transfer function which ranges from 0 to 1 as it is shown in Figure 5-4. Once the training was completed and in testing stage, the output that generated the highest value was picked as the recognized accent in a 0-1 decision function.

Figure 5-5 shows the convergence of the network when it was training for the */aa/* single scenario.

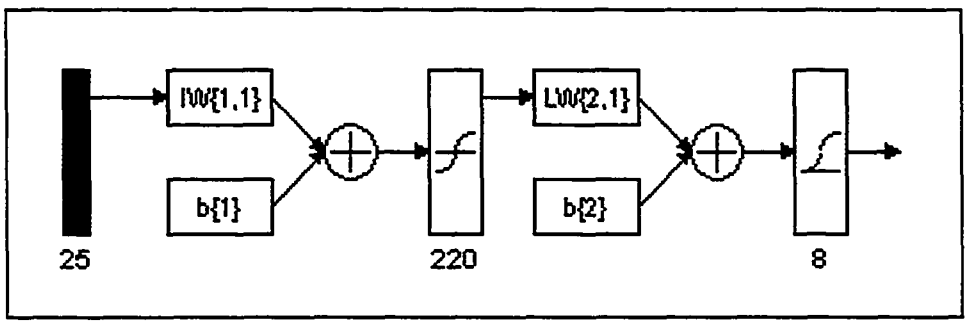

Figure 5-4: The implementation neural network classifier in MATLAB

To get an objective comparison of the results, we fixed the number of epochs to 1000 for all simulation scenarios that we ran. In the following part of this section we will show the result of our study on how changing the architecture of the neural network would affect the recognition performance. As the starting point, we used a scaled conjugate gradient back-

propagation learning neural networks. Later on we will show how other training algorithms perform to identify accent.

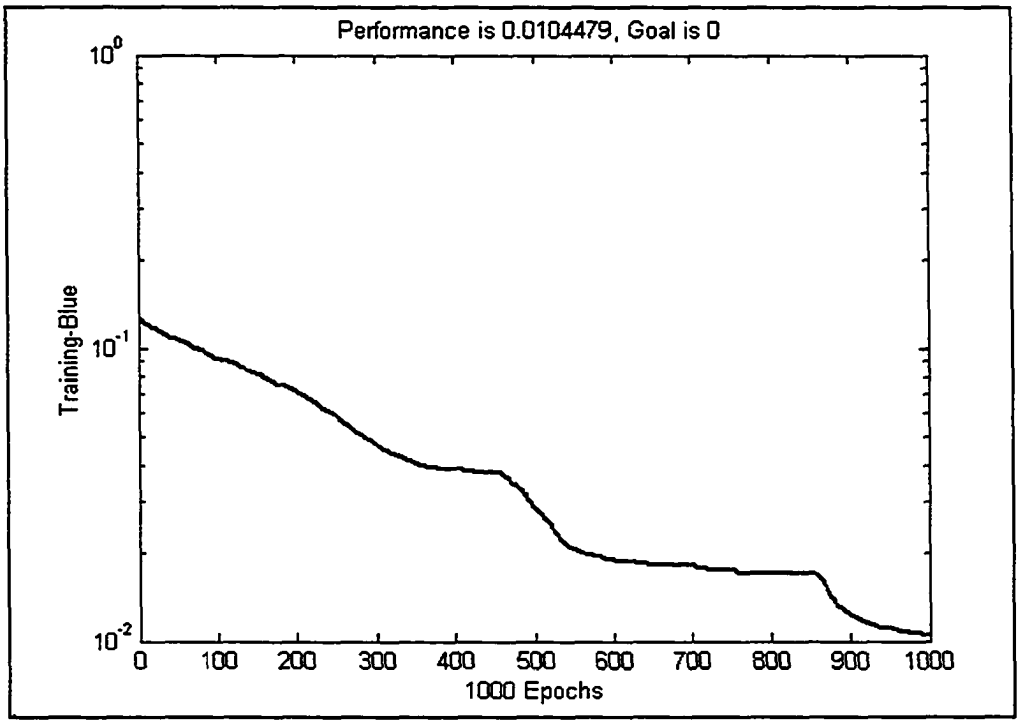

Figure 5-5: Convergence of the neural network in 1000 epochs

5.2.1 Pre-processing of the Speech Data

Pre-processing of the data was similar to the previous experiment with a few minor differences. By parsing the *.phn* files for *SA1* sentence in *TIMIT*, we extracted the phonemes that we needed for the experiment each in a separate file. We later removed the 1024 *NIST WAVE* heading and normalized the samples to be used for *LPC* reflection coefficient calculations.We used a sequential combination of three *MATLAB* functions of *real()*, *poly2rc()* and *lpc()* to calculate the *LPC* reflection coefficients.

5.2.2 The results

We carried out several different experiments to investigate how the model is sensitive to variations of its parameters. The parameters that we studied included the following:

- the combination of phonemes to train and test the neural network classifier;
- the learning algorithm that was used for training;
- the frame length;
- the dispersion of collected frames around the centre of phoneme;
- and the architecture of the neural network.

Combination of phonemes

To establish a baseline and to identify what grouping of the phonemes would result in better accent recognition performance, we carried out 14 different scenarios in two groups of different combination of the three phonemes */aa/*, */er/* and */ix/*. In the first group of experiments, or the phoneme dependent scenarios, we used the same combination of phoneme(s) for both training and testing. For example, for one scenario we used */aa/* and */eh/* to train the network and used a separate test data set of the same set of phonemes to test the trained network. Table 5-1 Identification rate results with the same phoneme on training and testing (M: Male, F: Female) shows the results of the first group f experiments. As we can see from this table, the recognition rate for this particular experience is 75%.

SCENARIO GROUP1	IDENTIFICATION RATE
All 3 phonemes, Male	47.22%
All 3 phonemes, M & F	48.21%
/eh/, /ix/ phonemes, M&F	52.6%
/aa/, /ix/ phonemes, M&F	68.14%
/aa/, /eh/ phonemes, M&F	75.00%
/eh/ phoneme, M&F	80.00%
/ix/ phoneme, M&F	54.05%
/aa/ phoneme, M&F	77.78%

Table 5-1 Identification rate results with the same phoneme on training and testing (M: Male, F: Female)

In the second group, or the phoneme independent identification scenarios, we used all three phonemes in testing i.e. regardless of the group of phonemes that we used to train the network, in the test stage data sets from all three phonemes were used. As we would expect, the recognition rate in the second group of experiments are lower since there is smaller correlations between training and test data sets.

Table 5-2 Identification rate results when testing with all three Phonemes (M: Male, F: Female) shows the results of the second group of experiments. As we expected, the phoneme dependent scenarios produced better accent identification accuracy.

As we can observe from the results, the phonemes */aa/* and */er/* provide better accent discrimination than */ix/*. When the neural network was trained and tested with instances of */er/*, the identification accuracy was 80%.

This validates our assumption that temporal evolution of the vocal tract could constitute good accent discriminative features. When we show our study of the effect of frame dispersion on recognition error rate, this assumption will become more justified.

At the same time, we can observe that the phoneme */ix/* for the classification purpose was not as good as the other two. Figure 5-6 Identification rates for both groups of experiments (M: Male only speakers) shows a comparison of the results for both scenarios.

