AUTOMATIC SPEECH RECOGNITION SYSTEM FOR TURKISH SPOKEN LANGUAGE

DOĞAN DALVA

IŞIK UNIVERSITY

2012

AUTOMATIC SPEECH RECOGNITION SYSTEM FOR TURKISH SPOKEN LANGUAGE

DOĞAN DALVA
B.S., Electronics Engineering, Işık University, 2009

Submitted to the Graduate School of Işık University
In partial fulfillment of the requirements for the degree of
Master of Science
In
Electronics Engineering

IŞIK UNIVERSITY
2012

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

AUTOMATIC SPEECH RECOGNITION SYSTEM FOR TURKISH SPOKEN LANGUAGE

DOĞAN DALVA

APPROVED BY:

Assist. Prof. Dr. Ümit Güz Thesis Supervisor	Işık University	_____
Prof. Dr. Sabri Arık	Işık University	_____
Prof. Dr. Serdar Özoğuz	Istanbul Technical University	_____

APPROVAL DATE: 21.06.2012

AUTOMATIC SPEECH RECOGNITION SYSTEM FOR TURKISH SPOKEN LANGUAGE

Abstract

The transmission and storage of speech sounds is possible for decades. In addition by using signal processing techniques, it is also possible to process speech signals. By using time and frequency analysis of speech signal and several machine learning algorithms, it is possible to build a system which is used to recognize spoken words. Such systems are called Automatic Speech Recognition systems.

In our work, we have used the Automatic Speech Recognition system for Turkish spoken language which has built by BUSIM speech group. However, the output of the recognizer is the list of spoken words. Even for humans it is a very hard task to understand a text without punctuation symbols. Hence to build more complex recognizer whose goal to perform topic segmentation and topic summarization, the output of ASR should be divided into sentences at first.

Our goal is to build a system which performs the sentence segmentation. In our work we have used ASR system to obtain word level and phoneme level time marks and by using that time marks with the audio files, we have extracted prosodic features, where the prosodic properties of speech contains information about the punctuation in the text, which is not available at the output of ASR system.

TÜRKÇE DİLİ İÇİN OTOMATİK KONUŞMA TANIMA SİSTEMİ

Özet

Uzun yıllardan beri ses ve konuşmaların saklanması ve iletilmesi mümkündür. Ayrık zamanlı ve sürekli zamanlı işaret işleme yöntemleri sayesinde ses ve konuşma işaretleri de işlenebilmektedir. Bununla beraber, eğitilebilen algoritmalar kullanılarak Otomatik Konuşma Tanıma ve Otomatik Konuşmacı tanıma sistemleri de geliştirilebilmektedir.

Bu çalışmada Boğaziçi Üniversitesi'nde bulunan "BUSİM speech group" tarafından geliştirilmiş, Türkçe dili için otomatik konuşma tanıma sistemi kullanılmıştır. Bu sistem; konuşmacıların söylediği kelimeleri bir liste halinde dökebilmektedir. Ancak; bir insan için bile noktalama işaretlerinden yoksun bir metinden bilgi alabilmek oldukça zordur. Bu sebepten dolayı konu bölütleme veya konu özetleme gibi daha ileri uygulamaları yapabilmek için, öncelikle cümle bölütleme işleminin yapılması gerekmektedir.

Dil bilgisine uygun bir yazılı metindeki noktalama işaretleri, diksiyonda vurgu ile belirtilmektedir. Başka bir deyişle bu özellikler konuşma işaretinin bürünsel özellikleridir. Amacımız, Otomatik Konuşma Sisteminin çıktıları ile ses işaretinin bürünsel özelliklerini kullanarak cümle bölütlemesini otomatik yapabilen bir sistem geliştirmektir.

Acknowledgements

There are many people who helped to make my years at the graduate school most valuable. First, I thank my supervisor, Assist. Prof. Dr. Ümit Güz and my co-supervisor, Assist. Prof. Dr. Hakan Gürkan. Having the opportunity to work with them in such project was intellectually rewarding and fulfilling.

I also thank Assist. Prof. Dr. Murat Saraçlar; who contributed much to the development of this research starting from the early stages of my work. Many thanks to members of BUSIM Speech Group at Boğazici University who patiently answered to my questions and helped me to exceed problems.

The last words of thanks go to my family. I thank my parents, my mother Leya Dalva, my father Salvator Dalva and my grandmother Fortüne Dalva, for their patience and encouragement. Lastly I thank my beloved Ece Öztürk for her endless support through this long journey.

To my family

Table of Contents

List of Tables

List of Figures

List of Symbols

ASR	Automatic Speech Recognizer
IWR	Isolated Word Recognition
CSR	Continuous Speech Recognition
DTW	Dynamic Time Warping
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
IPA	International Phonetic Alphabet
ARPA	Advanced Research Project Agency
IHC	Inner Hair Cells
IL	Intensity Level
LL	Loudness Level
SPL	Sound Pressure Level

DTFT	Discrete Time Fourier Transform
DFT	Discrete Fourier Transform
STFT	Short Time Fourier Transform
MFCC	Mel Frequency Cepstral Coefficients
LPC	Linear Predictive Coding
OS	One Stage Algorithm
VOA	Voice of America
STM	Segment Time Marks
LB	Level Building Algorithm

Chapter 1

Introduction

Since the beginning of human civilization, speech was the most widely used communication tool by humans. However by the development of the civilization and technology humans started to use mails to communicate, furthermore by the development of the technology telegraph became most widely used communication system.

On the other hand scientists were studying how to capture sound and speech waveforms. In the late 1850's Frenchman Édouard-Léon Scott de Martinville has invented the phonautograph [1], which is the earliest known device for printing the sound waves onto a roll of paper. In the following years phonograph by Thomas Edison and gramophone by Emile Berliner has developed and by the use of those machines the printed waveforms of phonautograph has recovered. In early 1900's electrical and magnetic sound recording systems has also developed by the scientists. In the following years by the invention of telephone by Alexander Graham Bell, the audio signal started to be transmitted.

Today, speech signals are converted to an electrical signal by using microphone, and it may processed or stored by using digital signal processing methods then reconstructed and heard by using headphones or loudspeakers. In addition the facility of storing and transmitting speech signals, the speech recognition applications can be performed.

There are several examples to Automatic Speech Recognition (ASR) systems such as Isolated Word Recognition (IWR) systems and Continuous Speech Recognition (CSR) systems. An example of a basic ASR system is Isolated Digit Recognizer, which recognizes

the utterance of digits spoken by the speaker. On the other hand a complex ASR system recognizes the words in a spoken language. However the list of recognized word utterance, which is the output of the ASR system, is not enough to understand the meaning of the spoken message even for the humans. Hence by using prosodic features, we have developed a system which can be trained and used to detect sentence boundaries. The sentence boundary detection is the first step of the further studies such as topic segmentation and topic summarization.

1.1 The Overview of the Thesis

At the beginning of Chapter 2, the speech chain and the anatomical structure and models of human sound production and sound perception of human has discussed. In Chapter 3, the theoretical background of speech signal processing is given. In Chapter 4 the speech recognition problem, approaches of ASR, complexity of the ASR system, building a speech recognition system and performance evaluation of speech recognizers has introduced. In Chapter 5, the speech recognition system modeling by using Dynamic Time Warping (DTW) and Hidden Markov Modeling (HMM) has introduced and also the Hidden Markov Model Toolkit (HTK) has described with applications used in this thesis. In Chapter 6, the definition of prosody, the reason of using prosodic features, the prosodic features and Mary Harper's prosodic feature extraction tool on *Praat* has described. In Chapter 7, usage of a supervised machine learning algorithm *icsiboost* which is based on *AdaBoost* has explained in order to train a model which detects sentence boundaries. Finally Chapter 8 includes the tests and conclusion of the thesis.

Chapter 2

Communication of Human

2.1 The Speech Chain

Speech is used to communicate information from a speaker to a listener. There are several levels which are related with the steps of speech production, speech perception and speech chain such as linguistic level, physiological level and acoustic level which is shown at Figure 2.1 [2].In addition Figure 2.2 [2] shows the block diagram representation of the speech chain.

At the beginning of the whole process, a message or an idea has formed on speaker's brain. Initially we can represent that message as text, where text is the combination of letters, i.e. text symbols, where an example shown at Figure 2.3 for Turkish and English spoken languages with phonemes. This step is shown at Figure 2.2 [2] at the left block of the discrete input block.

In the following step, the text symbols should be converted into a symbolic representation of the sequence of sounds corresponding to the spoken version of the message, i.e. phonetic symbols. This step is related with linguistic level of speech chain.

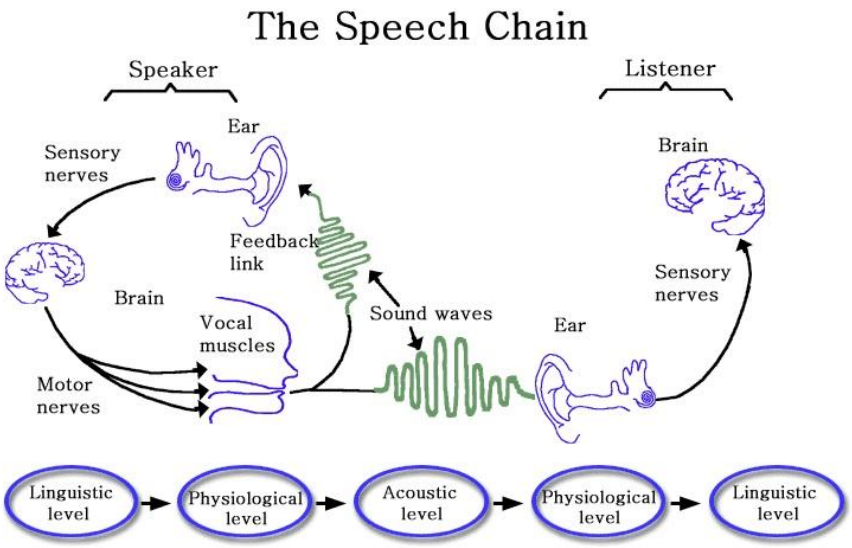


Figure 2.1 The speech chain

The text symbols are represented by alphabet. There are many kinds of widely used modern alphabets such as Latin, Greek, Hebrew and Arabic alphabets. Either in Turkish or in English languages, Latin alphabet is used. On the other hand, the phonetic symbols are represented by International Phonetic Alphabet (IPA), which is defined as an alphabetic system of phonetic notation and based preliminary on Latin alphabet and used mostly in linguistics and phonetics. In addition those phonetic symbols are labeled using computer-keyboard-friendly code called ARPAbet (Advanced Research Project Agency), which does not require special fonts and is thus more convenient for computer applications, so that is used by mostly engineers and software developers.

In spite of there is difference between alphabet and ARPAbet for English language, i.e. there are difference between phoneme and letter transcriptions of a word. However the text symbols and phonetic symbols are same at Turkish language. In Turkish, text symbols are equivalent to the phonetic symbols [3]. Figure 2.3 shows an example which denotes the spoken and written words in Turkish and English [32] languages.

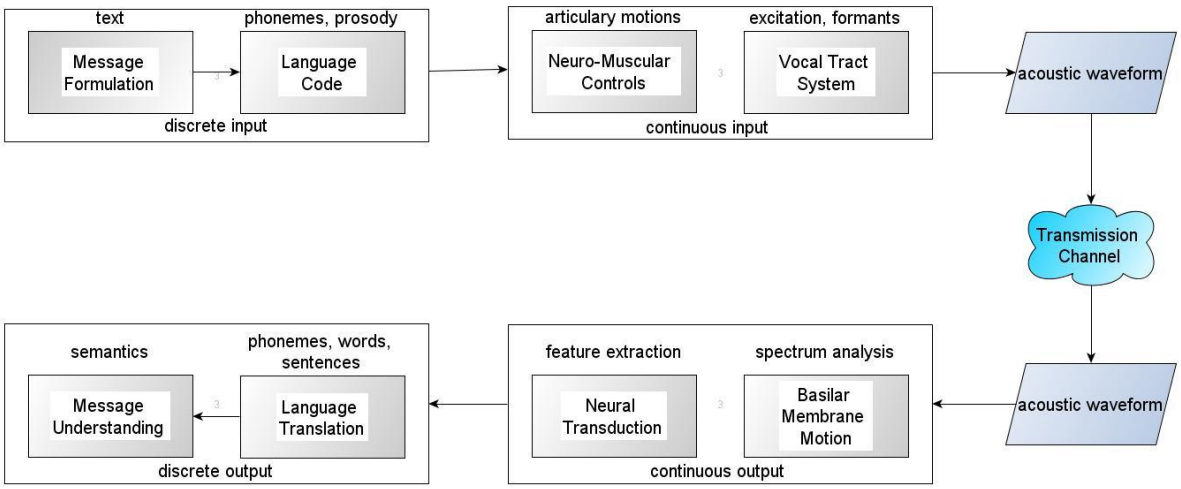


Figure 2.2 Block diagram representation of the speech chain

At the next step speaker uses his/her vocal tract system to generate the speech. This step is called physiological level of the speech chain, which is shown at Figure 2.2 first and second blocks of continuous input part. The speaker uses his/her neuro-muscular controls which are the combination of the tongue, lips, teeth, jaw and velum and articulatory motions to obtain sound of the desired spoken message. And finally sound waves i.e. acoustic waveforms are propagate through the transmission channel which is called the acoustic level, which has shown at Figure 2.2.

Afterwards the sound wave i.e. acoustic wave reaches to the both listeners and speakers ears. The continuous output step in Figure 2.2 represents the Physiological level of listener side in Figure 2.1. Furthermore, the speaker also captures his/her own speech as feedback. Hence we can assume that in this step speaker is also becomes a listener while hearing his/her own speech. The spectrum analysis and feature extraction processes starts at the listener side. In this step, the acoustic wave is converted to electrical signals and those electrical signals are delivered to brain by neurons.

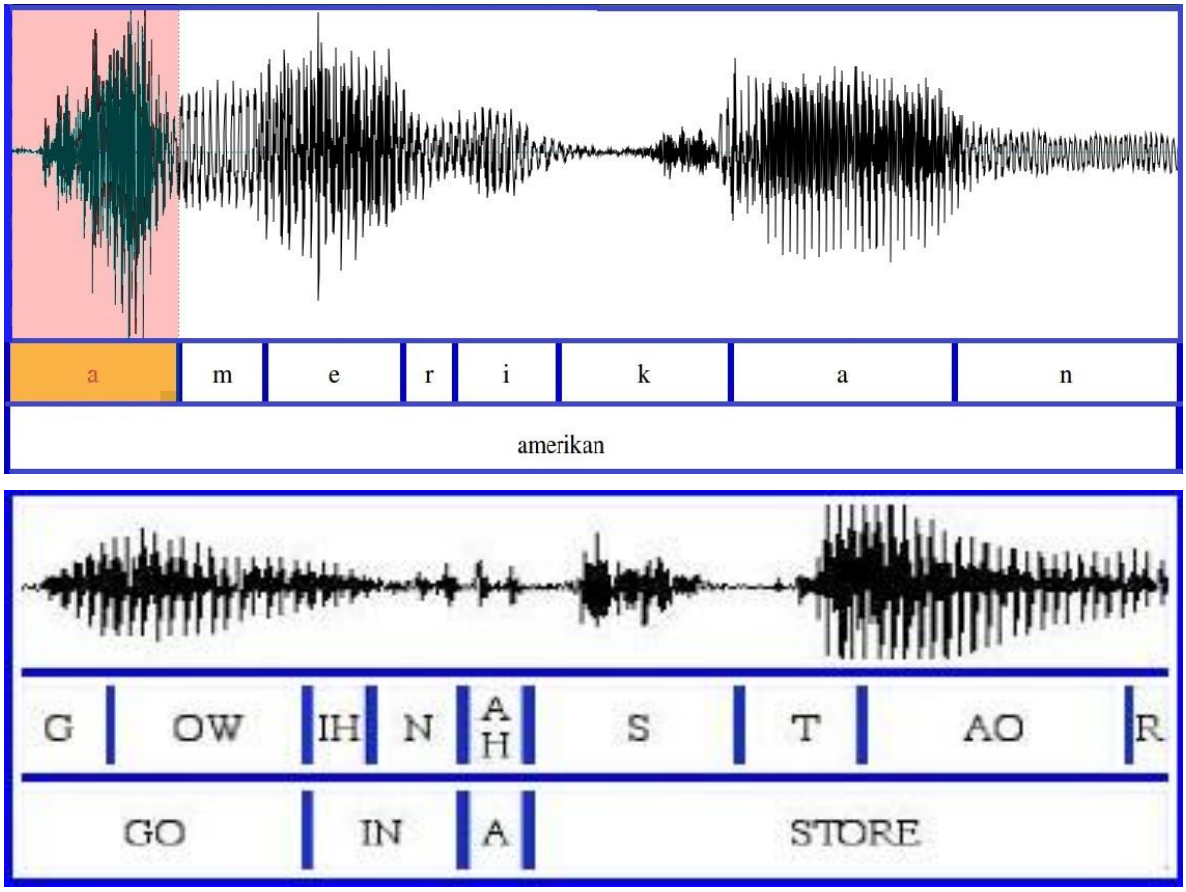


Figure 2.3 Turkish and English spoken and written words for given examples. The upper side of the figure shows Turkish and the down side shows English alphabet and ARPAbet representation of the given words

Finally, at the discrete output step which is shown at Figure 2.2 and represents the linguistic level at Figure 2.1, the electrical signal which is delivered to the brain by neurons, is analyzed in order to extract phonemes, words and sentences, and then the listener understands the message by his/her semantics knowledge.

In the following chapters, the steps of the listener side such as spectrum analysis, feature extraction, language translation processes will be discussed more detailed with computer applications but first in this chapter we will focus on human speech production and perception.

2.2 The Process of Human Speech Production

In this section and following section, the physiological level of speech chain either speaker or listener side will be explained briefly. In this section we will focus on human vocal tract system. Figure 2.4 shows the human vocal tract system schematically [4]. The human vocal tract and human speech production explained below [2].

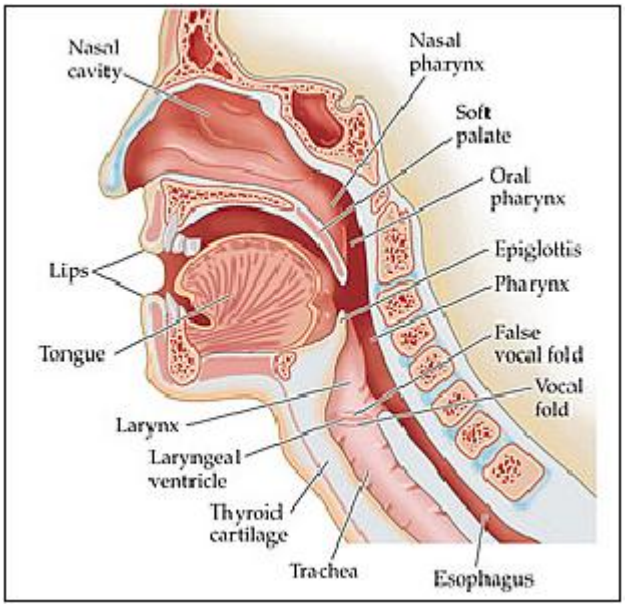


Figure 2.4 Human vocal tract system

The human vocal tract system starts at vocal cords, or glottis, which takes place under Esophagus in Figure 2.4 and ends at the lips. The average length of male vocal tract is approximately 17-17.5 cm and the cross sectional area of the vocal tract is 20 cm^2. The region between nostrils, i.e. nose holes and velum, where velum is near nasal pharynx at Figure 2.4 is defined as nasal tract.

The procedure of speech production starts with the entrance of air to the lungs which is known as normal breathing. Generally there is neither speech nor sound produced at this step. The vocal cords are tensed in order to speak or produce sounds.

While the air inside the lungs is expelled, the vocal cords vibrate by Bernoulli-Law variations of air pressure in the glottis. The glottis converts the air into quasi-periodic pulses by opening and closing. The quasi-periodic pulses are frequency-shaped while passing through the Pharynx, mouth cavity and nasal cavity. Finally the sound is determined by the positions of the jaw, tongue, velum, lips and mouth.

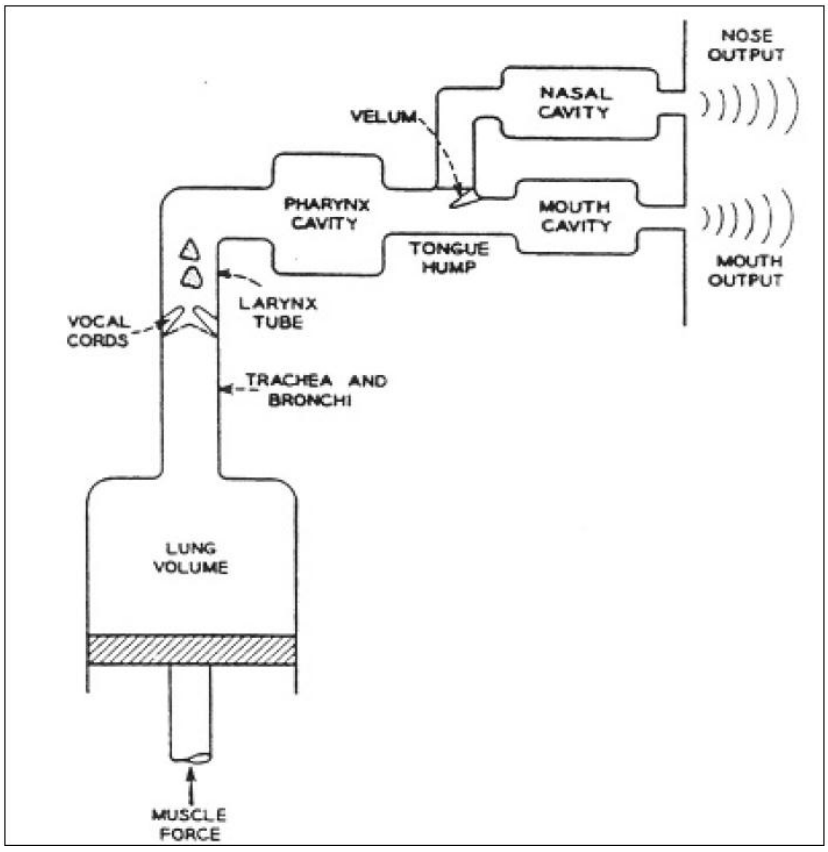


Figure 2.5a Schematic view of overall speech production mechanism

Figure 2.5a [5-6] shows a schematic model of speech production mechanism of human. As described above, at first the air is transmitted into the trachea and vocal cords by muscle force. If the vocal cords are tensed, quasi-periodic or voiced speech sounds are obtained such as /a/, /e/, etc. In the case of relaxed vocal cords, the air flow hits a constriction in the vocal tract. If the constriction is partial, then unvoiced sounds such as initial sounds in the corresponding words /show/, /say/ is produced. If the constriction is full, there will be no voice until the pressure on constriction suddenly released.

In that case a brief transient sounds such as initial sounds at the beginning of words /cat/, /talk/, and /pressure/ are obtained [2]. Figure 2.5b shows another schematic diagram of human speech production mechanism. [2-7].

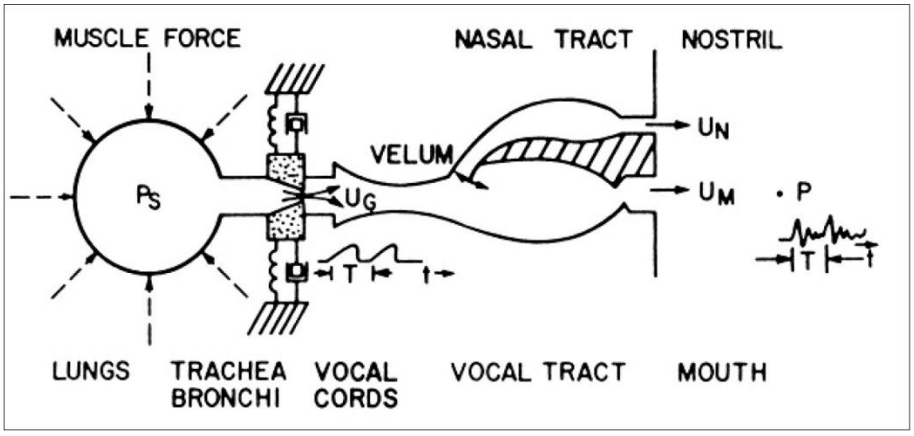


Figure 2.5b Schematic view of overall speech production mechanism

2.3 The Process of Human Hearing and Perception the Sound

In the previous section, the anatomy and basic models of speech production mechanism of human, which is the physiological level of speech chain in speaker side. Below, the physiological level of listener side, i.e. the human perception of sound is described. In addition it is well known that speaker also becomes listener while he/she hears his/her speech as feedback.

At the following, the anatomy of ear will be described first afterwards the range of human hearing will be given and finally the perception of sound will be discussed.

2.3.1 Anatomy and Function of the Ear

The whole hearing process i.e. the function of the ear is called as auditory system. The input of human auditory system is the acoustic signals which have physical intensity and frequency attributes, and the output is perceived sound, where those are called

psychophysical observations. The intensity of the acoustic signal is perceived as loudness and the frequency of the signal is perceived as frequency. The auditory system divides into three parts such as; acoustic to neural conversion, neural transduction and neural processing.

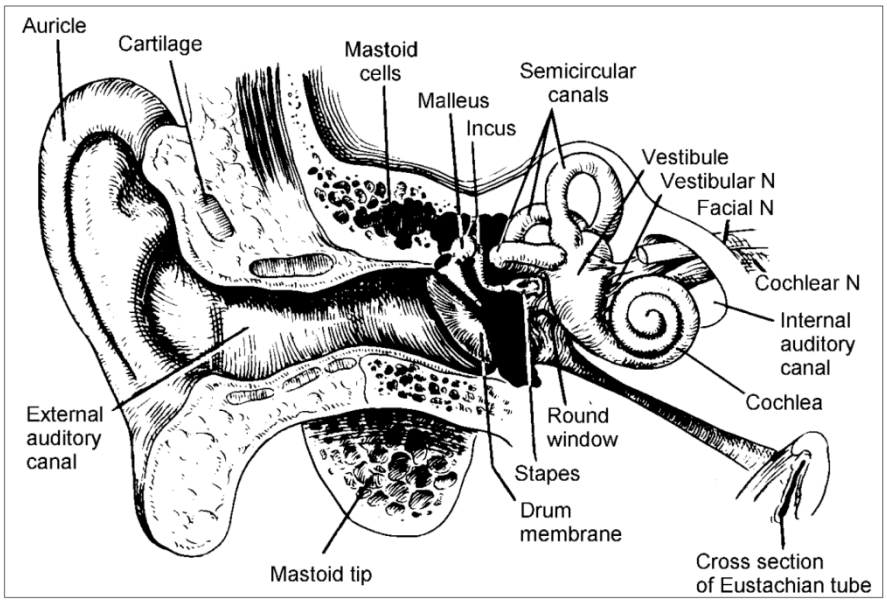


Figure 2.6 Schematic diagram of human ear. Hallowell and Silverman et. al. 1970.

The overview of whole process is, at first the acoustic energy of acoustic signals effects drum membrane and drum membrane makes periodic movements. Those periodic movements are delivered to inner ear via middle ear. The inner ear converts those periodic movements into neural signals and neural signals are delivered to brain and finally brain processes those neural signals [8].

The human ear, shown at Figure 2.6 [9-10], divides into three parts such as outer ear, middle ear and inner ear. The outer ear starts at auricle of the ear and ends at drum membrane. The middle ear is starts at the drum membrane and ends at the end of Eustachian tube. Finally the inner ear is formed by semicircular canals and cochlea.

The Auricle picks as much as possible acoustic energy and delivers into external auditory canal. The length of auditory canal is approximately 3 cm and it delivers captured acoustic energy to the drum membrane. The outer side of drum membrane conducts with air and inner side of drum membrane conducts to the bones of middle ear. Because of the structure, the drum membrane cannot deliver the vibrations in the air linearly.

The middle ear is formed by three bones which are called malleus, incus and stapes, and Eustachian tube. The Eustachian tube expels the exudates inside middle ear and balances air pressure between inner side and outer side of drum membrane. The role of bones is to compensate the pressure of vibrations delivered to inner ear and protects nerve cells of inner ears.

The inner ear is formed by semicircular canals, which balances human body, and the cochlea, which converts sound pressure signals from the outer ear into electrical impulses. The cochlea is the most important organ in sound perception. The cochlea has snail-like structure and the length of the snail structure is approximately 3 cm if it is opened through a line. The periodic movements which are delivered by outer ear, reaches to the cochlea. The basilar membrane which is a part of cochlea includes 30.000 inner hair cells. The basilar membrane analyses the captured periodic movements spectrally, in a non-uniform frequency scale. Finally inner hair cells convert mechanical vibrations into neural signals.

The frequency of captured sound is inverse proportional to wavelength of the captured sound, and the sound reaches to a sequence of inner hair cells in a range where the length and location of the range is related with wavelength of the sound. Hence the frequency information of a sound is coded as spatial position of induced neurons of basilar membrane as shown at Figure 2.7 [3, 8, 11].

2.3.2 The Perception of Sound

In the previous section, the human ear i.e. human auditory system has described. In this section we will focus on perception of the sound by humans.

Initially we will describe the basilar membrane mechanics. Then we will define bark scale, which is used to model captured frequencies of sound. Then we will define the ranges of human hearing in terms of the intensity and frequency of the sound, and finally we will define the loudness level in terms of phones and pitch.

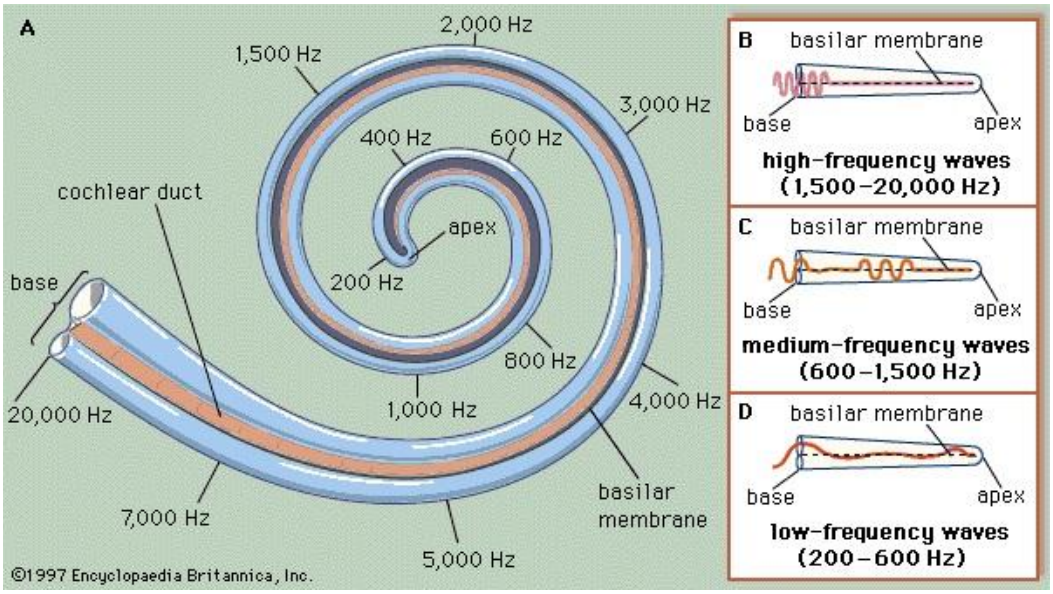


Figure 2.7 Frequency perception of basilar membrane. “A” shows the cochlea. “B, C and D” shows the perceived sounds spatially according to their frequencies

2.3.2.1 Basilar Membrane Mechanics

The sound waves propagate through air forces the drum membrane to vibrate coherently with its own frequencies. The coherently vibrations are transmitted to perilymph liquid and it vibrates a certain region of basilar membrane [8]. There exists inner hair cells (IHC) connected along the whole basilar membrane inside the cochlea. The structure of basilar membrane is analyzed at [12-15] and summarized at [2]. According to that studies the following important findings about basilar membrane is given below.

At first, the IHCs are distributed along the basilar membrane. IHCs are motion-to-neural converters, i.e. they convert the captured motion on corresponding location of basilar membrane to neural signals, where different frequencies excites different points alone the basilar membrane.

Each of the IHCs is connected to their corresponding 10 nerve fibers. The thicknesses of those 10 fibers are different. High motions are transmitted via thinner fibers and lower motions are transmitted via thicker fibers. The mechanical realization of basilar membranes is thought as non-uniform filter banks, i.e. cochlear filters. Each filter in the bank of filters has their own frequencies and bandwidths. The relation of the center frequency and bandwidth is that, while the center frequency is rising, the bandwidth of corresponding filter rises up exponentially. More details about filter bank representation and the range of human hearing is given below.

2.3.2.2 Critical Bands and Bark Scale

Figure 2.8 shows an idealized version of an equivalent basilar membrane filter bank. There are ideal band pass filters are shown, where each line represents the borders of each filter bank. A healthy human ear can percept the frequencies between approximately 15 Hz to 20 kHz in general and that frequency range is divided into 25 idealized critical band-pass filters.

The relation between the center frequency of each ideal band pass filters and bandwidth of corresponding ideal band pass filters are shown at Equation 2.1 [2]. Where F_c represents the center frequency of an ideal band pass filter and ΔF_c represents the bandwidth of corresponding ideal band pass filter.

$$\Delta F_c = 25 + 75\left[1 + 1.4\left(\frac{F_c}{1000}\right)^2\right]^{0.69} \tag{2.1}$$

The band pass filters according to the critical band theory of hearing are defined as ideal filters at above. However in reality the filters are not ideal. Overlaps between the filters shown at Figure 2.8 will occur in reality. Similarly, the IHC groups in basilar membrane according to a selected center frequency cannot vibrate alone. The adjacent IHCs will also vibrate with them [2]. A good example about the movements of IHCs is given at [8]. The IHCs are thought as grains in a grain farm, and the center frequency F_c of acoustic sound is thought as a small wind which affects only a certain area of the farm. The grains which are at the center of the affected area will move very fast, while going through the borders of the selected area, the movements of grains will be slower. However the grains which are near the borders of the area will also move slowly.