SCENARIO GROUP 2	IDENTIFICATION RATE
/eh/, /ix/ phonemes, M&F tested with all	26.67%
/aa/, /ix/ phonemes, M&F tested with all	38.67%
/aa/, /eh/ phonemes, M&F tested with all	62.67%
/eh/ phoneme, M&F tested with all	24.00%
/ix/ phoneme, M&F tested with all	10.67%
/aa/ phoneme, M&F tested with all	52.00%

Table 5-2 Identification rate results when testing with all three Phonemes (M: Male, F: Female)

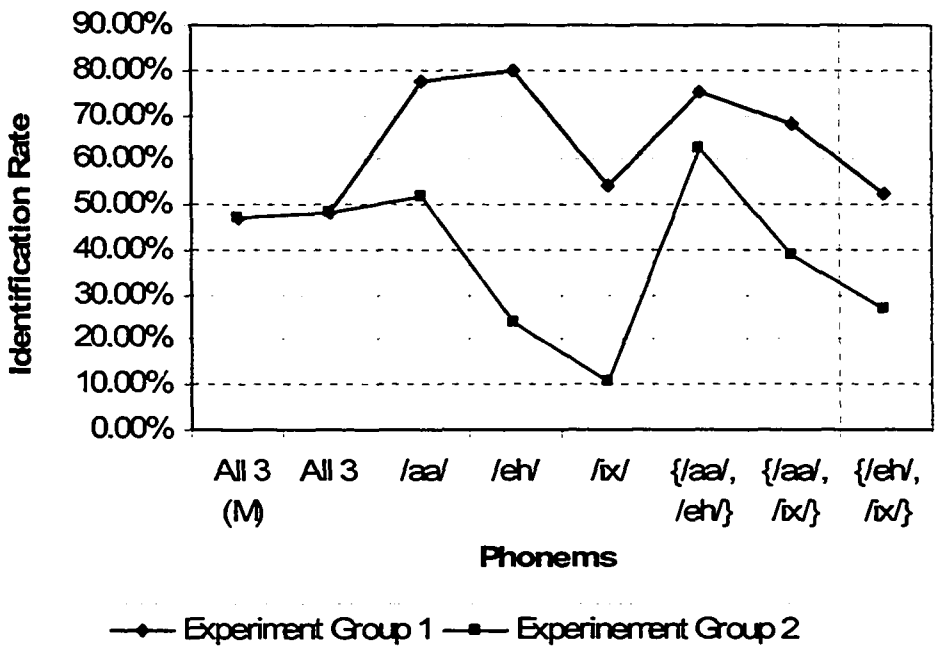

Figure 5-6 Identification rates for both groups of experiments (M: Male only speakers)

We can also see when we bundled */ix/* with any other combination, we obtained lower identification accuracy. Another interesting fact that is illustrated by these results is that using a single gender or both genders does not significantly impact the accuracy of identification. This may be an indication that gender information is embedded into the *LPC* reflection coefficients and that manually separating them does not add any extra information to the classifier.

The experiments this far have resulted in identifying the group of phonemes that can produce a better set of accent discriminating features i.e. */aa/* and */eh/* and */ix/* one that does not show as similarly good accent distinguishing properties as the other two. To conduct these experiments as the base line several choices were made, including the frame size, frame dispersion, the size and architecture of the neural network as well as its learning algorithm. In the follow up experiments, some of the choices were examined and the affect of changing them to other values and parameters was studied. As we will see, with fine tuning these model inputs, we are able to increase its recognition performance.

Neural Network learning algorithm

Once a network is built and initialized with pseudo-random weights and biases, it can be trained to classify patterns. The training is a process of presenting the network with a set of examples of proper classification. During the training, the weights and biases of the network are iteratively adjusted to minimize the network classification error. This error is sometimes called the network performance. The most commonly used performance function for feed

forward networks is mean square error or *MSE*. *MSE* is the average squared error between the network outputs and the target outputs. The training can be performed using different algorithms or functions. Each of these training algorithms can produce different convergence speeds and generalization capabilities in different situations.

The training functions that we examined in our experiments all use gradient descent of the performance function to determine how to adjust the weights to minimize the error. The gradient is determined using back-propagation technique, which involves performing computations backwards through the network which is derived using the chain rule in calculus.

The gradient descent algorithm can be implemented either in batch or in incremental modes. In the batch mode all inputs are presented to the network before the network parameters are adjusted. In incremental mode, the gradient is calculated and network parameters are adjusted after each input is presented to the network. We will examine 6 different training algorithms and attempt to identify the ones that perform better in our accent identification model:

- Gradient Descent training (*TRAINGD*)

Gradient Descent training or GD is the basic implementation of the batch steepest descent training algorithm in *MATLAB*. The weights and biases are updated in the direction of the negative gradient of the performance function. This algorithm's convergence is not very fast and it is prone in falling into local minima. Other enhancements to this algorithms have been suggested to make its convergence faster and to make it smarter in avoiding local minima.

- Gradient Descent with Momentum (*TRAINGDM*)

As an enhancement to basic GD algorithm, this function incorporates analysing error surface trends in order to select the direction that the network parameters should be adjusted. This is called momentum. Momentum allows the network to respond not only to the local gradient, but also to the recent trends in the error surface. Without momentum the neural network can fall in a shallow local minima. With momentum the network might get a chance to slide through these minima and generally it can converge faster.

- Gradient Descent with Adaptive learning rate (*TRAINGDA*)

Although the performance of a learning algorithm is very sensitive to it's learning rate, with the basic GD and GDM algorithms the learning rate is kept constant during the training. If the learning rate is too small, the convergence takes a long time. If the learning rate is too large, it might skip over the optimal point. By changing the learning rate during training, we can improve the performance of the steepest descent algorithm. The gradient descent algorithm with adaptive learning curve will choose the largest learning step size but it makes it a function of the local error surface. Adaptive learning rate can cause the network to converge faster.

- Gradient Descent with Adaptive learning and Momentum (*TRAINGDX*)

This algorithm is a combination of the adaptive learning rate and the momentum. It generally converges faster than either one alone, and faster than the basic gradient descent algorithm.

- Resilient Back-propagation training (*TRAINRP*)

Most neural networks use a sigmoid transfer function in their hidden layers. The sigmoid family of functions map an infinite input range to a finite output range. In order to do this mapping, their slope must approach zero as the input gets large and hence the derivatives get very small at these points. This can cause a problem in gradient descent algorithms since they evolve very slowly because the changes in weights and biases are very small even when they are far from their optimal values. To address this issue resilient propagation training is introduced. In this enhancement, the steepest gradient descent only uses the sign of the derivatives to update the weights and biases (to increase or decrease the weights and biases). The magnitude of the update is determined by another method or set by a parameter as an input value to the model. This enhancement can make the standard GD algorithm much faster to converge.

- Scaled Conjugate Gradient training (*TRAINSCG*)

In basic training algorithms weights and biases are adjusted along the direction of the steepest gradient descent. This does not necessarily generate the fastest convergence. The conjugate training algorithms search in conjugate directions which generally generate faster convergence. Scaled conjugate gradient training is a search optimized algorithm that was developed by Moller [49] and it converges faster than most of the algorithms that we have experimented with.

Since the performance of each of the experiment can be affected by the initial point that the network starts learning, for each experiment, we carried out at least three runs and averaged

the results to stabilize each run and get a better and more judgmental performance comparison. In the case that the results don't stabilize, it has been suggested that the output of up to twenty runs to be averaged. If an outlier was detected during any of the runs, it was taken out of the averaging. In each of these experiments, the training was capped at 1000 epochs. As is shown in Table 5-3 Performance of different training functions and Figure 5-7 Performance of different training functions, the scaled conjugate learning algorithm produces the lowest error rate.

Training Algorithm	TRAINGD	TRAINGDM	TRAINGDA	TRAINGDX	TRAINRP	TRAINSCG
Error count	64	62	54	46	31	27
Error rate	85%	82%	72%	61%	41%	36%

Table 5-3 Performance of different training functions

Frame Length

To establish a base line, we originally used frames of 30 *ms* long. The speech signal is assumed to be pseudo-stationary in the range of 10 ms to 20 ms. We will see how this fact would affect the performance of the accent recognition. As we already discussed, we collected three frames from each phoneme samples all having equal length. Changing the length of the frame will change several elements that can potentially affect the accuracy of the accent identification.

First, if the frame is too short, there will not be enough speech data to calculate a meaningful feature vector that is representative of the random process behavior of the speech signal. On the other hand shorter frames would allow us to distant frames more apart from each other which in turn would increase the likelihood that the temporal evolution of the feature vector is captured. Shorter frames are less likely to overlap specially in phoneme samples that have a shorter length.