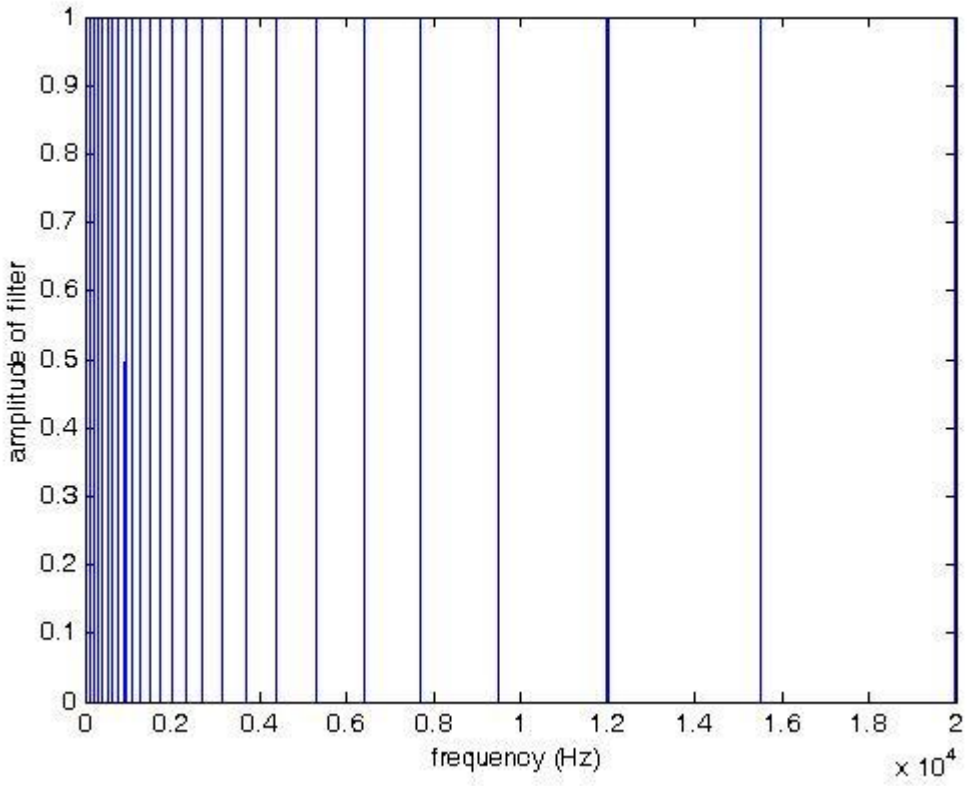


Figure 2.8 Schematic representation of band pass filters according to the critical band theory of hearing

Recall that the IHCs are distributed along the basilar membrane and each of the IHCs are connected to 10 fibers with different thickness. Hence the center frequency of a captured sound is detected the location of moving IHCs and location of the IHCs which has the fastest movement.

Previously we have also mentioned that the perceivable frequency ranges by human has divided into 25 idealized critical band-pass filters which are shown at Figure 2.8. The Bark Scale is used to index those idealized critical band-pass filters such as, the lowest center frequency and corresponding band pass filter is indexed with a value z=1, and the other filters are indexed as increasing the value of "z" while going through to higher frequencies. Table 2.1 shows the Bark Scale. The neural outputs of cochlea can be modeled mathematically. The Bark Scale is often used to transform uniform frequency analysis using FFT into Bark Scale spectrum in such models. [2, 17-20]

2.3.3.3 The Range of Human Hearing

Sound is defined as variations of air pressure that propagate as waves through air or media although; hearing is the ability to sense and process the sound [2]. However inherently like all systems, human sound perception system has specific limits. In other words humans cannot percept all of regular air pressure vibrations as sound. The range of sounds that can be percept by humans is defined by two criterions such as the frequency of the sound and the intensity of the sound. The intensity of the sound will be defined later however we can define the limits of hearing in terms of frequency and intensity. As mentioned before and shown at Figure 2.9 [45], humans can percept sounds which has frequency between 15 Hz to 20 kHz in general. The sounds which as frequency under 15 Hz are defined as infrasound and the sounds which as frequencies greater than 20 kHz are defined as ultrasounds. Humans cannot percept ultrasounds but some of animals for instance dogs and vespertilian bats can percept those sounds.

However to percept a sound, the sound should be not only in suitable frequency range but also it has a suitable intensity. Human can percept sounds between approximately 10^{-12} W/m^2 and 10 W/m^2. Equivalently the range is 0 dB to 120 dB, where the upper limit is defined as pain threshold.

z	F_u, F_l	F_c	z	ΔF_c	Z	F_u, F_l	F_c	z	ΔF_c
Bark	Hz	Hz	Bark	Hz	Bark	Hz	Hz	Bark	Hz
0	**0**				**12**	**1720**			
		50	**0.5**	**100**			**1850**	**12.5**	**280**
1	**100**				**13**	**2000**			
		150	**1.5**	**100**			**2150**	**13.5**	**320**
2	**200**				**14**	**2320**			
		250	**2.5**	**100**			**2500**	**14.5**	**380**
3	**300**				**15**	**2700**			
		350	**3.5**	**100**			**2900**	**15.5**	**450**
4	**400**				**16**	**3150**			
		450	**4.5**	**110**			**3400**	**16.5**	**550**
5	**510**				**17**	**3700**			
		570	**5.5**	**120**			**4000**	**17.5**	**700**
6	**630**				**18**	**4400**			
		700	**6.5**	**140**			**4800**	**18.5**	**900**
7	**770**				**19**	**5300**			
		840	**7.5**	**150**			**5800**	**19.5**	**1100**
8	**920**				**20**	**6400**			
		1000	**8.5**	**160**			**7000**	**20.5**	**1300**
9	**1080**				**21**	**7700**			
		1170	**9.5**	**190**			**8500**	**21.5**	**1800**
10	**1270**				**22**	**9500**			
		1370	**10.5**	**210**			**10500**	**22.5**	**2500**
11	**1480**				**23**	**12000**			
		1600	**11.5**	**240**			**13500**	**23.5**	**3500**
12	**1720**				**24**	**15500**			
		1850	**12.5**	**280**					

Table 2.1 Fu, Fl represents upper and lower frequencies of critical bandwidths ΔF_c centered at Fc. [2.1,15]

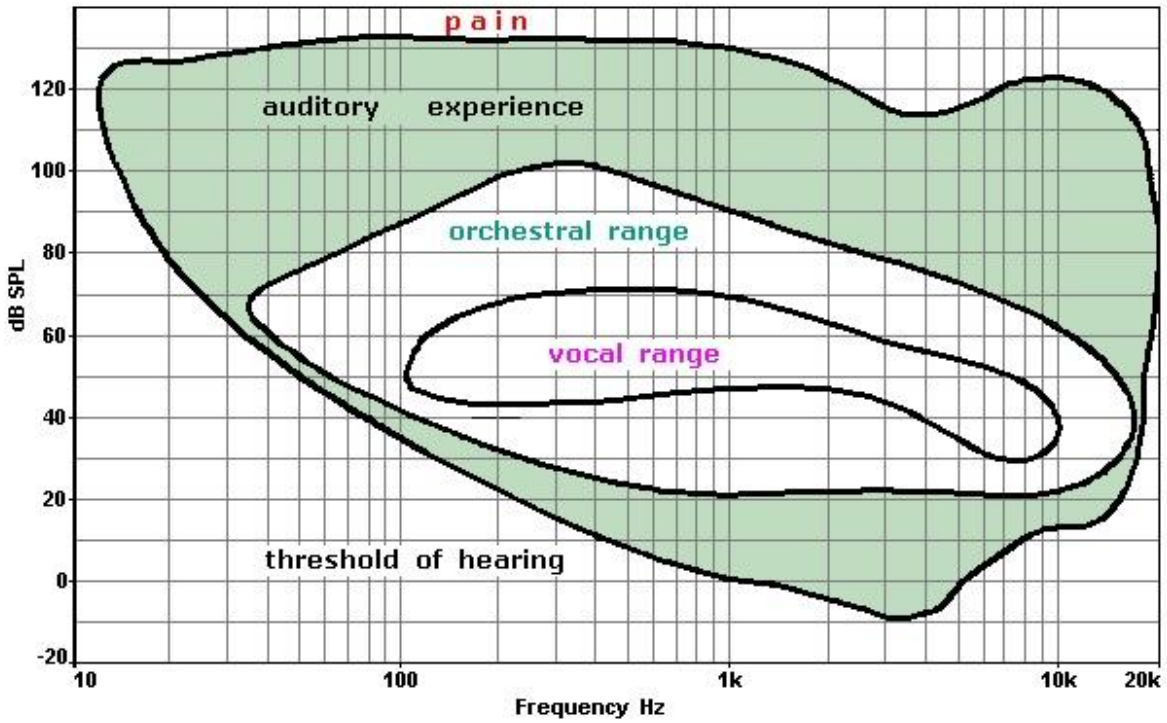


Figure 2.9 Ranges of human hearing

2.3.3.4 The Intensity and Intensity Level of Sound

The sound is combined by the superposition of varying sinusoidal signals, so sound pressure is related with pressure of sinusoidal waves propagates through the space. The sound pressure is much smaller when compared with atmospheric pressure. The intensity of sound can be measured physically as intensity level (IL) and sound pressure level (SPL). The acoustic intensity is defined as average flow of energy, i.e. power, through a unit area measured in Watts/m^2. The threshold of hearing is measured as 10^{-12} W/m^2 and defined as reference point I_0. The intensity level (IL) of a sound is measured as logarithmic ratio with respect to I_0. The mathematical representation of intensity level is shown at Equation 2.2.

$$IL = 10 \log \left(\frac{I}{I_0} \right) dB \tag{2.2}$$

The sound pressure level (SPL) is defined as logarithmic ratio of intensity P^2 of the sound which defined in terms of sinusoidal sound wave amplitude P travelling through space to

the amplitude of the hearing threshold intensity, $P_0 = 2 \times 10^{-12}\ newtons/m^2$at room temperature and standard pressure. Equation 2.3 shows the mathematical representation of SPL. Note that in general SPL is used widely [2].

$$SPL = 10 \log\left(\frac{P^2}{P_0^2}\right) = 20 \log\left(\frac{P}{P_0}\right) dB \tag{2.3}$$

The range of human hearing in terms of intensity level was mentioned while defining the ranges of human hearing. We can divide the total range into sub ranges as shown at Table 2.2 [2].

SPL (dB)	Sound Source	SPL (dB)	Sound Source
170	Pain	80	Blow Dryer
160	Jet Engine – close up	70	Noisy Restaurant
150	Artillery Fire	60	Conversation Speech
140	Rock Concert	50	Office Background Noise
130	22 Caliber Rifle	40	Quiet Conversation
120	Thunder	30	Whisper
110	Subway Train	20	Rustling Leavers
100	Power Tools	10	Breathing
90	Lawn Mower	0	Threshold of Hearing

Table 2.2 SPLs for a range of sound sources.

2.3.3.5 Loudness Level

Previously we have defined the intensity level of the sound and at Table 2.2 we have shown some examples about the intensity level of some sound sources. However the loudness level (LL) of a sound is not directly proportional with only the sound pressure level (SPL) of a sound. Suppose that we have a sound which has 1 kHz frequency and 60 dB intensity

level. If we fix the SPL and reduce the frequency of the sound slowly, as shown at Figure 2.10 with horizontal line, we will sense that the loud of the sound is descending, furthermore if we reduce the frequency to 30 Hz, we will not hear the sound.

On the other hand, we may fix the SPL of a 1 kHz sound to 60 dB as a reference sound. Then if we tune the loudness of a second sound which has a center frequency 125 Hz according to the reference sound, at the beginning we may adjust the SPL of the second sound as 60dB but we will sense that the loudness of two sounds are not equal, so we will increase the SPL of the second sound until 80 dB, which has shown at Figure 2.10 with vertical line, and we will sense that both sounds has equal loudness level [8]. Hence, we should define the loudness level not only as a function of SPL but also the frequency of the sound. The loudness level (LL) of a tone is defined as the IL or SPL of the sound at 1 kHz [2]. Therefore the loudness level (LL) is defined with *phons.* Figure 2.10 [46] shows the phon curves, where each of the phon curves are depends on both frequency (Hz) of the sound and SPL (dB) of the sound.

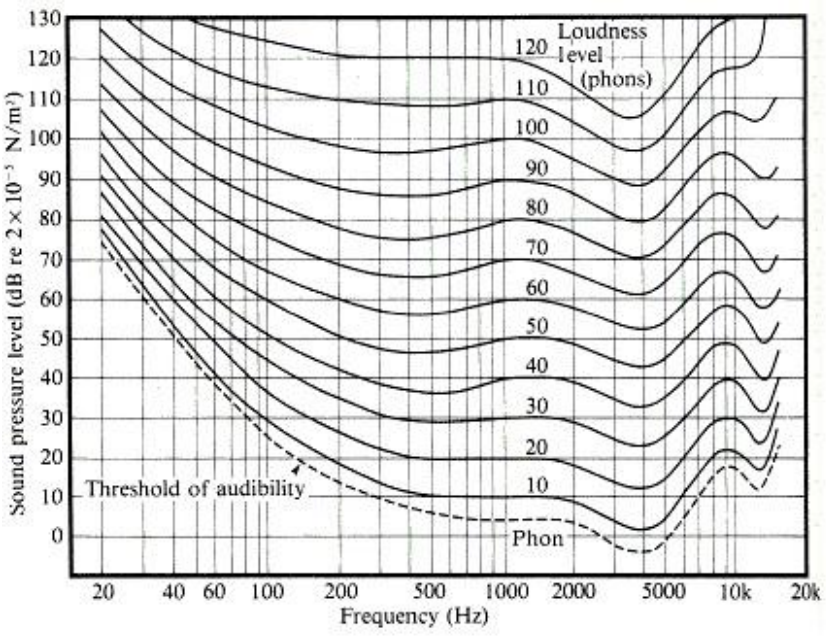


Figure 2.10 Loudness Level (LL) Curves.

2.3.3.6 Pitch

The frequency of a sound, which is an objective quantity, is percept by human as pitch of the sound and that is a subjective quantity it implies that the pitch of the sound is highly correlated with the frequency of the sound. In Figure 2.7, the distribution of IHC's and the frequency perception regions in cochlea has shown. The unit of the pitch is measured by *Mels* where the word "mel" is derived from the word "melody"[2]. The way that researchers have related the frequency of a sound with pitch of sound was [2], at the first 1000Hz is adjusted to 1000*mels* for normalization purposes, then it is asked to several respondents to adjust the frequency of a second tone until it has half the pitch of the first tone. By performing similar tasks with many respondents, the relation in Equation 2.4 obtained between pitch and frequency.

$$pitch\ (mels) = 1127 log_e(1 + Frequency(Hz)/700) \tag{2.4}$$

Chapter 3

Speech Signal Processing

3.1 Speech Properties and Speech Waveform

Speech is defined as sequence of ever-changing sounds according to the spoken language grammar rules. The speaker encodes the message by using ever-changing sounds according to the grammar rules of spoken language. Furthermore the speech signal depends on the sounds occurred before and after the current sound and state of vocal organs [2].

In previous section, we had mentioned that the text message was converted to symbolic representation of sequence of sounds, i.e. utterance. Furthermore in Figure 2.3 we have gave an example of Turkish and English utterance of words. As it mentioned in Chapter 2, the elements of utterance are called phonemes, and they are represented by ARPAbet, where table [23] shows ARPAbet for modern Turkish.

In addition, the speech signal varies slowly with time such as approximately 5-15 sounds per second [2]. If we capture approximately 20 – 40 milliseconds length speech signal, we will observe that the signal on that captured frame is quasi-periodic. For instance, Figure 3.1 shows a 16 kHz 16-bit sampled 0.5 seconds long, (8000 samples) male voice, says “see you later.” Figure 3.2 shows five 25 milliseconds long frames of that speech signal. If we consider first two frames, where frames are divided by vertical lines, we see that the signals are similar. Then if we consider second and third frames, again there is similarity between

those two signals. Furthermore if we consider fourth and fifth frames, we see the same signal. Hence short segments of the speech signal are isolated and processed as if they were short segments from a sustained sound with fixed i.e. not time varying properties. Hence this property of speech signal leads short-time processing methods [2], which will be discussed at sections 3.2 and 3.4.

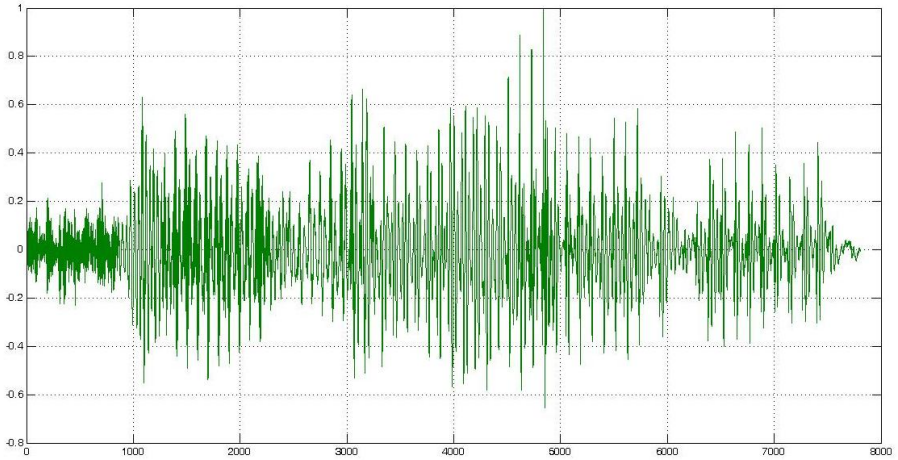


Figure 3.1 16 kHz 16-bit sampled male speech signal.

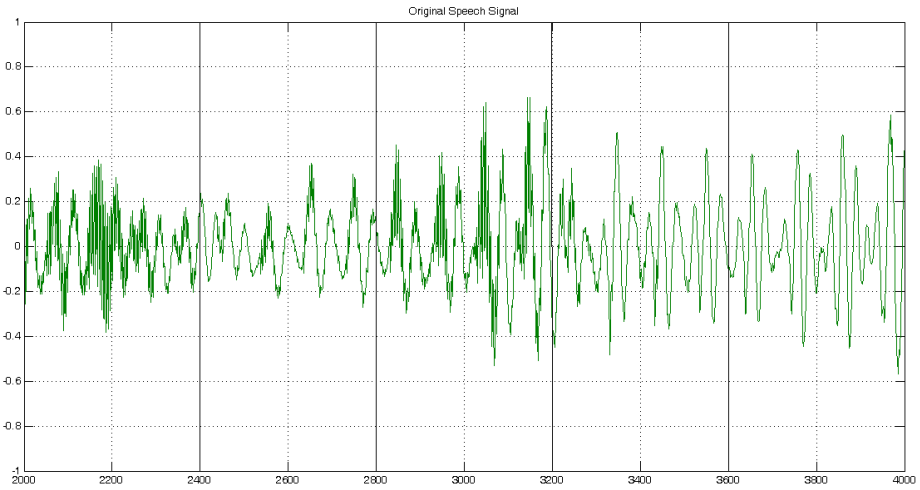


Figure 3.2 Five 25 milliseconds long cascade frames.

When we consider a speech signal in 5 seconds frame, we will see three main types of regions such as voiced, unvoiced and silence or background noise regions. The production

of voiced and unvoiced sounds was described in the previous chapter. Figure 3.3 shows three seconds of Turkish spoken words “Irakta dokuz amerikan askerinin” by a female speaker. The voiced regions are shown by the “V” sign, and the unvoiced regions are shown by “U” sign.

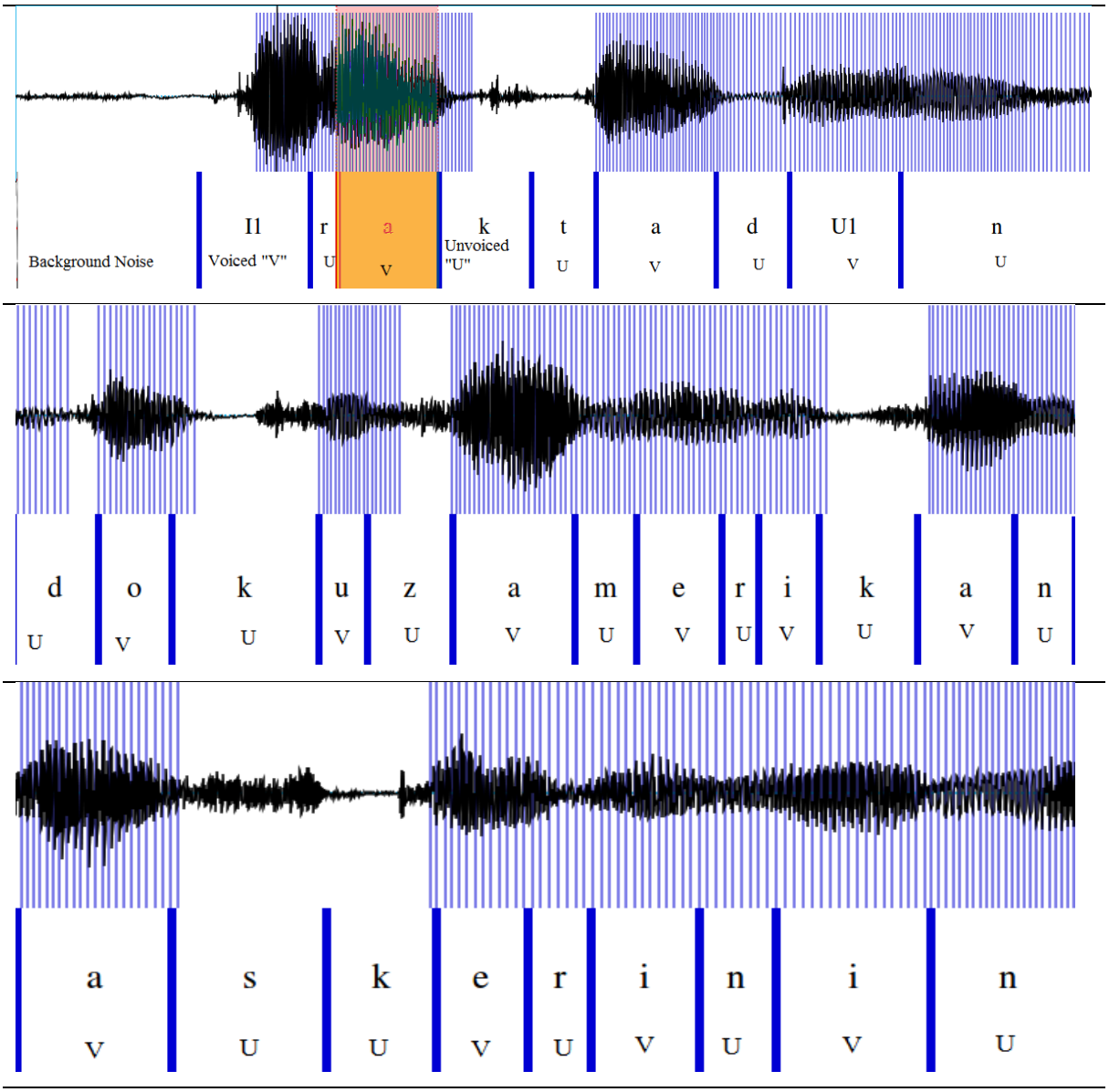


Figure 3.3 Voiced, Unvoiced and Silence/Background Noise regions of speech signal.

Additionally Figure 3.4 shows one word “ancak” of a female speaker. The voiced, unvoiced and silence regions can be seen better at this figure.

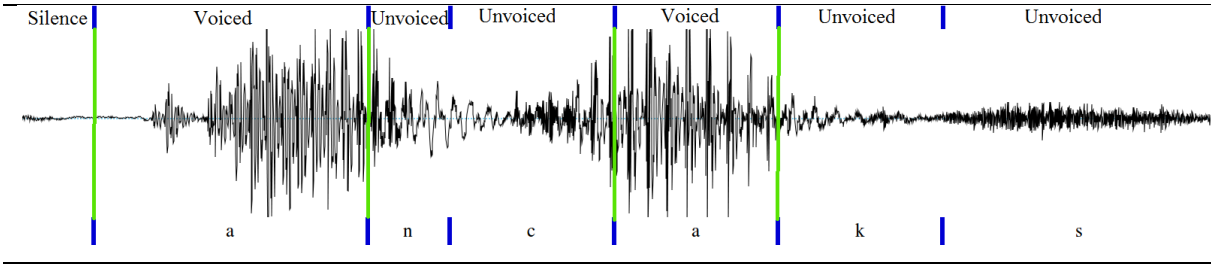


Figure 3.4 Voiced, Unvoiced and Silence regions of one word. The "s" at the seventh frame is the first letter of the next word.

3.2 Short-Time Fourier Representation of Speech

Previously, the quasi-periodic property of speech signal has introduced. Then we have mentioned that this property of speech signal was leaded short-time analysis of speech signal. Obviously the speech signal is not processed not only in time domain, but also in frequency domain. The details of speech processing in frequency domain will be discussed at section 3.5, however above we will define the short-time representation of speech and spectrogram which is a gray-scale graph shows magnitudes as darkness, of all frequencies at vertical axis and for all time instances at horizontal axis.

Equation 3.1 shows the short-time Fourier representation of speech signal. In Equation 3.1 $x_c(t)$[2] represents the whole speech signal, $\hat{t}$ represents a specified analysis time, Ω represents radian frequency and $w_c(\hat{t}-t)$ represents time-localized analysis window. Hence Equation 3.1 takes the Fourier Transform of time-localized speech signal window.

$$\mathcal{F}\{X_c(\hat{t},\Omega)\} = \int_{-\infty}^{+\infty} w_c(\hat{t}-t)x_c(t)e^{-j\Omega t}dt \tag{3.1}$$

In 1930's Fourier representation of speech was printed by using sonograph, also called sound spectrograph [3.2-3]. The sonograph and mechanical block diagram of sonograph has shown at [2]. Today instead of such machines, computer programs such as Wavesurfer, MATLAB spectrogram built-in function, Praat, etc. is being used. However computer

programs perform short-time Fourier transform in discrete time which has shown at Equation 3.2.

$$X_{\hat{n}}(e^{j\omega}) = \sum_{-\infty}^{+\infty} w[\hat{n} - m]x[m]e^{j\omega m} \tag{3.2}$$

Where; $\hat{n}$ represents a specified time analysis, w represents normalized frequency for discrete-time signals, $x[m]$ represents whole speech signal and $w[\hat{n} - m]$ represents a time localized window of $x[m]$. Furthermore analog frequencies $\Omega_k = (2\pi k/N)F_s$ for $k = 0,1,\dots,N/2$ can be evaluated efficiently by using N-point fast Fourier transform (FFT) algorithm [2]. Figure 3.5 shows a general algorithmic block diagram printing spectrograph.

As shown at Figure 3.5, at first step the speech signal is converted to electrical waveforms by using microphone and visualized as speech signal at time domain, which shown at upper part of Figure 3.6. At the second step a frame of speech signal is taken, where maximum 3 or 4 second long frames are used to visualize the spectrogram of the speech signal which shown at the down part of figure 3.6 for a single Turkish word "*amerikan*". At the third step, the whole frequency ranges are set, where 0 Hz is the last line of the horizontal paper and 5 kHz is the top line of the horizontal paper. At the fourth step the frequency range between 0-5 kHz is scanned by using band pass filters. According to the selection of the bandwidth of the band pass filter, either narrowband spectrograms (BW = 30-90 Hz) or wideband spectrograms (BW = 300-900 Hz) could be displayed.

Finally the resulting short-time spectrum $\left|X_{\hat{n}}\left(e^{j(2\pi k/N)F_s}\right)\right|$can be displayed as function of $\hat{n}$ and k on computer screen either colored or grayscale image. Figure 3.6 shows an example of such gray scale image, which is viewed by using the *Praat* [58] for a Turkish word "*amerikan*". The average energy of selected frequency is printed by black color on corresponding frequency location (vertical dimension) and corresponding time location (horizontal dimension).

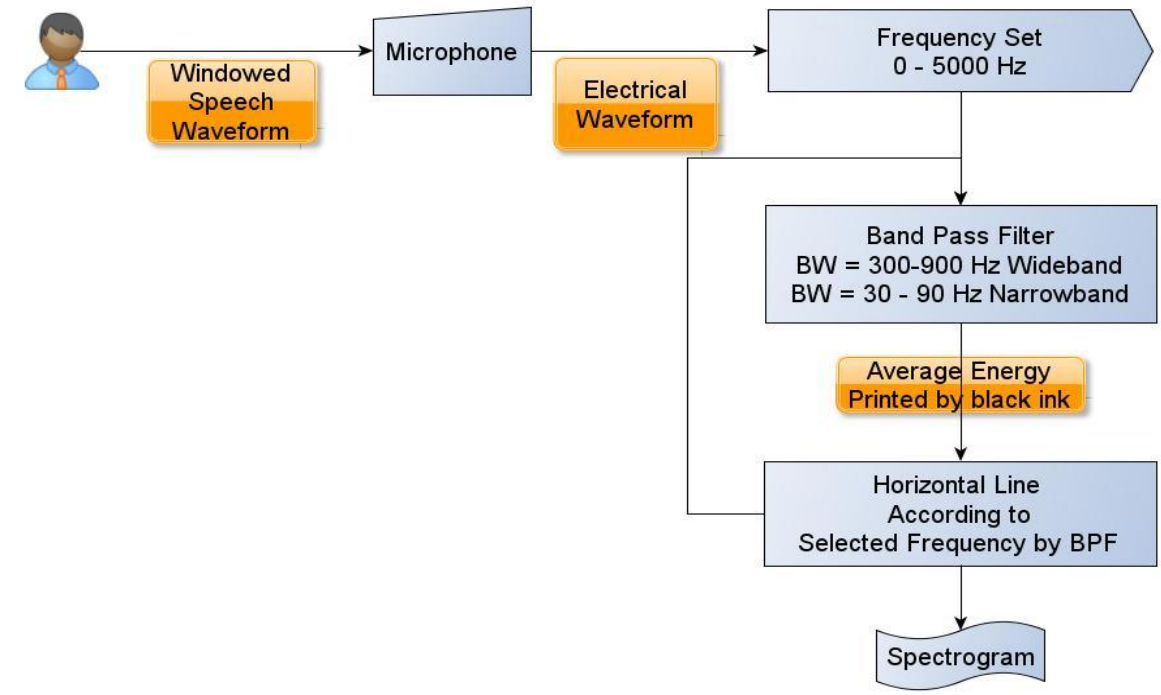


Figure 3.5 Algorithmic block diagram of performing spectrograms by computer programs.

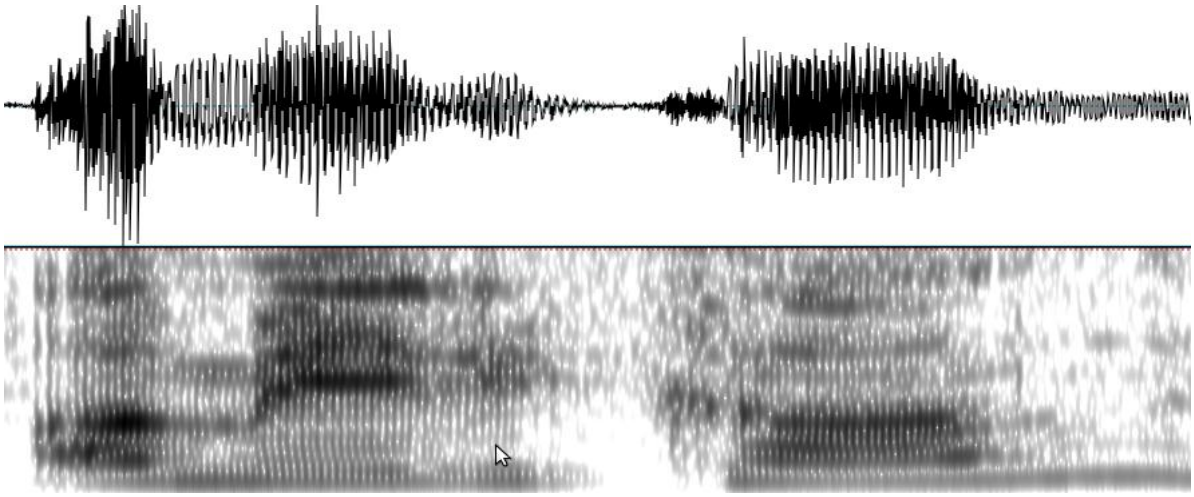


Figure 3.6 Electrical speech signal at upper part and corresponding spectrogram for the Turkish word “amerikan”.

There are several advantages and disadvantages of using either narrowband or wideband spectrograms. Wideband spectrograms have good temporal resolution but its poor at

frequency resolution. On the other hand, narrowband spectrograms are better in frequency resolution but worse in temporal i.e. time resolution [2]. Moreover in wideband spectrograms, voiced regions are visualized by vertical striated lines due to the periodicity of the time waveform and the harmonics of fundamental frequencies are shown at vertical dimension of narrowband spectrogram as horizontal striated appearance. If the window is long enough, several pitch periods could also be visualized. Figure 3.7 shows such fundamental frequencies with red dots and pitch periods with blue dots and also intensity with yellow line is displayed for the signal shown at Figure 3.6.

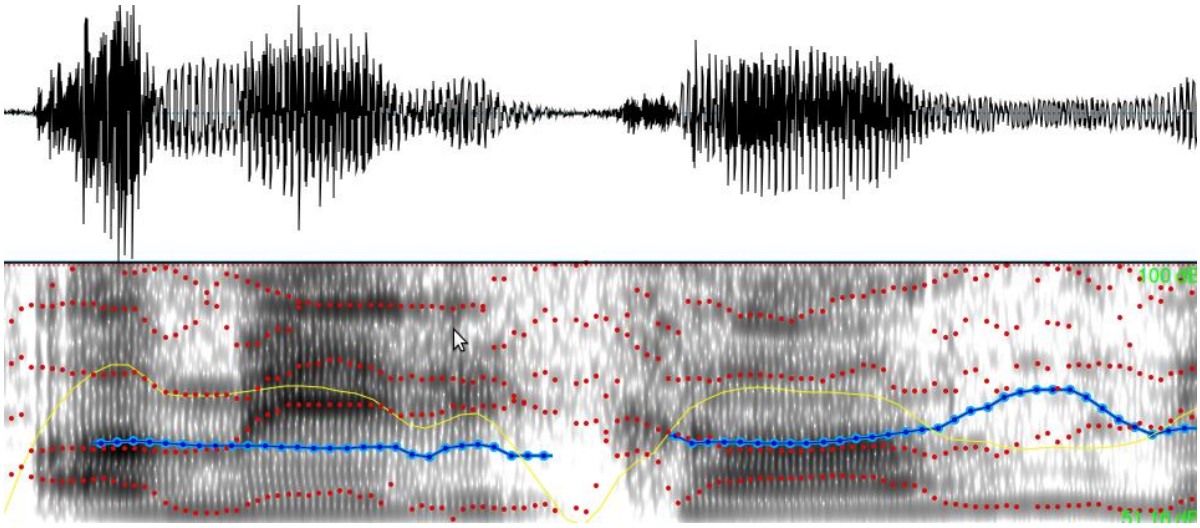


Figure 3.7 Electrical speech signal at upper part and corresponding spectrogram for the Turkish word “amerikan”, with intensity, pitch and fundamental frequency displays.