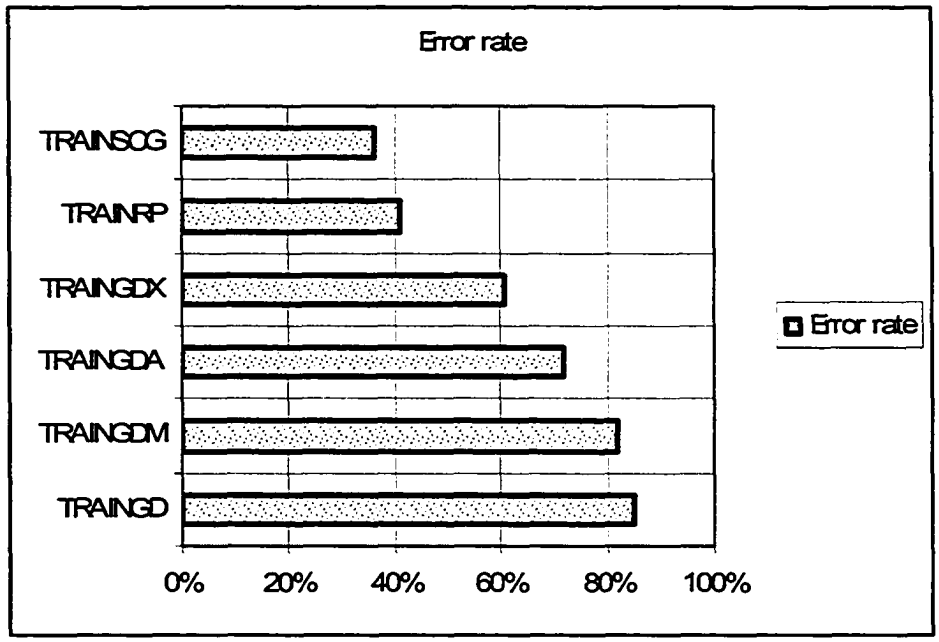

Figure 5-7 Performance of different training functions

Second, if the phoneme is too long, the pseudo-stationary assumption of the speech signal is compromised. Longer frames would also increase frame overlap and can in turn lessen the degree that they capture the temporal evolution of the feature vector. We can clearly see the effect of varying frame length in Table 5-4 and Figure 5-8. It should be noted that all three

phonemes were used in studying the effect of frame length to ensure that we have a broader exposure to capture any effect that any of these phonemes might have on the results. We can see that when the frame length is too short i.e. 20 samples which equals only 2.5 *ms*, the error rate is as high as 43%. As we increase the frame length, the error rate drops and its minimum is at 120 samples or 15 *ms*. At this frame length, the error rate is 31% which in comparison to our original error rate of 53% for this scenario is a big improvement. As we would expect, increasing frame length would degrade the pseudo-stationary property of speech signal and it would also increase frame overlap that both have negative impact on the recognition error rate. When the frame length is increased to 360 samples (or. 45 *ms*), the error rate jumps to 61%.

Frame Length (in Samples)	20	60	120	160	200	240	360
Fame Duration (in ms)	2.5	7.5	15	20	25	30	45
error %	43%	38%	32%	36%	43%	53%	61%

Table 5-4 The effect of frame length on recognition performance

Frame Dispersion

In this thesis the core information that is collected from speech signals in order to capture accent discrimination is the way the speaker changes his/her speech production organs during uttering a phoneme. In our experiments, we attempted to capture this temporal variation by collecting multiple frames from each phoneme sample.

Multiple frames are in fact multiple modelled representations of the talker's speech production organs in different times during the utterance. The farther the frames are dispersed, the more chance we have to capture this temporal evolution given that the frames are long enough to contain meaningful data to model the speech production organs.

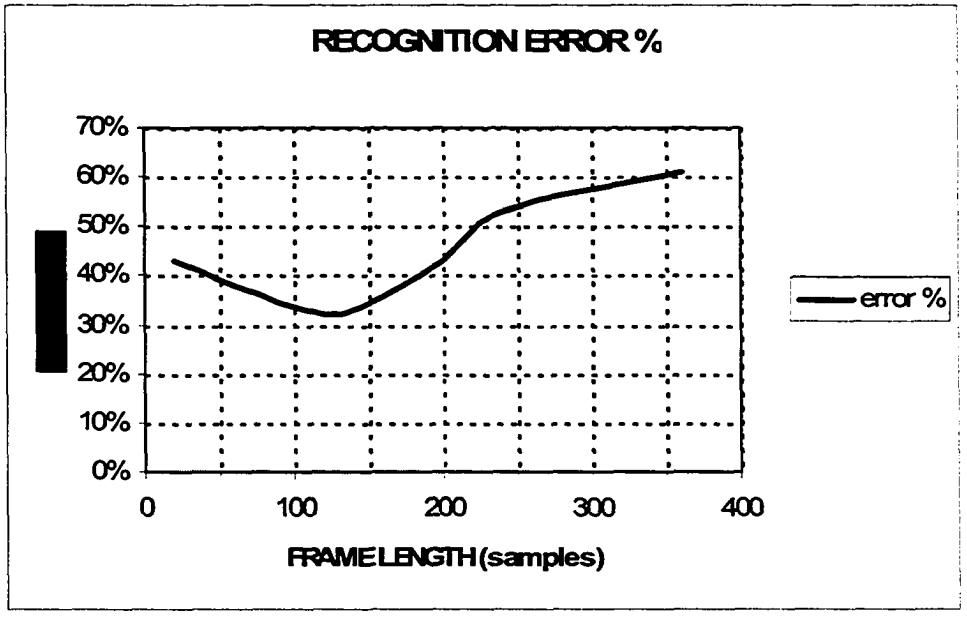

Figure 5-8 Change of recognition performance in response to change of frame length

In studying the effect of frame dispersion on accent recognition performance in our model, we fixed the frame length to 120 samples.

This is the frame length that produced the minimum recognition error rate. We used all three phonemes to capture any potential adverse effect that they might have on the results. Since the frame length is fixed, depending on the length of the phoneme, we get different distances between the adjacent frames. For some of the shorter phoneme samples, we may not be able to eliminate frame overlap even when we use maximum frame dispersion. We used four different levels of frame dispersion in this experiment as shown in Figure 5-9. It should be noted that in this figure only 3 different dispersion levels are illustrated.

We started with maximum dispersion meaning that one frame was centred in the middle of the phoneme and the other two were pushed to the edges of the phonemes. This is shown in Figure 5-9-a. On the opposite side, we had minimum dispersion which was complete overlap of all three frames which is shown in Figure 5-9-c.

We allowed two levels of frame dispersion in the middle of these two extreme situations similar to what is shown in Figure 5-9-b. As we would expect, maximum frame dispersion generated better recognition performance since it allows for higher level of capturing the temporal variation of the vocal tract trajectory from its modelled *LPC* reflection coefficients. Table 5-5 and Figure 5-10 show the results.

When the distance between two neighboring frames is maximum i.e. we have maximum frame dispersion, we observe the best recognition performance of 31% which is what we saw in our previous experiment.