In addition, the band pass filter bandwidths are highly dependent on pitch frequency of the speaker. For instance for a male speaker who has a pitch frequency approximately at 100Hz, band pass filter with 300 Hz bandwidth is used and for a female speaker who has approximately 200 Hz pitch frequency, band pass filter with 600 Hz bandwidth is used [2].

3.3 Acoustic Phonetics

In order to design best speech processing systems such as either speech synthesis or speech recognition systems, we should know articulatory properties of all sounds (phonemes) of the language which we study on. Previously, at Figure 2.3 we have shown some examples of the relation between utterance of phonemes for corresponding Turkish and English words and we have mentioned that in English, the written language and utterance of phonemes i.e. orthography were different, and in Turkish they were exactly same. Furthermore we have introduced phoneme alphabets such as IPA and ARPAbet. Below a list of Turkish phonemes with IPA and ARPAbet representations and also an alternative representation of Turkish phonemes which we have used in HTK (Hidden Markov Toolkit) and Mary Harpers prosodic feature extraction tool on *Praat* are shown. There are several Turkish letters which are not found in English. The Unicode representations of that letters are given at Table 3.2.

Turkish phonemes are divided into several classes such as vowels and consonants, in addition vowels and consonants are also divided into several sub-classes. Below those classes and properties of those classes will be given. Initially Table 3.3 shows the list of vowels and consonants."

The vowels are produced by extraction of air in the lungs by vibrating vocal cords without any brush. Also there is no restriction on vocal tract. The position of tongue, lips and chin affects to produce vowels. The vowels are divided into two classes such as back vowels and front vowels depending on the position of tongue and exit of sound. Furthermore the back and front vowels are also divided into two classes such as rounded vowels and unrounded vowels, where the position of lips determines this class and also unrounded and rounded vowels are divided into two classes which are called wide and close vowels and this property belongs to the position of mouth [24]. Table 3.3 shows the vowels in Turkish alphabet according to their classes.

IPA Alphabet	IPA Phoneme	ARPAbet	HTK	IPA Alphabet	IPA Phoneme	ARPAbet	HTK
A	/a/	AA	a	M	/m/	M	m
B	/b/	B	b	N	/n/,/ɲ/	N, NX	n
C	/dʒ/	JH	c	O	/o/	OW	o
Ç	/tʃ/	CH	C1	Ö	/œ/		O1
D	/d/	D	d	P	/p/	P	p
E	/e/,/æ/	EH, AE	e	R	/r/	R	r
F	/f/	F	f	S	/s/	S	s
G	/g/	G	g	Ş	/ʃ/	SH	S1
Ğ	/ɯ/		G1	T	/t/	T	t
H	/h/	H	h	U	/u/,/ʊ/	UH	u
İ	i	IY	i	Ü	/y/	Y	U1
I	/ı/	IH	I1	V	/v/	V	v
J	/ʒ/	ZH	j	Y	/j/	JH	y
K	/k/	K	k	Z	/zh/	ZH	z
L	/l/,/ɬ/	L	l				

Table 3.1 Condensed list of phonetic symbols for Modern Turkish[23].

Letter	Unicode	Capital Letter	Unicode
ç	231	Ç	199
ğ	287	Ğ	286
ı	305	İ	304
ö	246	Ö	214
ş	351	Ş	350
ü	252	Ü	220

Table 3.2 Turkish letters which does not exist in English Alphabet[23].

Vowels	a, e, ı, i, o, ö, u, ü
Consonants	b, c, d, f, g, ğ, h, j, k, l, m, n, p, r, s, ş, t, v, y, z

Table 3.3 Vowel and Consonant letters of Modern Turkish alphabet.

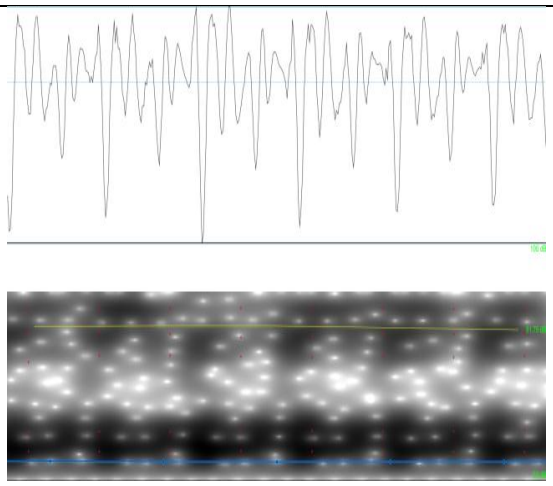
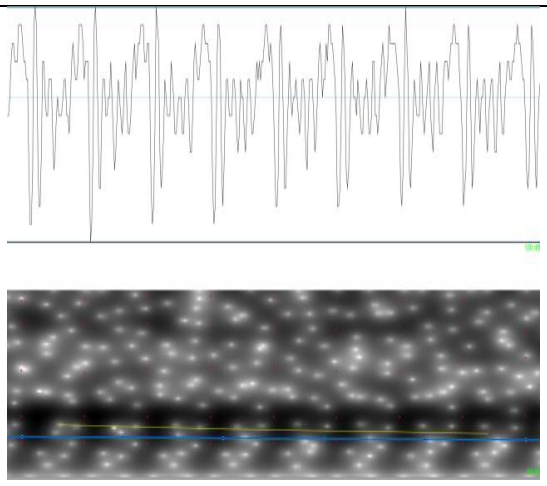
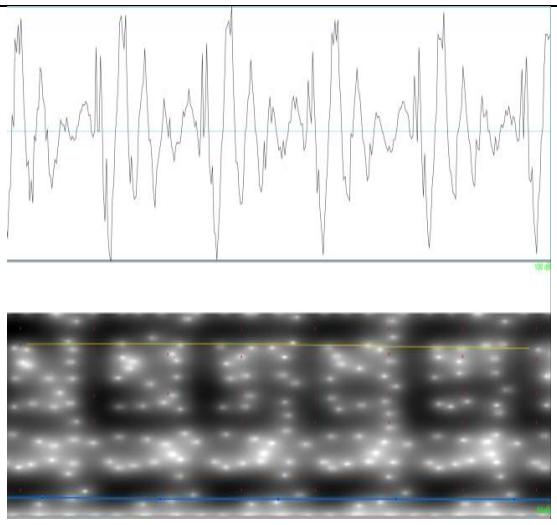
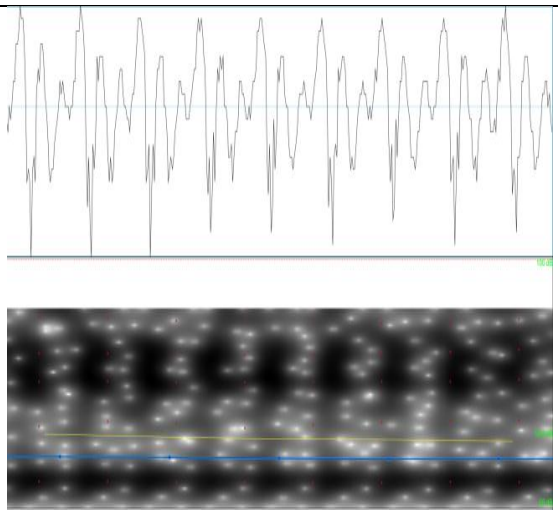
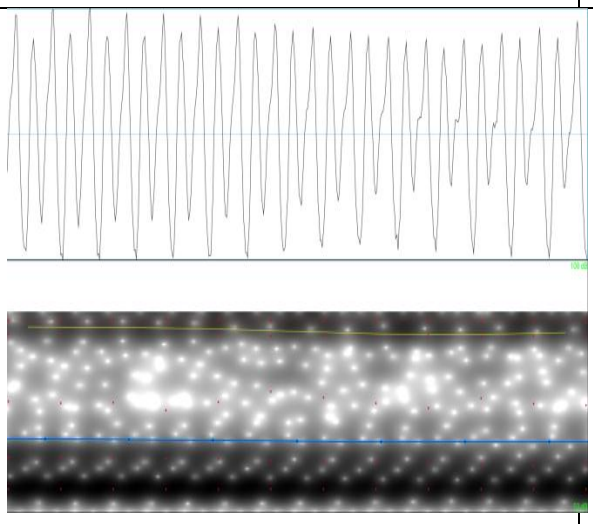
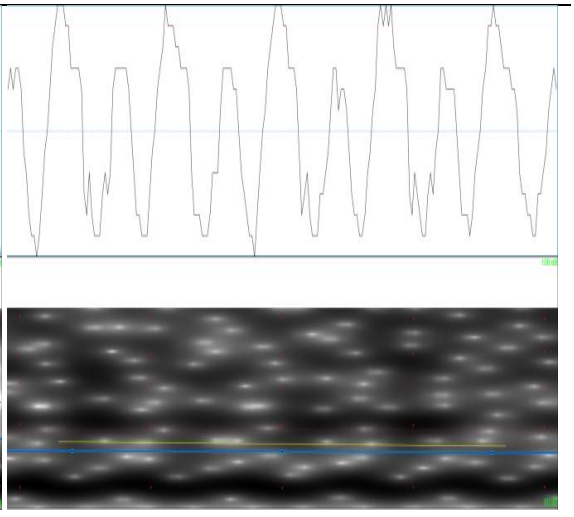
Vowels	Unrounded Vowel		Rounded Vowel	
	Wide	**Close**	**Wide**	**Close**
Back Vowel	a	I	o	u
Front Vowel	e	i	ö	ü

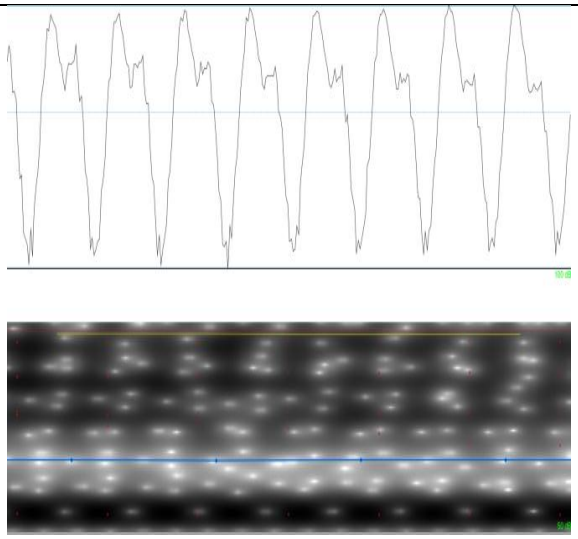
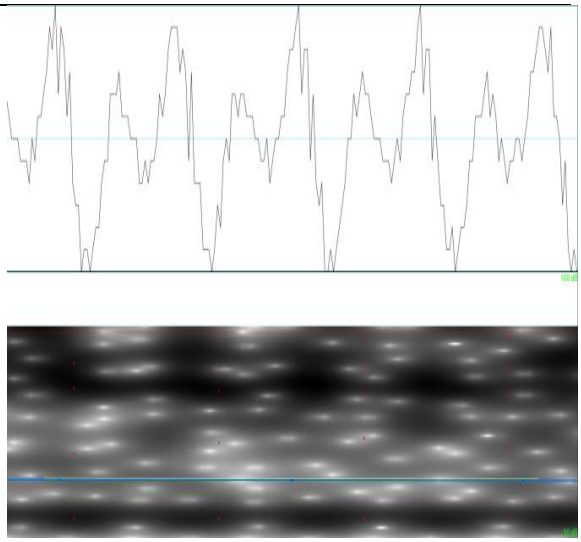
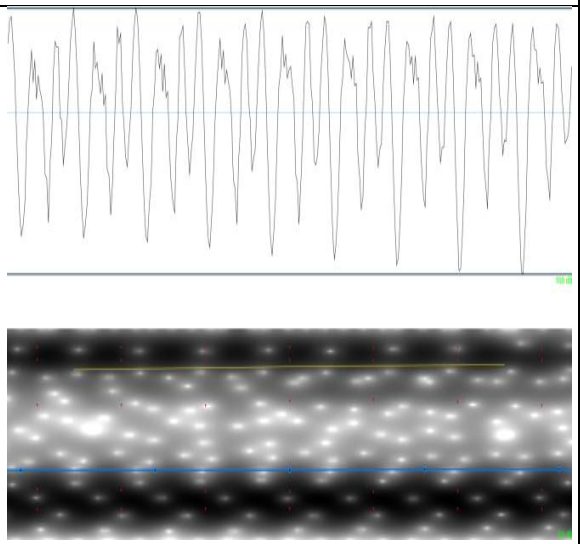
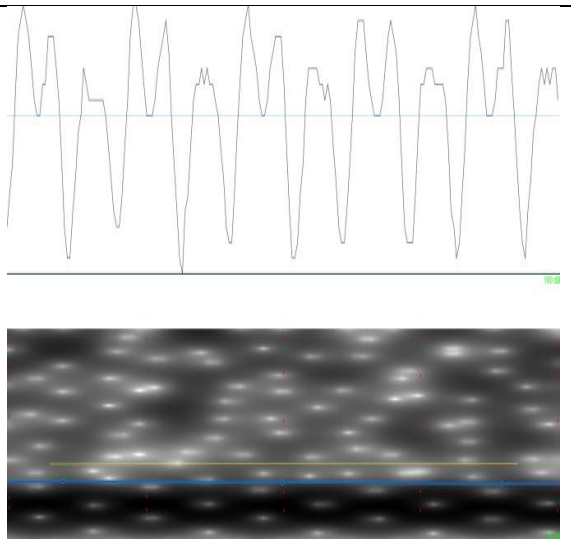
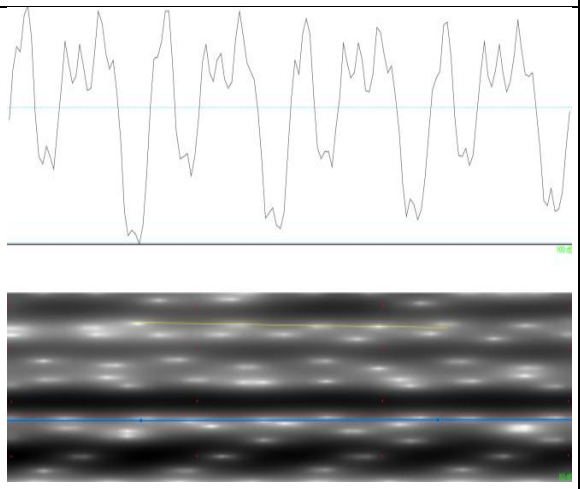
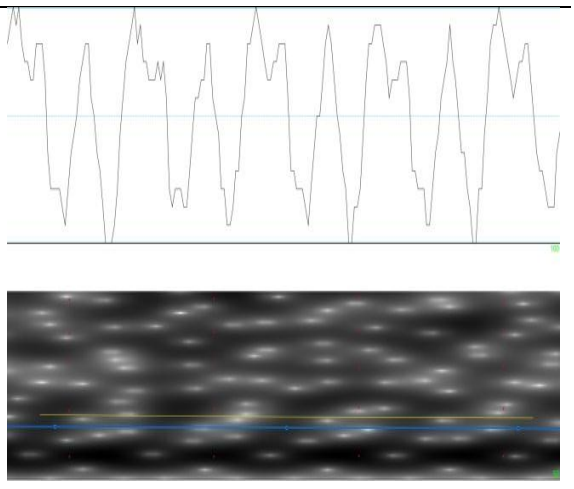
Table 3.4a Classification of vowels belongs to modern Turkish alphabet.

In addition the cross-sectional area varies along the vocal tract determines the resonant frequencies i.e. formants and thus the sound that is produced [2]. The timbre of the sound is belongs to first three formants, first two of them (F1 and F2) are the most effective ones. The formant differences are defined as the difference between first and second formant. This difference is highly correlated with the wideness of mouth while producing the sound. The time-domain representations, spectrograms and formant frequencies of vowels are shown below.

Vowel	F1	F2	F3
a	628.9	1259.3	2706.2
e	485.6	1834.0	2614.1
ı	537.4	1577.5	2722.0
i	286.1	2177.9	2942.7
o	467.7	1064.5	2695.4
ö	543.9	1516.7	2549.3
u	309.9	908.8	2400.9
ü	372.1	1632.7	2369.3

Table 3.4b Average Formant Frequencies of vowels spoken by adult men [56]

	Male	Female
a		
e		
ɪ		

i		
o		
ö		

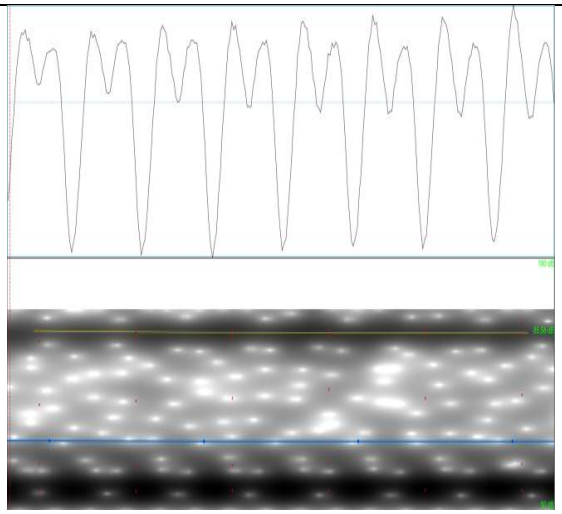
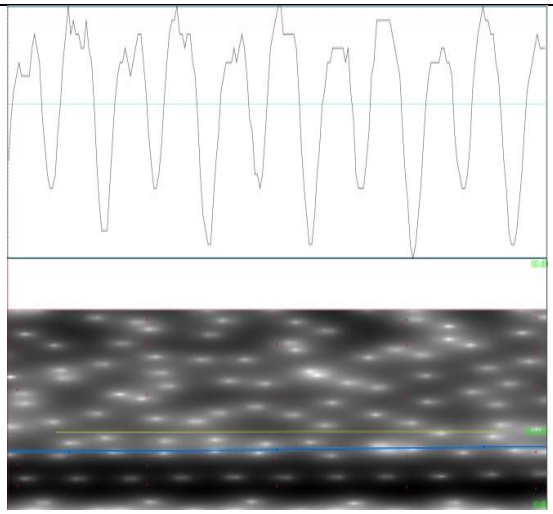
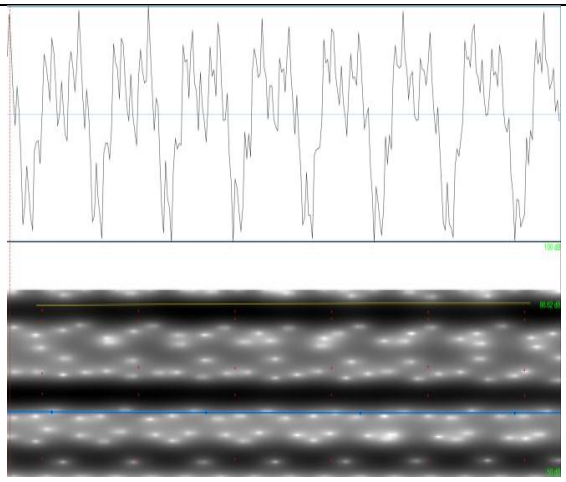
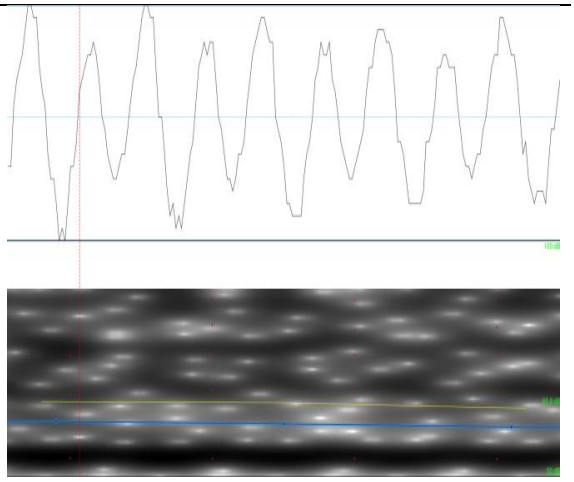
u		
ü		

Table 3.5 The vowels in modern Turkish Language. The signals in time domain and spectrum representations (0-5kHz) are shown by *Praat* for male (left side) and female (right side) speakers.

The consonants are divided into two main classes such as voiced consonants and unvoiced consonants; also they are divided into sub-classes which are shown at Table 3.6. On the other hand, consonants are divided into subclasses according to occurrence location. Table 3.7 shows this classification of consonants [2, 24]. The production of consonants is explained below [2].

The semivowels has vowel like nature. In order to produce semivowels, the vocal tract is constricted but no turbulence is created so the air flow is not totally blocked. The transition

with adjacent vowels (the vowels before or after the semivowel) is so smooth and because of that the semivowels are affected by adjacent vowels.

Consonants	Fricatives	Stops	Nasals	Semivowels
Voiced	c, j, v, z	b, d, g	m, n	ğ, l, r, y
Unvoiced	ç, f, h, s, ş	t, k, p	-	-

Table 3.6 Classification of consonants in modern Turkish is shown.

Labial Consonants	b, p, m
Labial-Dental Consonants	f, v
Dental Consonants	d, t, n, s, z
Palatal-Dental Consonants	c, ç, j, ş
Front-Palatal Consonants	g, k, l, r, y
End-Palatal Consonants	ğ
Glottis Consonants	h

Table 3.7 Classification of consonants in modern Turkish according to occurrence location is shown.

The nasals are produced by levering velum in order to air flows from nasal tract. They are produced with glottal excitation.

The unvoiced fricatives are produced as follows. The steady air flow is constricted in vocal tract and becomes turbulent. The produced unvoiced consonant sound is depends on the location of the constriction, i.e. constricted vocal organ.

The counterparts of the unvoiced fricatives are voiced fricatives. The similarity with unvoiced fricatives is the locations of the constrictions are same. However, the main difference is that in voiced fricatives there are two excitation sources are involved in order

to produce those sounds. While vocal cords are vibrating also turbulent occurs in the neighborhood of the constriction.

The voiced stops are produced by building up pressure behind somewhere in oral tract by constriction and suddenly releasing the pressure. The location of constriction effects the phoneme such as /b/ produced at lips, /d/ is produced at back teeth and /g/ is produced at velum. The voiced stops are transient and non-continuant sounds.

The unvoiced stops are similar with voiced stops. The counter parts of voiced stops /b/, /d/ and /g/ are /p/, /t/ and /k/ respectively. The main difference is that during the period of total closure of the vocal tract, as the pressure builds up, the vocal cords do not vibrate, which is similar with the difference between voiced and unvoiced fricatives.

The affricates i.e. whispers are not take place at Table 3.6. They are dynamic sounds which can be modeled as the concatenation of a stop and a fricative such as combining /d/ and /*ž*/ in order to produce the phone /j/. Also /h/ is an affricative which is produced by exciting vocal tract by steady air flow without vibrating vocal cords; the turbulent flow is produced at glottis.

3.4 Speech Signal Processing in Time Domain

The analysis speech signals in time domain is that to analysis the electrical signal recorded by the microphone. On the other hand, in the following sections Fourier representations of speech signal will be used. The analysis of sound propagation in the human vocal tract, Rabiner, Schafer et al, leads to the source/system model for speech production which has shown at Figure 3.8. This model will help us to model and process the speech signal. Previously we have mentioned that the speech signal was varied slowly and we have also showed how to frame the speech signal. In this section we will compare rectangular windowing which has shown at Figure 3.2 with hamming windowing while introducing several time domain representations such as short-time log energy, short-time zero crossing

rate and short-time autocorrelation function which are simple to implement and useful to estimate to important parameters of the speech signal.

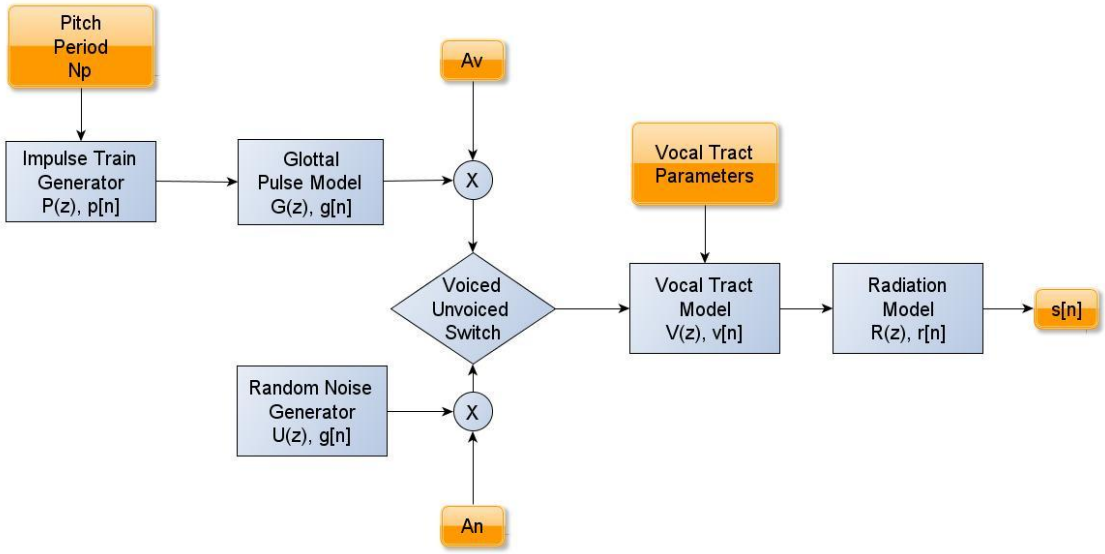


Figure 3.8 Model for speech production and synthesis [2].

Figure 3.8 shows also the information carried by speech signal s[n]. At the beginning N_p defines the time-varying pitch period in terms of samples. The pitch period F_p is defined by using sampling frequency F_s and N_p as $F_p = F_s / N_p$ in voiced signals. Then the glottal pulse model is represented by g[n] in time domain. The convolution of g[n] and p[n] is multiplied with A_V where A_V represents time-varying amplitude of voiced excitation. On the other hand for unvoiced case, the random noise generator g[n] represents the consonants between vowels. The voiced and unvoiced transitions are modeled as a switch where voiced signals are quasi-periodic and have steady variations of fundamental (pitch) frequency and unvoiced regions are modeled as pseudo-random noise. That difference between voiced and unvoiced regions provides useful clues to estimate voiced and unvoiced parts of the speech signal

3.4.1 Short-Time Analysis of Speech Signal

As mentioned before, speech signal varies slowly in time. Furthermore, speech signal is not a periodic signal in a long time-period however in time periods 20-100msec, the speech

signal is observed as stationary and quasi-periodic. This property of speech signal leads short-time analysis of speech as shown at Figure 3.2. The general representation of short-time analysis is shown at Figure 3.9, where s[n] represents the input speech signal. The s[n] is filtered by using a linear filter in order to isolate a frequency band. Filtered signal is represented by x[n]. In the next step, the *T(.)* represents either linear or non-linear transform in order to make some property of speech signal more significant. Those transformations will be discussed below more detailed. Finally the *LPF* block is the low-pass windowing sequence, i.e. framing operation for a particular time "$\hat{n}$". The output $Q_{\hat{n}}$ represents vector of values depends on particular analysis time "$\hat{n}$". Equation 3.3 is the mathematical representation of $Q_{\hat{n}}$ which has shown at Figure 3.9.

As shown in the Equation 3.3, the transformed signal is convolved with the impulse response of a linear filter, i.e. $\tilde{w}[n]$. Equation 3.4 and 3.5 shows rectangular and Hamming window respectively and Figure 3.10 and 3.11 shows corresponding impulse responses for number of samples L is 21.

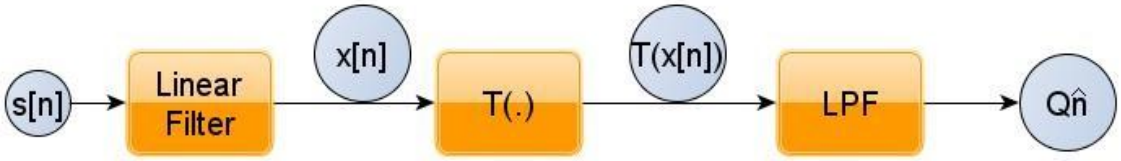


Figure 3.9 General representation of short-time analysis in time domain [2].

$$Q_{\hat{n}} = T(x[n]) * \tilde{w}[n] \tag{3.3}$$

In short-time speech signal processing, the starting point of the window is shifted by 1< R < L samples at each step. Previously we have defined $\hat{n}$ as a particular time of windowing operation, where it is integer multiplies of *R*. Hence, the output is down sampled by a factor of *R;* i.e. the short-time representation is evaluated at times $\hat{n} = rR$, where *r* is an integer [2]. Doing short-time evaluations at integer multiplies of *R<L* causes overlap between the frames, and in general the length of overlap is selected as a little more than half of the frame length. For instance consider the Equation 3.5 and Figure 3.11. For each frame, the edges of the frame amplitude will be reduced, and at the either previous or following frame,

the reduced part of the original signal takes place on at the middle of the window, so the amplitude of that samples will be maximized at that step. Hence, framing by using overlap avoids data loss.

$$W_{Rectangular}[n] = \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & otherwise \end{cases} \tag{3.4}$$

$$W_{Hamming}[n] = \begin{cases} 0.54 - 0.46cos\big(2\pi n/(L-1)\big), & 0 \leq n \leq L-1 \\ 0, & otherwise \end{cases} \tag{3.5}$$

3.4.2 Short-time Energy and Magnitude

The energy of a given signal is calculated by using Equation 3.6. However when we apply Equation 3.6 to whole speech signal, we will get only a scalar value and we will get no information about time-dependent properties of the signal.

$$E = \sum_{m=-\infty}^{+\infty} (x[m])^2 \tag{3.6}$$

Hence short-time analysis approach will be used in order to evaluate the energy of frames, then we will obtain a vector of energy values of corresponding frames, then we will have much more useful data. Equation 3.7 shows energy calculation by using short-time processing approach [2].

$$E_{\hat{n}} = \sum_{m=-\infty}^{+\infty} (x[m]w[\hat{n}-m])^2 = \sum_{m=-\infty}^{+\infty} (x[m])^2 \tilde{w}[\hat{n}-m] \tag{3.7}$$

In the left side of Equation 3.7, the speech signal multiplied by $w[\hat{n}-m]$ first, i.e. it has filtered in time domain. The original signal is convolved with the impulse response of used window. For instance Equation 3.4 used for rectangular window and Equation 3.5 used for Hamming window.

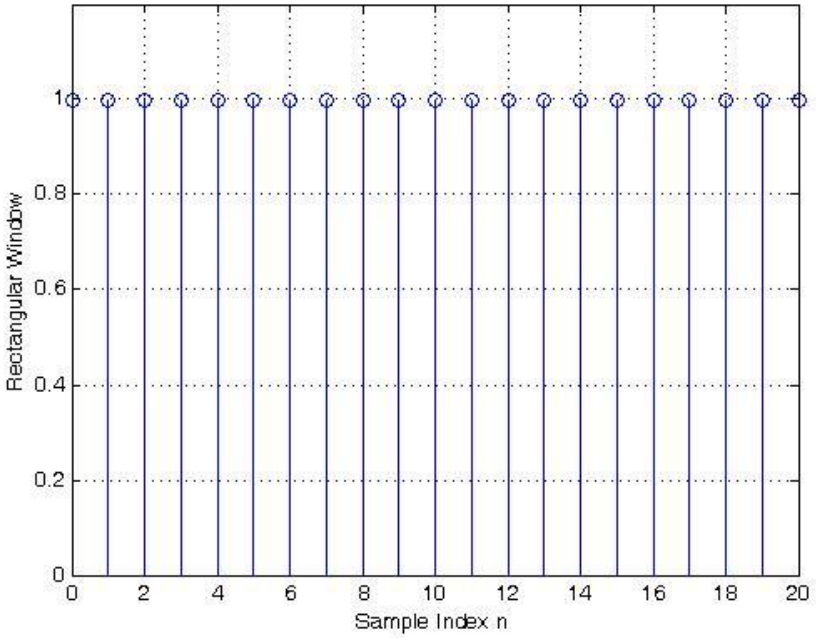


Figure 3.10 Impulse response of rectangular window.

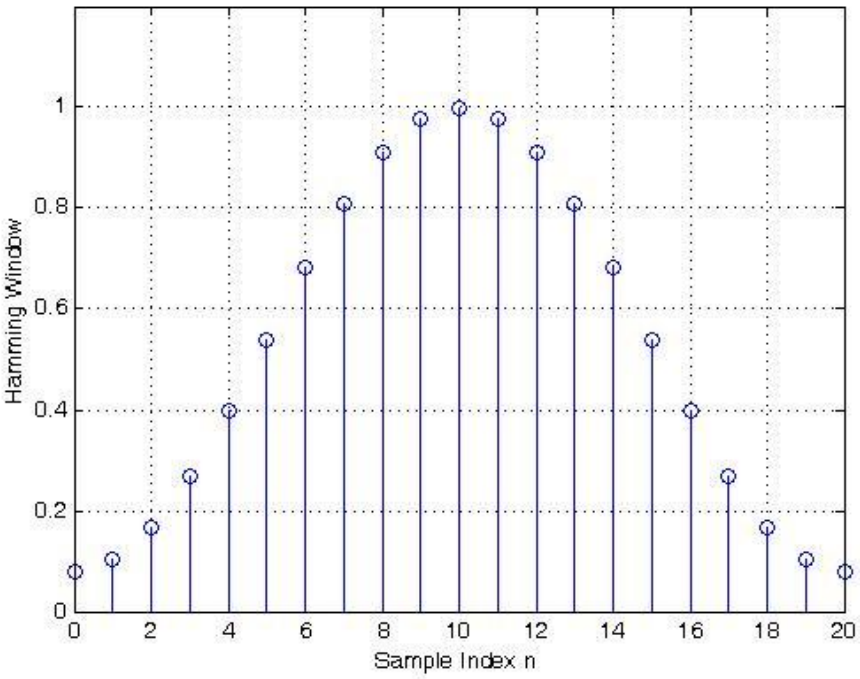


Figure 3.11 Impulse response of Hamming window.