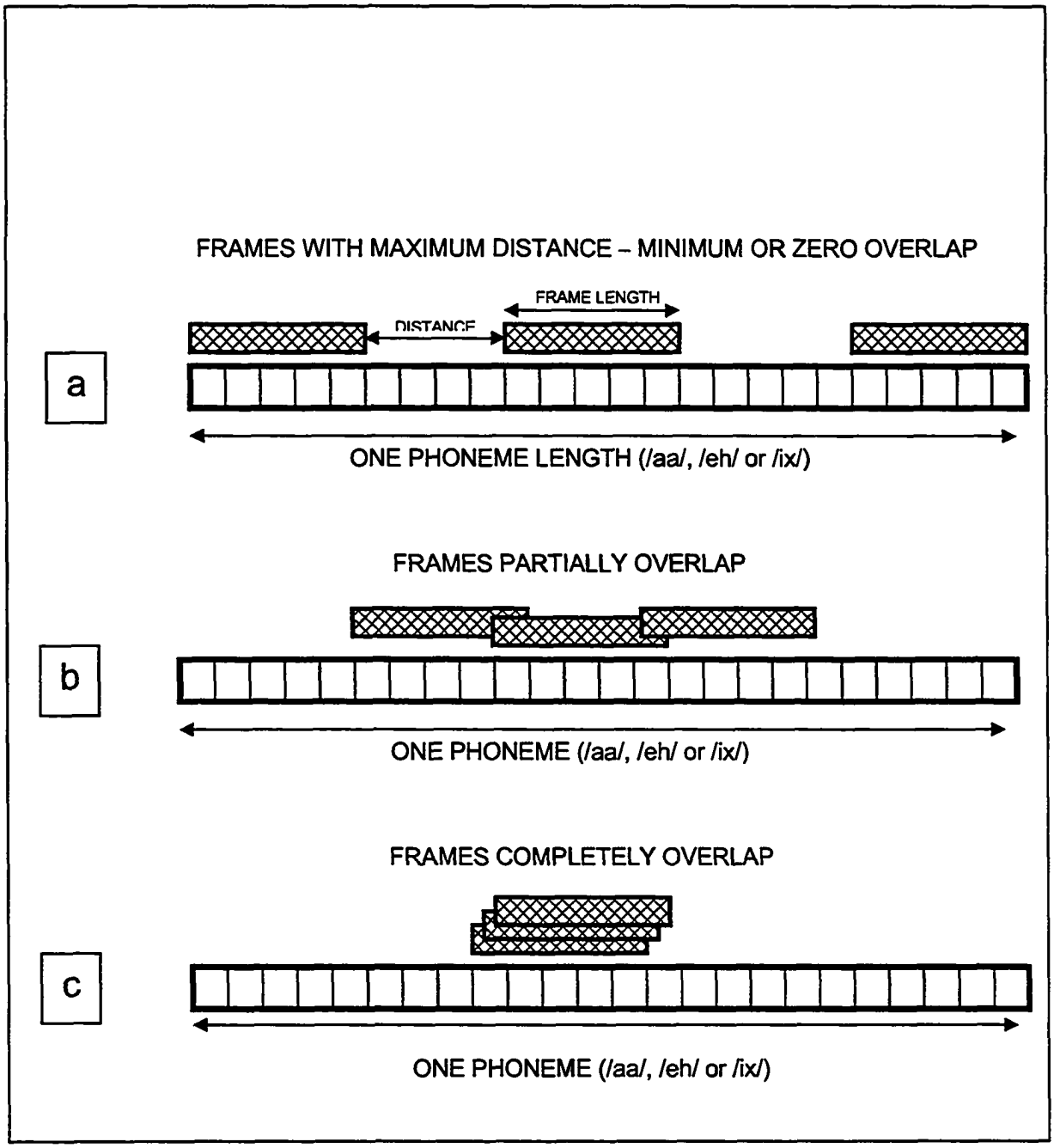

Figure 5-9 Change of recognition performance in response to change of frame dispersion

Temporal Distance (Frame dispersion)	100%	66%	33%	0%
Error Rate	31%	42%	48%	67%
Error Count (out of 75)	23	32	36	50

Table 5-5 The effect of frame dispersion on recognition performance

As we collected the frames closer and closer to each other e.g. with 66% and 33% dispersion, we observed a higher recognition error rate of 42% and 48% respectively. When we had zero frame dispersion or complete overlap, we observed the error rate of 67%.

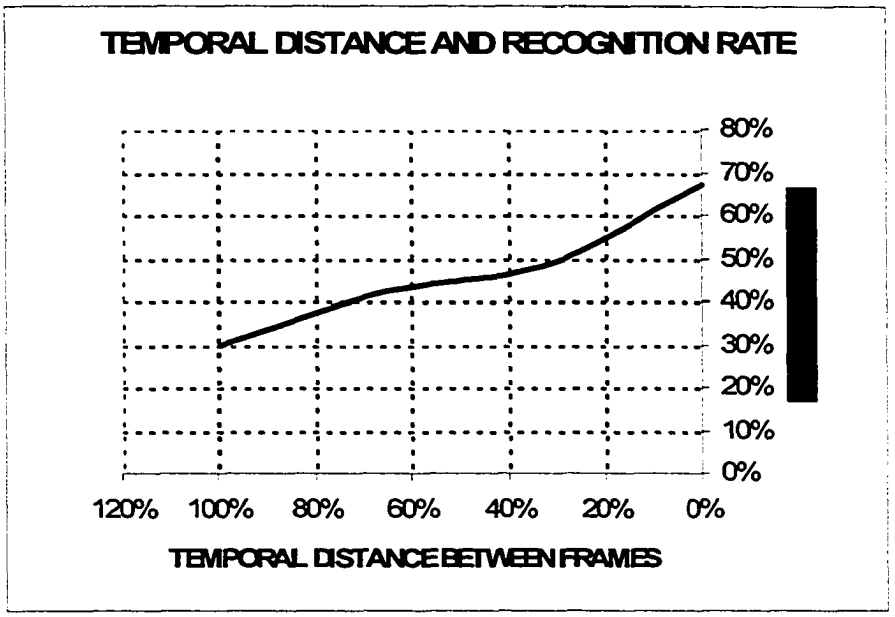

Figure 5-10 Change of recognition performance in response to change of frame dispersion. (Frame length is set to 120)

Architecture and size of the Neural Network

Size and architecture of a neural network can impact its performance greatly. As it was discussed in Chapter 2, there is no widely accepted scientific procedure to determine the appropriate size and structure of the neural network for a particular application. The usual procedure involves a trial and error process to determine the right size. As was shown earlier in this chapter, to build a benchmark, we originally used a network with one hidden layer with 220 nodes in it. Later on we examined how changing the size of the network, mainly its hidden layer, can affect the ability of the network to learn and generalize the patterns. As it was noted earlier, to produce an objective and comparable set of results, the number of epochs was fixed to 1000. We also used the frame length of 120 which as we saw earlier produced the lowest error rate. For all the phoneme samples, the frame dispersion was also maximized to leverage the temporal evolution of the feature vectors to the extent that was possible. It was shown earlier that this approach will minimize recognition error rate. As is shown in Table 5-6 The effect of hidden layer size on recognition performance and Figure 5-11 Error rate as function of the number of nodes in the hidden layer, when the number of nodes in the hidden layer increases, the network's ability to learn the patterns in the training data series and to generalize these patterns increase as well. We notice a sharp increase in accent recognition rate as nodes are added from 1 to 50. When we add nodes to the network above 50, the rate of increase in the recognition rate is not as fast as it is in the range below 50.When there is only 1 node in the hidden layer, the recognition error rate is very high at 87%. (This equals a recognition rate of only 13%). As more nodes are added, we observe improving recognition error rate of 80% at 5, 76% at 10, 53% at 25 and 36% at 50.

Nodes in Hidden Layer(s)	1	5	10	25	50	100	220	300
Error count	65	60	57	40	27	26	24	22
Error rate	87%	80%	76%	53%	36%	34%	32%	29%

Table 5-6 The effect of hidden layer size on recognition performance

From this point the improvement slows down and for adding another 50 nodes, the error rate reduces only to 34%. When there are 300 nodes placed in the hidden layer, the error rate is 29%. We can conclude that at 50, this network is performing relatively satisfactory. This can be attributed to over fitting issue that was discussed in Chapter 2. At this stage, the network is too adapted to training inputs that can not effectively classify the test inputs.