The right side of Equation 3.7 shows this operation in the general form which has given at Equation 3.3, where the operation *T(.)* represent the squaring operation and $\tilde{w}[\hat{n}-m]$ represent the windowing operation $\tilde{w}[n]$. Figure 3.12 shows the short-time energy calculation process. For instance for L-point rectangular window, the calculation of short-time energy at time instance $\hat{n}$ has shown at Equation 3.8.

In time domain using either rectangular or Hamming window does not makes any improvement on energy calculation of the speech signal, however increasing the frame length *L* will decrease the bandwidth of the low pass filter and energy contour will be much more smoother [2].

$$E_{\hat{n}} = \sum_{m=\hat{n}-L+1}^{\hat{n}} (x[m])^2 \tag{3.8}$$

Figure 3.12 Short-time energy calculation process [2].

There are several examples at the following figures. At first Figure 3.13 shows a normalized speech signal, where DC mean value of the speech signal has removed. Then Figure 3.14 shows short-time energy functions by using rectangular window in different frame lengths. As shown in Figure 3.14 the frame length is increasing, the energy function gets smoother.

The difference of using either rectangular of Hamming window is much more prominent in frequency domain, where in frequency domain when the frequency response plots of rectangular windowed signal and Hamming windowed signal is compared, we will see that Hamming windowed plot will have much more smoother curves.

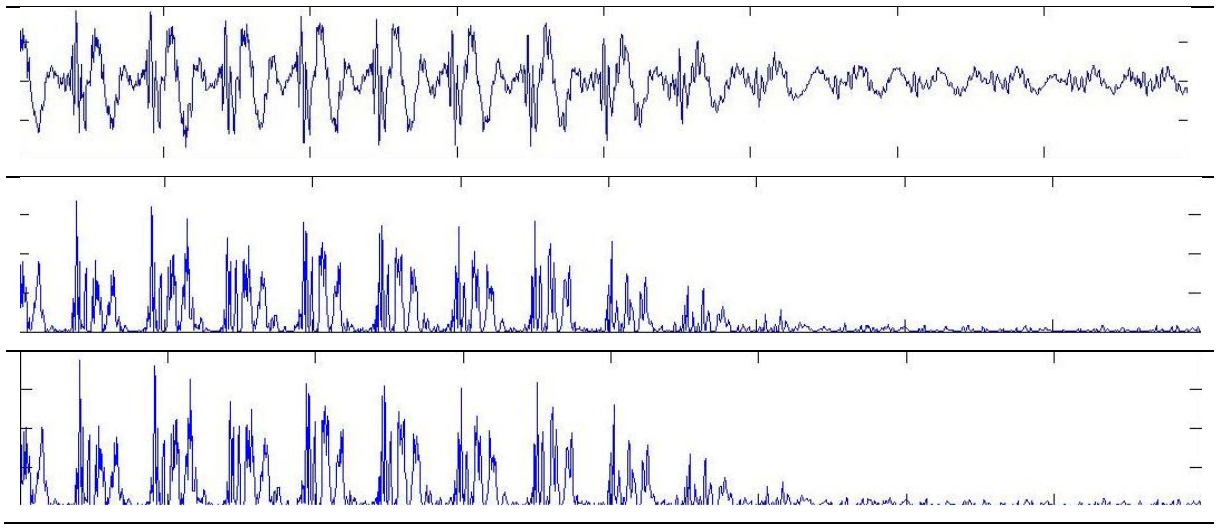


Figure 3.13 The signal at top shows 0.1 second long normalized speech signal. The figure at the middle shows energy signal of the speech signal calculated by using rectangular window and the figure at bottom shows energy signal of the speech signal calculated by using Hamming window. L=1601 samples, R=800 samples and Fs=16000Hz.

As shown in Figure 3.11, Hamming window reduces the energy of signal in the edges. In other words, while using Hamming window as a low pass filter, the high frequency components are reduced much more than rectangular window. The examples are shown at section 3.5.

On the other hand, the energy function gives clues about the locations of voiced and unvoiced segments in the time domain. For instance in Figure 3.13 a 0.1 seconds long speech signal has shown with energy signals. At the approximately first 0.07 seconds we see that we have a voiced speech signal (see Figure 3.4 also) and there are periodic movements at energy signal. However in approximately last 0.03 seconds, we see the unvoiced region of the speech signal and the energy signal is similar with a noise signal. Furthermore, the energy of voiced segments is much greater than unvoiced parts.

In other words, it has high dynamic range. Hence, advantage of that is it is easier to segment voiced and unvoiced parts on the other hand the short-time energy function is very sensitive to large signal levels [2].

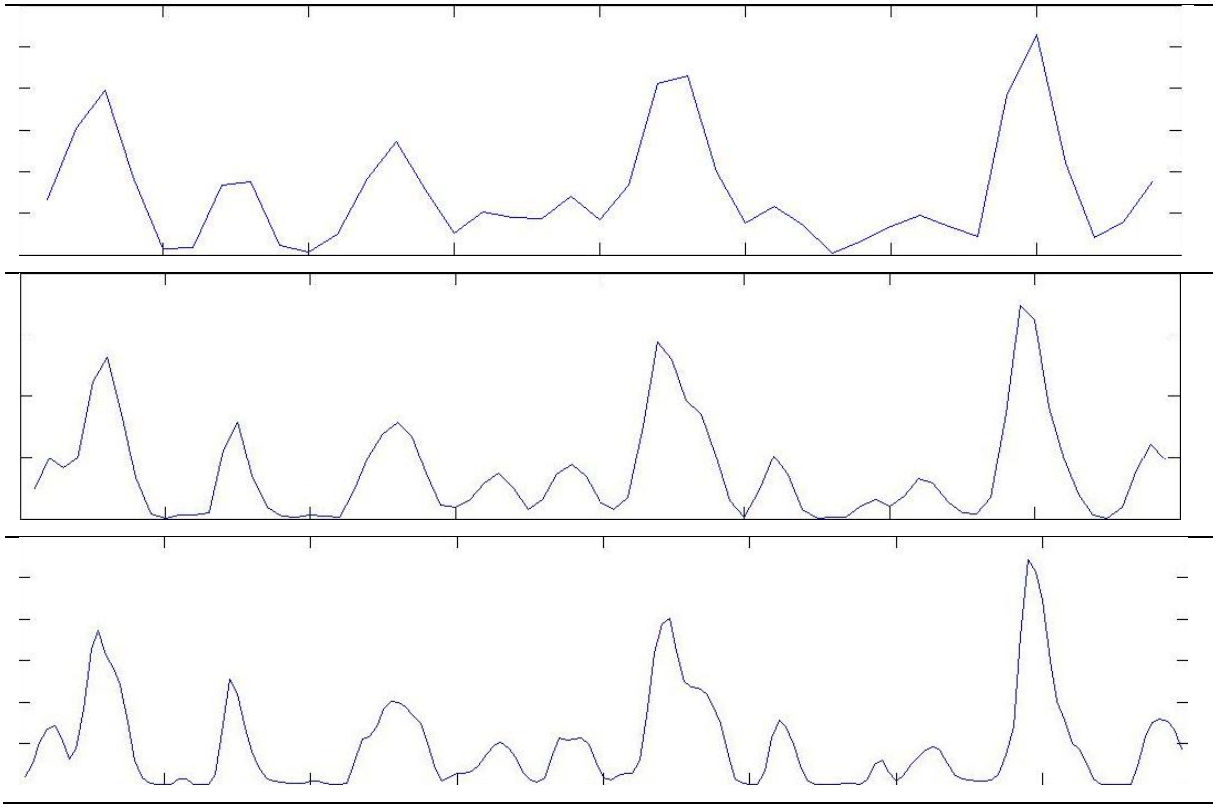


Figure 3.14 Windowed signals in different window and overlap lengths.In the first case L=1601 and R=800, in the second case L=801 and R=400 and finally at the last case L=401 and R=200.

There are two solutions to that problem. The first one is squaring the short-time energy function and the other one is instead of using magnitude function which has shown at Figure 3.15 and Equation 3.9.

$$M_{\hat{n}} = \sum_{m=-\infty}^{+\infty} |x[m]w[\hat{n}-m]| = \sum_{m=-\infty}^{+\infty} |x[m]|\tilde{w}[\hat{n}-m] \tag{3.9}$$

Figure 3.15 Short-time magnitude calculation process [2]

3.4.3 Short-time Zero Crossing Rates

Previously we have discussed that the energy of the signal helps us to detect unvoiced and voiced parts of the speech, where in the unvoiced regions the energy levels are low and at the voiced regions the energy levels are high. However, the clues of detecting voiced and unvoiced regions are not only the high or low energy levels of the signal, but also zero crossing rates. Below, at first we will define the zero crossings and how to calculate zero crossing rates. Then we will show how zero crossing rates are affected by DC noise component of the signal and finally we will show the zero crossing rates at voiced and unvoiced signals.

The zero crossing is occurred when in the time instance *n* of a signal *x* intersects the x-axis. In other words if

$x[n] > 0\ and\ either\ x[n+1]\ or\ x[n-1] < 0\ or\ x[n] < 0\ and\ either\ x[n+1]\ or\ x[n-1] > 0;$ i.e. when $sqn(x[n] \neq x[n-1])\ or\ sqn(x[n] \neq x[n+1])$.

The zero crossing rate is defined as the number of crossings per unit of a time. Figure 3.16 shows the zero crossings for sinusoidal signals. In the first signal from the top, the DC component of the signal is zero. There are two zero crossings per cycle. In the second line of the figure, a little DC component has added to sinusoidal signal. The whole signal has shifted to up and the zero crossing locations have changed. For a signal whose amplitude is constant, the zero crossings per unit of a time will not change. However, if we consider a signal which is the combination of sinusoidal signals with different frequencies and amplitude, this kind of shifting may cause to lose some of the zero crossings. In the third line, we have added a higher DC component, and the signal has shifted upper. In that signal we see that there are no zero crossings. Because of such reasons, in many speech signal processing and signal processing applications, the DC component of the signal is removed first.

As shown at the Figure 3.16 there are two zero crossings per one cycle of the sinusoidal signal. However we are interested in crossings per unit of samples. So we should consider also cycles per samples. If we consider the Equation 3.10 we will see that the crossings per cycles are multiplied with cycles per sample, where cycles per sample are the ratio of frequency of the signal and the sampling frequency.

$$Z^{(1)} = 2\frac{crossings}{cycle}\ x\ \frac{F_0}{F_S}\frac{cycle}{samples} = \frac{2F_0}{F_S}\frac{crossings}{sample} \tag{3.10}$$

The Equation 3.10 calculates the zero crossings per one sample. However we are interested in zero crossings on a fixed interval such as length of used either rectangular, hamming or another suitable window. In other words we are interested in the number of crossings in the interval of *M* samples. So we multiply the $Z^{(1)}$ with *M* in order to find crossings per M samples and represented by $Z^{(M)}$.

The $Z^{(M)}$ is used to estimate the frequency of a sine-wave by using Equation 3.11, where F_e is the approximate sinusoidal frequency of a given zero crossing rate $Z^{(1)}$ of a signal. Furthermore if the signal is single sinusoid, then $F_e = F_0$ and if the signal is not sinusoid, then F_e represents equivalent sinusoidal frequency of the signal [2].

$$F_e = 0.5F_sZ^{(1)} \tag{3.11}$$

In short-time analysis, the zero crossing rate calculations are shown at Figure 3.17 and Equation 3.12. The first step is to determining *sqn* function of the signal, then first difference equation is applied to *sqn* function of the signal as shown in Equation 3.12 and at the second block of Figure 3.17. Afterwards absolute value is taken and finally the windowing operation is applied. In general, rectangular windows are preferred [2].

In Equation 3.12 inside the absolute value operation, for each sample pairs, if there a zero crossing exist the value of $|sqn(x[m]) - sqn(x[m-1])|$ will be 2 and 0 otherwise. So ½ coefficient stands for normalize this count, and L_{eff} represents effective window.

Previously we have mentioned that energy distribution of the speech signal was giving us important clues to detect voiced and unvoiced regions. Also zero crossing rates give us important clues on detecting voiced and unvoiced signal. In general voiced signals have high energy and low zero crossing rates and unvoiced signals have low energy and high zero crossing rates.

$$Z_{\hat{n}} = \frac{1}{2L_{eff}} \sum_{m=-\infty}^{+\infty} |sqn(x[m]) - sqn(x[m-1])|\, \hat{w}[\hat{n} - m] \tag{3.12}$$

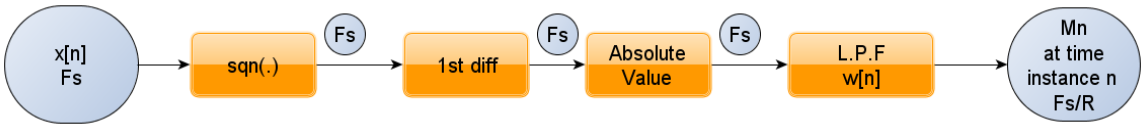


Figure 3.17 Block diagrams of short-time zero crossing rate calculation.

3.5 Speech Signal Processing in Frequency Domain

In speech processing applications, not only time-domain analysis used but also the frequency-domain analysis has been used.

In this subsection the frequency-domain analysis techniques such as Discrete-Time Fourier Analysis (DTFT), Discrete Fourier Transform (DFT), short-time Fourier Analysis (STFT) will be introduced. In addition the effect of using either rectangular or Hamming window in frequency-domain analysis and spectrographic displays will be introduced.

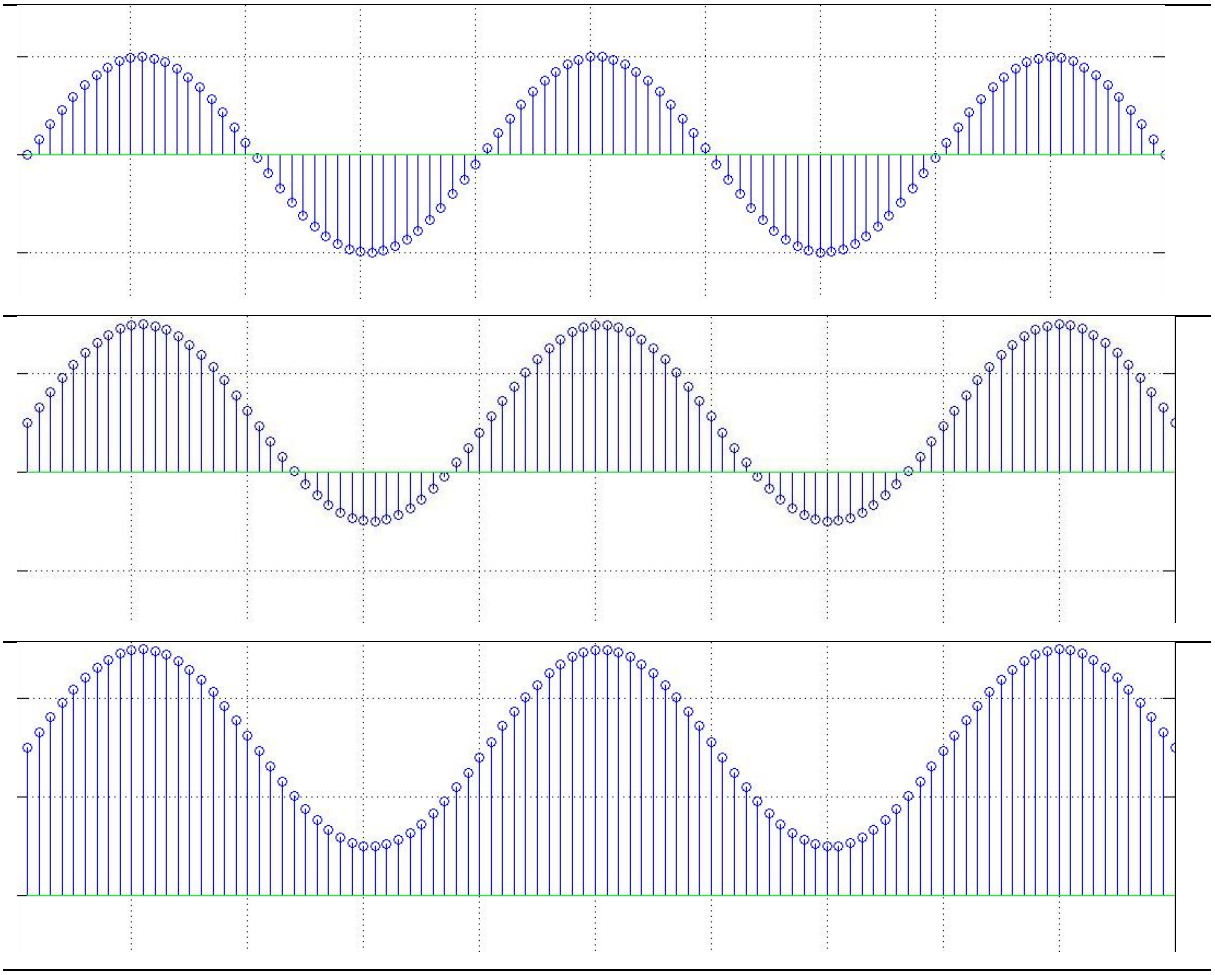


Figure 3.16 Zero crossings and effect of DC component of the sinusoidal signal to zero crossing rates. The green line shows the x-axis.