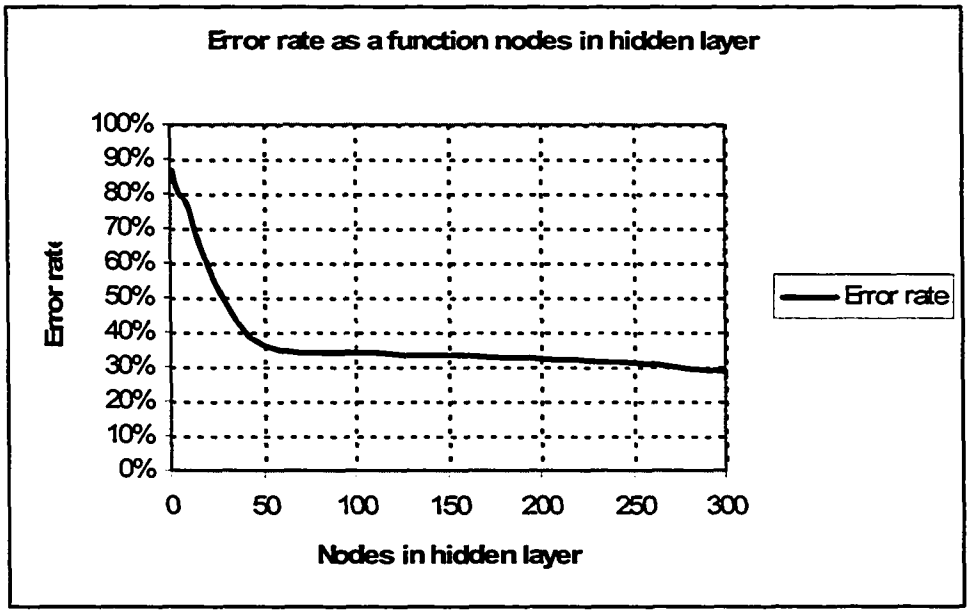

Figure 5-11 Error rate as function of the number of nodes in the hidden layer

5.3 Summary

Accent identification is a problem closely related to language identification. Automatic accent identification can help in improving the performance of speech recognition systems, by adapting the recognition engines to the appropriate accent. It has also been used as a powerful tool in forensic and crime investigations. In this chapter we demonstrated that a neural network classifier can be used as the core learning module for an accent identification system. In an attempt to build a base line model, we showed that a feature space that is spawned from temporal evolution of vocal tract attributes that are modeled using *LPC* coefficients, show good accent discriminating properties to train the classifier. We also showed that such a system will perform relatively well even in recognizing dialectal differences in the English language.

In further experiments we illustrated how changing different assumptions and inputs to the base line mode would affect the recognition error rate. We showed that the training algorithm that was used to train the classifier could impact the results. Our findings showed that a training algorithm based on scaled conjugate gradients introduced by Muller produced the lower error rate compared to the basic gradient descent algorithm and some of its enhanced variations. We showed that frame length and frame dispersion are critically important in lowering the accent identification error rate. Although we started with a frame length of 30 *ms* as the base line experiment, we showed that the frame length of 15 *ms* would generate lower error rate than other lengths. Shorter frame lengths don't capture enough meaningful information from the speech signal to robustly represent the speech signal behaviors. Longer frame lengths compromise the pseudo-stationary property of the speech

signal that is assumed in speech signal processing. They also increase the frame overlap which in turn lowers the ability of the model to learn the temporal evolution of the feature vector space. We also showed that the recognition error rate is minimum when the frame dispersion is maximized. This is consistent with our core assumption that the temporal evolution of speech production organs which is inherently embedded in LPC reflection coefficients can be a viable resource for accent discrimination and classification.

CHAPTER 6 Future Work and Conclusions

6.1 Discussion of results

As discussed in Chapter 4, performance of an automatic speech recognition system can be improved in the case of non-native speakers or speakers with distinct dialects by adaptating the recognition engine to the accent of the speaker. This can be achieved by training multiple engines, one for each accent, identifying the accent of the incoming speech, and routing it to the engine that is trained for that accent.

In Chapter 5 it was shown that an accent classification model could be built by using the temporal evolution of the model of the talker's speech production organs. It was shown that a neural network classifier can give relatively good results. It was also illustrated that the extent that the temporal variations are captured, affects the accuracy of the accent identification.

6.2 Thesis Contributions

In this research work we set up an experimental environment based on *HMM* speech recognizers and *TIMIT* speech corpora to study the effects of present accent on the performance of speech recognition systems. We validated the assumption that accent has

negative impact on the performance of a speech recognition system that is trained by a cocktail of different accent speech tokens which is the case for many commercially available solutions. We later showed that by introducing a mechanism to identify the input speech accent, and by selecting a speech recognition engine that is trained for the identified accent, we can improve the speech recognition performance.

In the second stage of this work we proposed an accent classification approach based on tracking speaker's vocal tract variations during vowel utterance. We used *LPC* feature trajectories as a model to track temporal evolution of the vocal tract. Based on this work a paper was presented at the *IEEE 2005 conference on Computational Intelligence in Homeland Security and Personal Safety*. This paper was published in the proceeding of the conference.

Our complete proposed solution is composed of an accent classifier as a pre-processing stage with a bank of engines that are each trained with a particular accent.

6.3 Suggestions for Future Work

In this thesis accent adaptation was investigated on American English accents. There is an opportunity for future work to examine the proposed accent adaptation approach on databases of speech samples from speakers with different native languages. Another opportunity is to experiment with adapting the same recognition engine to different accents rather than using multiple instances of trained recognition engines. In accent classification, a future study can be carried out on leveraging the output of the sigmoid functions in the neural

network classifier to reduce error rate rather than applying them to binary decision functions as this thesis did. Another opportunity is to use other supervised and unsupervised classification methods as the classifier engine for accent identification.

Appendix A: Some Useful Tools

In this appendix the source listing for *w_xtract* and *p_xtract* utilities are provided. The tool *w_xtract* was developed to parse <*.wrd*> files in the *TIMIT* database and extract words from Sa1 sentences and store them in separate files. The tool *p_xtract* does the same things for phonemes. Both tools are written in Borland's Turbo Pascal.

7.1 Source listing for *w_xtract* utility

```
Program WordExtract;

 Uses Strings, CRT, WinDos, Dos, HDefines, Defines, Utility, CoefAnal, Access, Math,
DSPU;

(*
 This program is a utility for extracting words from TIMIT database:

  1. Usage: w_extract <wav file> <word no.>
  1. Reads the .wrd file for the .wav file in the command line param
```

```
  2. From the original .wav file, copies the header and samples from
    Begin sample to End sample in a target file 'word'.wan

  *)

const

 DataFileNo = 3;
 MaxDNo    = 2;
 MaxL1     = 8;
 (*MaxL2     = 64;*)
 HdrLen    = 512;

 L1DArray: Array[1 .. MaxL1]  of String[32] = (
 'DR1',
 'DR2',
 'DR3',
 'DR4',
 'DR5',
 'DR6',
 'DR7',
 'DR8'
 );


 DNArray : Array[1 .. MaxDNo] of String[32] = (

 's01-DR1M',
 's26-DR5M' );
```

```
var
 FNo, dNo, i, ii, k, Code, ChCount : Integer;
 Min, Max , J, BegSamp, EndSamp, Count, SpCount: LongInt;
 Min_i, Max_i : Integer;
 Finished: Boolean;
 LogEnergy : Single;
 SMPFile, WAVFile, TgtFile: File of Integer;
 WRDFile, TXTFile: Text ;
 byteBuffer: integer;
 MemBlk: Array[1 .. FrameLength] of Integer;
 SPos: Array[1 .. 4] of Integer;
 wavParam, wrdParam, txtParam, tgtParam : String[128];
 LineBuff, FilePath, DirPath, Speaker, Sex, Region  : String[128];
 Ch      : Char;
 Dir     : PChar;


begin

  SetCBreak(True);
  (*ClrScr;*)
  If ParamCount=0 Then
  Begin
   WriteLn('Usage: W_EXTRACT.exe <wav file> <word no.>');
   Halt(1);
  End;
  wavParam := paramstr(1);
```

```
(* support forUnix style file names *)

For i:=1 to Length(wavparam) do
 If wavParam[i] = '/' Then wavParam[i] := '\';

Val(Paramstr(2), K, Code);
wrdParam := copy(wavParam, 1, length(wavParam)-3)+ 'wrd';
txtParam := copy(wavParam, 1, length(wavParam)-3)+ 'txt';

Assign(WRDFile, WRDParam);
Assign(WAVFile, wavParam);

(* read the .wrd file and figure out begin and end sample # *)

Reset(WRDFile);
Reset(WAVFile);

(* setting the file pointer to the begining of the word *)

LineBuff := ''; I:=0; Count := 1; ChCount := 0; SpCount:=0;
While not eof(WrdFile)
Do
Begin
 read(WRDFile, Ch);
 Inc(ChCount);
 If Ch=' 'Then
 Begin
  SPos[Count] := ChCount;
  If Count < 3 Then Inc(Count);
```

```
End;
If Ch=Chr(10) Then
Begin
 Inc(I);
 Count := 1;
 If (I=K) Then
 Begin
  WriteLn('Processing Word: ', K, ' of file: ', WRDParam,
       ' and the word is: ', LineBuff);

  Val(Copy(LineBuff, 1, SPos[1]-1), BegSamp, Code);
  Val(Copy(LineBuff, SPos[1]+1, SPos[2]-SPos[1]-1), EndSamp, Code);

  (* Get the file path and build the target file name *)

  FilePath := wavParam;



  While (Pos('\', FilePath)<>0) AND (Length(FilePath)>=1)
   Do Delete(FilePath, 1,1);

  FilePath := Copy(wavParam, 1,
            Length(wavParam)-Length(FilePath));

  {
  Region  := Copy(FilePath, 3, 3);
  Sex     := Copy(FilePath, 7, 1);
  Speaker := Copy(FilePath, 8, 4);
```

```
FilePath := 'c:\speech\test\' + Sex + Region + Speaker;

WriteLn('Creating Directory: ', FilePath);

DirPath := FilePath+'#0';
Dir := @DirPath[1];
CreateDir(Dir);
}
tgtParam := Copy(LineBuff, SPos[2]+1, Length(LineBuff)-SPos[2]);
tgtParam := FilePath + {'\'} + tgtParam;
tgtParam := concat(tgtParam, '.WAV');

WriteLn('Creating target file: ', tgtParam);

(*Assign(TGTFile, tgtParam);*)
Assign(TGTFile, tgtParam);
ReWrite(TGTFile);

(* Handling the WAV header *)

for J := 1 to HdrLen do
begin
 read (WAVFile, MemBlk[1]);
 Write(TGTFile, MemBlk[1]);
end;

(* Skipping to the begin sample *)

for J := 1 to BegSamp-1 do
begin
```

```
        read(WAVFile, MemBlk[1]);
        MemBlk[1]:=1;
      End;
      for J := 1 to EndSamp-BegSamp do
      begin
        read(WAVFile, MemBlk[1]);
        Write(TGTFile, MemBlk[1]);
      End;

    End
    Else
    Begin
      LineBuff := '';
      ChCount := 0;
      Count := 1;
      Continue;
    End;
    LineBuff := '';
    ChCount := 0;
    Count := 1;
  End
  Else
  Begin
    LineBuff := concat(LineBuff,Ch);
  End;
 End;
end.
```

7.2 Source listing for p_*xtract* utility

```
Program PhonemeExtract;

 Uses Strings, CRT, WinDos, Dos, HDefines, Defines, Utility, CoefAnal, Access, Math,
DSPU;

(*
 This program is a utility for extracting vowel phpnems from TIMIT database:

   1. Usage: p_extract <wav file> <phoneme, no.>
   1. Reads the .phn file for the .wav file in the command line param
   2. From the original .wav file, copies the header and samples from
     Begin sample to End sample in a target file 'phoneme'.wan

   *)

const

  DataFileNo = 3;
  MaxDNo     = 2;
  MaxL1      = 8;
  (*MaxL2     = 64;*)
  HdrLen     = 512;
  FL = 960(*600*); { 2x15ms frames, each frame 480 bytes}

  L1DArray: Array[1 .. MaxL1]  of String[32] = (
```

```
'DR1',
'DR2',
'DR3',
'DR4',
'DR5',
'DR6',
'DR7',
'DR8'
);

(*L2DArray: Array[1 .. MaxL2] of String[32] = (


);*)

DNArray : Array[1 .. MaxDNo] of String[32] = (

's01-DR1M',
's26-DR5M' );


{FNameArray : Array[0 .. MaxWords] of String[8] =
 ('Suit', 'Greasy', 'Year' );}


var
 FNo, dNo, i, ii, k, Code, ChCount : Integer;
 Min, Max , J, BegSamp, EndSamp, Count, SpCount: LongInt;
 Min_i, Max_i , SerialCount : Integer;
 Finished: Boolean;
```

```
{SPFrame : SpeechFrame;}
LogEnergy : Single;
SMPFile, WAVFile, TgtFile, SERFile: File of Integer;
WRDFile, TXTFile: Text ;
byteBuffer: integer;
MemBlk: Array[1 .. FrameLength] of Integer;
SPos: Array[1 .. 4] of Integer;
wavParam, wrdParam, txtParam, tgtParam : String[128];
LineBuff, FilePath, DirPath, Speaker, Sex, Region, Serial : String[128];
PhStr: String[32];
Ch      : Char;
Dir     : PChar;

DirInfo: SearchRec;

begin

  SetCBreak(True);
  (*ClrScr;*)
  If ParamCount=0 Then
  Begin
    WriteLn('Usage: P_XTRACT.exe <wav file> '{<Phoneme no.>});
    Halt(1);
  End;
  wavParam := paramstr(1);

  (* support for Unix style file names *)

  For i:=1 to Length(wavparam) do
    If wavParam[i] = '/' Then wavParam[i] := '\';
```

```
{Val(Paramstr(2), K, Code);}
wrdParam := copy(wavParam, 1, length(wavParam)-3)+ 'phn';
txtParam := copy(wavParam, 1, length(wavParam)-3)+ 'txt';

WriteLn('TEST: Phn file: ', wrdParam, '  txt file:', txtParam);

Assign(WRDFile, WRDParam);
Assign(WAVFile, wavParam);
Assign(SERFile, 'd:\speech\serial.dat');


FindFirst('d:\speech\serial.dat', Archive, DirInfo);
If DosError <> 0 Then {File doesn't exist, Create it}
Begin
  Rewrite(SERFile);
  SerialCount:=0;
End
Else
Begin
  Reset(SERFile);
  Read (SERFile, SerialCount);
End;


(* read the .wrd file and figure out begin and end sample # *)

Reset(WRDFile);
Reset(WAVFile);
```

```
(* setting the file pointer to the begining of the phoneme *)

LineBuff := ''; I:=0; Count := 1; ChCount := 0; SpCount:=0;
While not eof(WrdFile)
Do
Begin
 read(WRDFile, Ch);
 Inc(ChCount);
 If Ch=' 'Then
 Begin
  SPos[Count] := ChCount;
  If Count < 3 Then Inc(Count);
 End;
 If Ch=Chr(10) Then
 Begin
  Inc(I);
  Count := 1;
  PhStr := Copy(LineBuff, SPos[2]+1, ChCount-SPos[2]-1);
  If ({I=K} (PhStr='aa') OR (PhStr='eh') OR (PhStr='ey') OR (PhStr='ix')
      OR (PhStr='bd') ) Then
  Begin

    WriteLn('Processing Phoneme: <', PhStr, '> of file: ', WRDParam);

    Val(Copy(LineBuff, 1, SPos[1]-1), BegSamp, Code);
    Val(Copy(LineBuff, SPos[1]+1, SPos[2]-SPos[1]-1), EndSamp, Code);

    (* Get the file path and build the target file name *)

    FilePath := wavParam;
```

```
While (Pos('\', FilePath)<>0) AND (Length(FilePath)>=1)
 Do Delete(FilePath, 1,1);

FilePath := Copy(wavParam, 1,
          Length(wavParam)-Length(FilePath));
Region   := Copy(FilePath, 5, 1);
Sex      := Copy(FilePath, 7, 1);
Str (SerialCount:4, Serial);
For J:=1 To 4 Do
  If Serial[j]=' ' Then Serial[j] := '0';


FilePath := 'd:\speech\accent\phn_tst4' ;

tgtParam := PhStr + Region + Sex+ Serial;
tgtParam := FilePath + '\' + tgtParam;
tgtParam := concat(tgtParam, '.WAV');


WriteLn('Creating target file: ', tgtParam);

Assign(TGTFile, tgtParam);
ReWrite(TGTFile);


(* Handling the WAV header *)

for J := 1 to HdrLen do
begin
```

```
 read (WAVFile, MemBlk[1]);
 Write(TGTFile, MemBlk[1]);
end;

(* Skipping to the begin sample *)

{Reset(WAVFile)};


for J := 1 to BegSamp-1 do
begin
 read(WAVFile, MemBlk[1]);
 MemBlk[1]:=1;
End;


(* extract 2 frames (each 480 bytes, 240 samp, 15ms) in the middle*)
(* FL = 960 *)


WriteLn('Phoneme Length: ', (EndSamp-BegSamp):6);




Begin
 for J := BegSamp to EndSamp do
 begin
   read(WAVFile, MemBlk[1]);
   Write(TGTFile, MemBlk[1]);
```

```
 End;
End ;



{

If EndSamp-BegSamp <= FL Then
Begin
 for J := 1 to EndSamp-BegSamp do
 begin
  read(WAVFile, MemBlk[1]);
  Write(TGTFile, MemBlk[1]);
 End;
End
Else
Begin

 For J := 1 to  Trunc((EndSamp-BegSamp-FL)/2) Do
 Begin
   (* Skip to the beginning of the middle frame *)
  read(WAVFile, MemBlk[1]);
  MemBlk[1]:=1;
 End;

 for J := 1 To FL do
 begin
  read(WAVFile, MemBlk[1]);
  Write(TGTFile, MemBlk[1]);
 End;
```

```
      End;
}
      Reset(WAVFile);
      Inc(SerialCount);
      If SerialCount = 9999 Then SerialCount :=0;
      Reset(SERFile);
      Write(SERFile, SerialCount);
      close(TGTFile);
     End
     Else
     Begin
      LineBuff := '';
      ChCount := 0;
      Count := 1;
      Continue;
     End;
     LineBuff := '';
     ChCount := 0;
     Count := 1;
    End
    Else
    Begin
     LineBuff := concat(LineBuff,Ch);
    End;

   End;

   Close(SERFile);

end.
```

Glossary

Accent - the relative prominence of a particular syllable of a word by greater intensity or by variation or modulation of pitch or tone

Affricates - sounds which result from the concatenation of a stop sound, followed by a fricative sound which shares the same place of articulation as the stop.

Allophones - phonemes that represent the same sound, but are produced by distinctly different positions of the vocal tract articulators.

Basilar membrane - an inner ear membrane which serves to transfer the mechanical disturbances of the eardrum into spatio-temporal disturbances along its length.

Cilia - a network of tiny filaments which transduce the spatio-temporal basilar membrane disturbances into electrical impulses.

Co-articulation - the overlapping of adjacent phonemes in a sentence caused by anticipatory movement of a given articulator.

Cochlea - the first of two fluid filled chambers in the inner ear which vibrate in response to the mechanical disturbances of the eardrum.

Dialect - a regional or social variety of a language distinguished by pronunciation, grammar, or vocabulary

Diphones - pairwise combinations of phonemes.

Diphthongs - sounds produced by varying the vocal tract configuration smoothly from one vowel position to another.

Frame - a finite segment of speech samples, usually about 10.0 to 20.0 msec in duration. The speech signal is generally considered to be statistically stationary within a frame.

Fricatives - sounds produced as a result of turbulence created by forcing glottal excitation through a narrow vocal tract opening.

Glottis - space in the larynx between the vocal folds.

Hybrid encoder - a speech encoder which uses aspects of both waveform and source encoding.

Intonation - pitch variation which occurs during normal conversation.

Manner of articulation - the manner in which the vocal tract articulators interact to produce a given phoneme.

Nasal - voiced sounds which are produced with the vocal tract totally restricted at some point in the oral cavity.

Phone - the physical sound produced when a phoneme is uttered.

Phoneme - an abstract unit of speech used to designate one of the 20 - 40 distinctive sounds which make up a language.

Pinna - the externally visible part of the ear.

Pitch - an acoustic phenomena of a sound which enables a listener to percep-tually place it on a scale going from low to high.

Plosives - see *stops.*

Prosodic features - Intensity, duration and intonation of a given segment of speech.

semi-vowel sounds - glides and liquids.

Source encoder - a speech encoder which attempts to reduce the number of data bits necessary for transmission of a speech signal, by relying on a model of the speech signal.

Speech encoder - a device which converts an analog speech signal into a series of bits.

Speech chain - the complete set of events which occur when one person uses speech in order to convey a message to another person.

Speech synthesis - artificial simulation of the speech production process.

Stops - sounds generated when air pressure built up behind a total vocal tract constriction is released.

Unvoiced speech - see *voiceless speech.*

Vector quantization - a source encoding technique which quantizes groups of signal samples, rather than each sample individually.

Voiced speech - speech generated as a result of vocal fold vibration. voiceless speech - speech generated without vocal fold vibration.

Vocal folds - two small muscular folds located in the larynx. Also known as the vocal cords.

Vocal tract – The airway used in the production of speech, especially the passage above the larynx, including the pharynx, mouth, and nasal cavities.

Vowels - sounds produced by exciting a fixed vocal tract with a periodic excitation produced by vocal cord vibration.

Waveform encoder - a speech encoder which attempts to create a facsimile reproduction of the input speech signal through direct digitization of the speech signal.

A List of Acronyms

ADC	Analog To Digital Converter
ADPCM	Adaptive Differential Pulse Code Modulation
ANN	Artificial Neural Network
APC	Adaptive Prediction Coding
ASR	Automatic Speech Recognition
bps	bits per second
CELP	Code Excited Linear Prediction
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DPWT	Discrete Parameter Wavelet Transform
DWT	Discrete Wavelet Transform
FWT	Fast Wavelet Transform
FM	Frequency Modulation
FT	Fourier Transform

GD	Gradient Descent
GMM	Gaussian Mixture Model
HMM	hidden Markov Model
IIR	Infinite Impulse Response
LP	Low Pass (Filter)
LPC	Linear Predictive Coding
MLP	Multi Layer Perceptron
MSE	Mean Square Error
OR	Operational Research
PCM	Pulse Coded Modulation
pdf	Probability Density Function
TIMIT	Texas Instrument Massachusetts Institute of Technology (Corpora)
VQ	Vector Quantization
VTLN	Vocal Tract Length Normalization

Bibliography

[1] Thomas W. Parsons, "Voice and Speech Processing", Mcgraw Hill Series in Electrical and Computer Engineering, 1986

[2] "Bell Canada: Emily – Bell Canada's Voice Recognition for 310-BELL Customer Care", Technical Report, CIPA-Canadian Information Productivity Awards, www.cipa.com, 2004

[3] D. O'Shaughnessy, "Speech Communication: Human and Machine", Addison-Wesley 1987;

[4] L. R. Rabiner, R. W. Schafer, "Digital Processing of Speech Signals", Englewood Cliffs, New Jersey, Prentice Hall, 1987

[5] P. R Kinnear, BSc MSc PhD CPsychol AFBPsS teaching web site, University of Aberdeen, UK, www.abdn.ac.uk/psychology/people/emeritus/pkinnear.shtml

[6] N. Weiner, Cybernetics, or control and communication in the animal and machine, 1948

[7] L. Rabiner, B.H. Juang, "Fundamentals of Speech Recognition", Prentice-Hall, Englewood Cliff, New Jersey, 1993

[8] A. Mudry, "Speaker Identification Using the Wavelet Transform", M. Eng. Thesis, Carleton University, 1997

[9] A. V. Oppenheim, R. W. Schafer, "Digital Signal Processing", Englwood Cliff, New Jersey, Printice Hall, 1975

[10] J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant, J. M. Tribolet, "Speech coding", IEEE Trans. Communications, pp. 710-737, April 1979

[11] D. E. Baker, "Noise: The Invisible Hazard", University of Missouri, http://muextension.missouri.edu/explore/agguides/agengin/g01962.htm

[12] X. Yang, K. Wang, S. A. Shamma, "Auditory Representations of Acoustic Signals", IEEE Trans. Information Theory, pp. 824-839, March 1992

[13] K. H. Davis, R. Biddulph, S. Balashek, "Automatic recognition of spoken digits", J. Acoust. Soc. Of America, 24:637-642, 1952 Cited in B.H. Juang, T. Cheng, "The Past, Present, and Future of Speech Processing", *IEEE Signal Processing Magazine*, 1998

[14] B.H. Juang, T. Cheng, "The Past, Present, and Future of Speech Processing", IEEE Signal Processing Magazine, 1998

[15] A. Syrdal, R. Bennet, S. Greenspan, "Applied Speech Technology", CRC Press, Inc., Boca Raton, Florida, USA, 1995

[16] S. Theodoridis, K. Koutroumbas, "Pattern Recognition", Academic Press, San Diego, California, USA, 1999

[17] A. Waibel, K. F. Lee, "Readings in Speech Recognition", Morgan Kaufmann Publishers, Inc. San Mateo, California, USA, 1990

[18] Myers, C.; Rabiner, L.; Rosenberg, A., "An investigation of the use of dynamic time warping for word spotting and connected speech recognition", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80. , Volume: 5, Apr 1980, pp. 173 - 177

[19] Widrow, B.; Lehr, M.A., "30 years of adaptive neural networks: perceptron, Madaline, and backpropagation", *Proc. IEEE* , Volume: 78 , Issue: 9 , Sept. 1990, pp.1415 - 1442

[20] J. Byorick, R. P. Ramachandran, R. Polikar, "Isolated Vowel Recognition Using Linear Predicitive Features and Neural Networks Classifier Fusion", *Proc. 5th ISIF/IEEE Int. Conf. on Information Fusion*, vol. 2, pp. 1565-1572, Annapolis, MD, 8 – 11 July 2002

[21] M. A. Abidi, R. C. Gonzalez, "Data Fusion in Robotics and Machine Intelligence", Academic Press, 1992, cited in Y. H. Hu, J. Hwang, "Handbook of Neural Network Signal Processing", CRC Press Inc., Boca Raton Florida, USA, 2002

[22] R. P. Lippmann, "An Introduction to Computing with Neural Networks", *IEEE ASSP Magazine*, April 1987, pp. 4-22

[23]Y. H. Hu, J. Hwang, "Handbook of Neural Network Signal Processing", CRC Press Inc., Boca Raton Florida, USA, 2002

[24] S. Lawrence, C. Lee Giles, A. Chung Tso, "What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation, Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, June 1996

[25]A. Waibel, K. F. Lee, "Readings in Speech Recognition", Morgan Kaufmann Publishers, Inc. San Mateo, California, USA, 1990, pp. 263

[26] Rabiner, L.R.," A tutorial on hidden Markov models and selected applications in speech recognition", Proc. IEEE , Volume: 77 , Issue: 2 , Feb. 1989
pp. :257 - 286

[27] S. E. Levinson, L. R. Rabiner, M. M. Sondhi, "Speaker Independent Isolated Digit Recognition Using Hidden Markov Models", Proc IEEE ICASSP 83, Boston, pp. 1049-1052

[28] Garofolo, J.S., Lamel, L. F., et al., "DARPA TIMIT Acoustic-phonetic Continuous Speech Corpus", U.S. Department of Commerce, 1993

[29]Babak Azimi Sadjadi, Vahid Tabatabaee, Seyed Bahram Zahir Azami, Caro Lucas, "Speaker Independent Speech Recognition Using Hidden Markov Model for Persian Isolated Words", Esteghlal Journal of Engineering, no. 13, 1993

[30] Heikki N. Koivo, "NEURAL NETWORKS: Basics using MATLAB Neural Network Toolbox", Helsinki University of Technology, 2005

[31] P. Boersma, D. Weenink, "PRAAT, a system for doing phonetics by computer, version 3.4", Institute of Phonetic Sciences of the University of Amsterdam, Report www.praat.org

[32] L. Mayfield Tomokiyo, "Recognizing Non-Native Speech: Characterizing and Adapting to Non-Native Usage in LVCSR", Ph.D. dissertation, Carnegie Melon University, 2001

[33] C. Huang, T. Chen, E. Chang, "Accent Issues in Large Vocabulary Continuous Speech Recognition", Technical Report, Microsoft Research Asia, 5F, Sigma Center, No. 49, Zhinchun road, Beijing China, 2002

[34] S. Goronzy, "Robust Adaptation to Non-Native Accent in Automatic Speech Recognition", Springer Verlag, Berlin, 1998

[35] M. W. Feng, "Speaker Adaptation Based on Spectral Normalization and Dynamic HMM Parameter Adaptation", *Proc ICASSP95*, USA pp. 407-410

[36] D. Pye, P. C. Woodland, "Experiments in Speaker Normalization and Adaptation for Large Vocabulary Speech Recognition", *Proc ICASSP 97*, Munich, Germany, [ICA97], pp. 1047-1050

[37] L. Neumeyer, A. Sankar, V. Digalakis, "A Comparative Study of Speaker Adaptation Techniques", *Proc Eurospeech95*, Madrid, Spain, pp. 1127-1130

[38] C. Teixeira , I. Trancoso and A. Serralheiro, " Accent Identification" in *Proc ICSLP'96*, vol.3, pp. 1784-1787

[39] J. H. L. Hansen, L. M. Arslan, "Foreign Accent Classification Using Source Generator Based Prosodic Features", *Proc ICASSP-95*, USA, pp. 836 - 839

[40] "Merriam-Webster's Collegiate Dictionary": Merriam-Webster Inc., 2003

[41] J. H. L. Hansen, U. Yapanel, R. Huang, A. Ikeno, "Dialect Analysis and Modeling for Automatic Classification", *Proc Interspeech 2004*, Jeju Island, South Korea, 2004

[42] D. C. Tanner, M. E. Tanner, "Forensic Aspects of Speech Patters, Voice Prints, Speaker Profiling, lie and Intoxication Detection", Lawyers And Judges Publishing Company, 2004

[43] M. Barakat, J. Ohala, F. Pellegrino, "Prosody as a distinctive feature for the discrimination of Arabic dialects", *Proc Eurospeech 99,* pp. 395-398, Budapest, Hungry, 1999

[44] J. L. Rouan, J. Farinas, F. Pellegrino, "Automatic modeling of rhythm and intonation for language identification", *Proc International Congress on Phonetic Sciences (15th ICPhS)*, pp. 567-570, 2003

[45] Q. Yan, S. Vaseghi, "A comparative analysis of UK and US English accents in recognition and synthesis", Proc IEEE ICASSP2002, vol. 1, Florida, USA pp. 413-416, 200

[46] P. J. Chesquiere, D. Van Compernolle, "Flemish Accent Identification Based on Formant and Duration Features", *Proc IEEE International Conference on* , Volume: 1 , 13-17 May 2002, pp. I-749 - I-752 vol.1, 2002

[47] S. Greenberg, "On the Origins of Speech Intelligibility in the Real World", ESCA workshop on Robust Speech Recognition for Unknown Communication Channels, pp. 23-32, France 1997

[48] M. Tanabian, R. Goubran, "Speech Accent Identification with Vocal Tract Variation Trajectory Tracking Using Neural Networks ", *Proc IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety,* Orlando, FL, USA, 31 March – 1 April 2005, pp. 130-134

[49] Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, Neural Networks", 6 (4), 1993, pp.525-533