Fourier representation of the speech signal includes information about the frequency responses of the vocal tract system. Equation 3.12 and 3.13 shows voiced speech and unvoiced speech components in the frequency domain respectively, where, in Equation 3.13a P(.) represents impulse train generator, G(.) represents glottal pulse model, V(.) represents vocal tract model and R(.) represents radiation model and in Equation 3.13b the term $\left|V\left(e^{jw}\right)\right|^2$ represents vocal tract model and $\left|R\left(e^{jw}\right)\right|^2$ represents radiation model.

$$X(e^{j\omega}) = A_v P(e^{j\omega}) G(e^{j\omega}) V(e^{j\omega}) R(e^{j\omega}) \tag{3.13a}$$

$$\Phi_{XX}(e^{j\omega}) = A_N^2 |V(e^{j\omega})|^2 |R(e^{j\omega})|^2 \tag{3.13b}$$

Discrete Fourier Transform (DFT) of a speech signal includes information about average pitch frequencies of the speaker. The Discrete Fourier Transform is a sampled in frequency version of Discrete-Time Fourier Transform (DTFT). At Equations 3.14 and 3.15 Discrete-Time Fourier Analysis and Synthesis equations are given and Equation 3.16 and 3.17 shows the analysis and synthesis equations of DFT and Equation 3.18 show the relation between DTFT and DFT.

$$X(e^{j\omega}) = DTFT\left\{X[n] = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n}\right\} \tag{3.14}$$

$$x[n] = IDFT\{X(e^{j\omega})\} = \frac{1}{2\pi}\int_{-\pi}^{+\pi} X(e^{j\omega}) e^{j\omega n} d\omega \tag{3.15}$$

$$X[k] = DFT\{x[n]\} = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi k/N)n}, \quad k = 0,1,\dots,N-1 \tag{3.16}$$

$$x[n] = IDFT\{X[k]\} = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j(2\pi k/N)n}, \quad k = 0,1,\dots,N-1 \tag{3.17}$$

$$X[k] = X(e^{jw})|_{w=(2\pi k/N)}, \quad k = 0,1,\dots,N-1 \tag{3.18}$$

However to observe time-varying properties of the speech signal better, short-time Fourier Analysis (STFT) is used, where the speech signal is divided into frames and windowed then either DTFT or DFT applied to the speech signal. The time-dependent short-time Fourier Transform is defined at Equation 3.19.

$$X_{\hat{n}}\left(e^{j\hat{w}}\right) = \sum_{m=-\infty}^{+\infty} w[\hat{n}-m]x[m]e^{-j\hat{w}m} \tag{3.19}$$

In equation 3.19 the term $\hat{n}$ represents time index and $\hat{w}$ represents the frequency variable, which is continuous and periodic with period 2π and the w[.] represents the window.

An alternative representation of Equation 3.19 can be obtained by changing summation index.

$$X_{\hat{n}}\left(e^{j\hat{\omega}}\right) = e^{-j\hat{\omega}\hat{n}}\tilde{X}_{\hat{n}}\left(e^{j\hat{\omega}}\right) = e^{-j\hat{\omega}\hat{n}} \sum_{m=-\infty}^{+\infty} x[\hat{n}-m]w[m]e^{j\hat{\omega}m} \tag{3.20}$$

The length and shape of the used window affects frequency resolution of the speech signal. By using windowed frequency representations, at first the analysis becomes independent from the part of the signal which is not intersected with the window. In addition, by analyzing the frequency representation of short-time segments of the speech signal, one can understand if the speech in analyzed time instance belongs to voiced or unvoiced regions and also if it belongs to a vowel and which vowel.

The STFT analysis of speech signal divides into two sub analyses such as narrowband analysis and wideband analysis. In the case of narrowband analysis, the window length is selected as several pitch periods. This representation shows fundamental frequency better. On the other hand, in the wideband analysis case, the window length is a little bit more than one period and in this case the formant frequencies are shown better.

In addition the shape of the filter is also affects the analysis. In section 3.4, the rectangular window and Hamming window were introduced. The advantage of using Hamming window instead of using Rectangular window is that, because of Hamming window suppresses the edge parts of the corresponding frame, the high frequency components are suppressed better, hence the sharpness of pitch harmonics becomes more smooth, hence formant frequencies of the signal can be captured better.

3.6 Homomorphic Speech Signal Processing

In section 2.3.2, the human perception of sound has introduced. The basilar membrane mechanics and, the critical and the bark scale have been introduced. Human perception sound, i.e., the frequency analysis performed in the inner air, can be represented by using Mel Frequency Cepstral Coefficients. In order to explain the extraction of Mel Frequency Cepstral Coefficients, at the beginning the Homomorphic analysis of the speech model and computing the short-time Cepstrum will be introduced.

The meaning of Cepstrum is; Inverse Fourier Transform of logarithm of spectrum [2] and used to determine fundamental frequencies of human speech and also to estimate pitch period. Equation 3.21 shows the Cepstrum of a signal x[n]. However in order to perform Cepstrum analysis, Homomorphic systems should be used. In this section, Homomorphic systems will be explained.

$$c[n] = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \log\left|X\left(e^{j\omega}\right)\right| e^{j\omega n} d\omega$$

$$where$$

$$X\left(e^{j\omega}\right) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n} \tag{3.21}$$

3.6.1 Homomorphic Systems

At the beginning of defining the Homomorphic system, to visualize the concept better at first the basic properties of linear systems will be shown. Afterwards the Homomorphic system will be defined and finally the purposes of Homomorphic filtering and the way of implementing Homomorphic filters will be shown in theoretically at the next sub section.

If a system verifies the properties which are shown in Equation 3.22 is called a linear system. As shown in the Equation 3.22, if an input signal is composed of an additive

combination of elementary signals, then the output is an additive combination of the corresponding outputs. In addition a scaled input results in a correspondingly scaled output.

$$y[n] = L\{x[n]\} = L\{\alpha x_1[n] + \beta x_2[n]\} = \alpha L\{x_1[n]\} + \beta L\{x_2[n]\} = \alpha y_1[n] + \beta y_2[n] \quad (3.22)$$

In Equation 3.22, the $y[n]$ can be thought as superposition of two linear systems. Oppenheim showed that classes of non-linear systems could be defined on the basis of a generalized principle of superposition; such systems are called Homomorphic systems [2]. As shown at Equation 3.23 the definition of Homomorphic systems are similar with linear systems. The difference is that instead of addition operation on linear systems, convolution operation is used.

$$y[n] = H\{x[n]\} = H\{x_1[n] * x_2[n]\} = H\{x_1[n]\} * H\{x_2[n]\} = y_1[n] * y_2[n] \quad (3.23)$$

Recall that in Equation 3.12 and 3.13 that, the frequency response of speech signal was combination of impulse train generator, glottal pulse model, vocal tract model and radiation model for the voiced speech and combination of vocal tract model and radiation model for unvoiced speech. It is also well known that multiplication in the frequency domain implies convolution in the time domain. Hence all of the information above is the background of Homomorphic filtering.

3.6.2 Homomorphic Filtering

Homomorphic filters are used in order to separate the components of a system. The desired component passes through the system and undesired component is altered. For instance Homomorphic systems are used in order to separate the convolved excitation (pitch) and vocal tract components (formants) of the speech model. Any Homomorphic system can be represented as a cascade of three Homomorphic systems which has shown at Figure 3.18 and Equations 3.24 and 3.25.

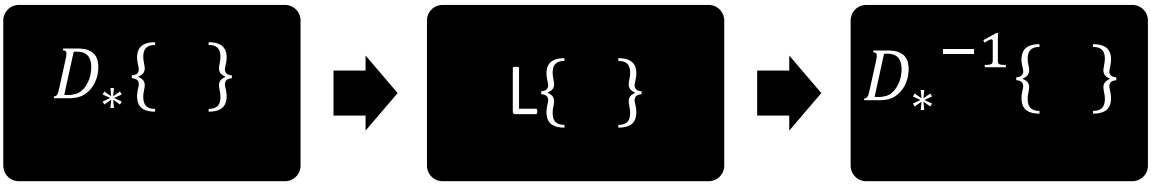


Figure 3.25 Canonic form of system for Homomorphic deconvolution in time domain [2].

In Figure 3.25 the system $D_*\{.\}$ is called characteristic system for convolution and $D_*^{-1}\{.\}$ is called inverse characteristic system for convolution. The properties of the characteristic system and the inverse characteristic system are shown at Equation 3.24 and 3.25 respectively.

$$\hat{x}[n] = D_*\{x[n]\} = D_*\{x_1[n] * x_2[n]\} = D_*\{x_1[n]\} + D_*\{x_2[n]\} = \hat{x}_1[n] + \hat{x}_2[n] \quad (3.24)$$

$$y[n] = D_*^{-1}\{\hat{y}[n]\} = D_*^{-1}\{\hat{y}_1[n] + \hat{y}_2[n]\} = D_*^{-1}\{\hat{y}_1[n]\} * D_*^{-1}\{\hat{y}_2[n]\} = y_1[n] * y_2[n] \quad (3.25)$$

The canonic form for deconvolution can be represented in frequency domain. The DTFT of $x[n] = x_1[n] * x_2[n]$is $X(e^{j\omega}) = X_1(e^{j\omega}) . X_2(e^{j\omega})$. Hence the representation of the canonic form for Homomorphic deconvolution in terms of DTFTs which is shown at Figure 3.26.

In the characteristic system for Homomorphic deconvolution in terms of DTFT, at first the DTFT of the signal $x[n]$ is computed.

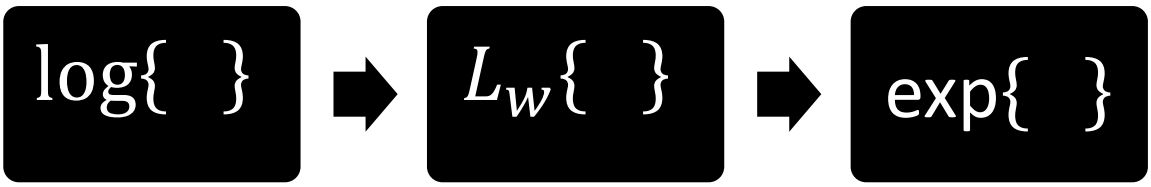


Figure 3.26 Canonic form of system for Homomorphic deconvolution in terms of DTFT [2].

At the second step, the multiplication operator in the term $X_1(e^{j\omega})$. $X_2(e^{j\omega})$ is mapped onto addition operator by using logarithm and finally by applying IDTFT, the term $X_1(e^{j\omega})$. $X_2(e^{j\omega})$ is represented as $\widehat{X_1}(e^{j\omega}) + \widehat{X_2}(e^{j\omega})$.

The term $L_w\{.\}$ represents conventional linear operator on the logarithm of DTFTs. As shown in Figure 3.25, there is a linear system at the middle block. The property of a linear system implies that the input $\widehat{X_1}(e^{j\omega}) + \widehat{X_2}(e^{j\omega})$ can be converted to the output$\widehat{Y_1}(e^{j\omega}) + \widehat{Y_2}(e^{j\omega})$.

Finally in the inverse characteristic system for Homomorphic deconvolution in terms of DTFTs, at first the DTFT of the signal $\widehat{Y_1}(e^{j\omega}) + \widehat{Y_2}(e^{j\omega})$ is computed. Then the exponential operator maps the addition operation onto multiplication operation and finally by applying IDTFT the term $Y_1(e^{j\omega})$. $Y_2(e^{j\omega})$ is obtained.

In computer implementations instead of using DTFT, using either DFT or FFT reduces the computational complexity.

3.7 Linear Predictive Analysis

In this section we will introduce the Linear Predictive Analysis of speech signal which is most widely used and powerful method used in automatic speech or speaker recognition systems to estimate the parameters of the discrete-time model such as pitch, formants, short-time spectra and vocal tract area functions, for speech production.

The Fourier representation of speech signal is used to view spectral magnitude. The Linear Predictive Coding (LPC) derives representation of spectral magnitude for signals. The LPC is computed efficiently and provides more accurate spectral resolution than non-parametric Fourier Transform techniques. Several information of speech signal can be estimated by applying LPC such as vocal fold vibration, shape of the vocal tract, and the frequencies and

bandwidth of spectral poles and zeros and LPC coefficients which are used in digital filters and in pattern recognition. [57].

Recall from short time analysis of speech signal that; the speech signal s[n] is considered a stationary signal in short-time slots. Consider the speech signal has sampled with a sampling period T, where s(n) = s(nT). In Z domain, the speech signal S(Z) is modeled as the combination of the excitation source U(Z) and spectral shaping filter which is related with shape of the vocal tract H(Z). The goal of applying LPC is to deconvolve S(Z) in order to estimate the excitation source model $\widehat{U}(Z)$ and vocal tract model i.e., filter$\widehat{H}(Z)$, where the hats represents the estimation.

$$S(Z) = U(Z) \ . \ H(Z) \tag{3.26}$$

The H(Z) can be modeled with constant coefficients where it has *p* poles and *q* zeros, where the $\hat{s}(n)$ is modeled as linear combination of *p* previous output symbol with *q-1* previous input symbols.

$$\hat{s}(n) = \sum_{k=1}^{p} a_k \hat{s}(n-k) + G \sum_{l=0}^{q} b_l \hat{u}(n-l) \tag{3.27}$$

Where in Equation 3.27 *G* represents the gain factor for excitation and a_k represents LPC coefficients which are characterizing all-pole $\widehat{H}(Z)$ model. Hence the $\widehat{H}(Z)$ can be found by using Equation 3.28.

$$\widehat{H}(Z) = \frac{\hat{S}(Z)}{\widehat{U}(Z)} = G \frac{1 + \sum_{l=1}^{q} b_l z^{-l}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{3.28}$$

Human hearing system is more sensitive to energy peaks in frequency analysis. This factor is related with vocal tract. Hence, if there is energy in a certain frequencies, it will be easier to characterize the sound. One of the Least Squares method is to characterize sound is that to minimize the mean energy in error signal over a frame, where the error signal is defined as sharp peaks separated in time by pitch period. Equation 3.29 represents the error signal and Equation 3.30 represents the energy.

$$e(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{3.29}$$

$$E = \sum_{-\infty}^{+\infty} e^2(n) = \sum_{-\infty}^{+\infty} \left[x(n) - \sum_{k=1}^{p} a_k x(n-k) \right]^2 \tag{3.30}$$

In Equation 3.30 the term x(n) represents windowed signal which is $s(n).w(n)$. To find LPC coefficients, the partial derivative of energy is set to zero $\frac{\vartheta E}{\vartheta a_k} = 0$ for all coefficients in order to find the coefficients which minimizes the energy.

On the other hand, in spectral estimation of LPC coefficients, Parseval's theorem is used.

$$E = \sum_{-\infty}^{+\infty} e^2(n) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \left| E\left(e^{jw}\right) \right|^2 dw \tag{3.31}$$

Inverse LPC filter is used in order to obtain error signal from the speech signal, where the characteristics of the inverse LPC filter is $A(Z) = G/H(Z)$.

LPC coefficients are extracted by minimizing the Energy expression in Equation 3.32. The minimum energy is also related with minimum average ratio of the signal spectrum to its LPC approximation.

$$E = \frac{G^2}{2\pi} \int_{-\pi}^{+\pi} \frac{\left| S\left(e^{jw}\right) \right|^2}{|H(e^{jw})|^2} dw \tag{3.32}$$

The optimum LPC order i.e. the value of the p should be selected in order to estimate formant frequencies of the speech. If the LPC order is selected low, then at the logarithmic plot versus frequency, the function will be too much smooth and it will not provide sufficient information about the formant frequencies. On the other hand, if the LPC order is selected too high, not only the computational complexity of the script will increase but also the harmonics of the formant frequencies will be also make peaks. Ideal order of LPC order is defined as between 8-12 at [57].

Chapter 4

The Speech Recognition Problem

In the previous chapters; we have studied human to human communication, human speech perception and production and also the properties of speech signal and analysis techniques. Now we are ready to study the communication between human and machines. The most significant advantage of using human to machine communication is cost reduction. Today most of the companies are using such systems at calling centers for several purposes. Furthermore those kinds of systems could be used not only by the companies but also humans. For instance people can listen to an e-book instead of reading it by using Text to Speech analysis and they can either control their computer or any device by their voices, where in that case Speech to Text synthesis will be used.

The overview of human to machine communication is, at the first step the human speech is analyzed by the system by using Automatic Speech Recognition system. The output of Automatic Speech Recognition system is the utterance of spoken words i.e. Speech-To-Text analysis has performed. In the second step, the system processes the input message according to a specified algorithm and finally responses to human by using Text-To-Speech Synthesis.

In our work, we have focused on the first step of human to machine communication, which is Automatic Speech Recognition (ASR) System for Turkish Spoken Language. The ASR in general is introduced in this chapter and approaches of ASR have explained. In Chapter

5, the Dynamic Time Warping, Hidden Markov Modeling approaches, acoustic models and language models has explained in more detail and finally the Hidden Markov Toolkit applications have explained.

4.1 Introduction to Automatic Speech Recognition

The ASR system is used to convert an input speech signal into transcription of spoken words, i.e. doing speech to text analysis, as efficient and accurate as possible. At the beginning we will show the overall speech recognition system. The speech recognition system has two main parts as shown at Figure 4.1 [2] which is called acoustic processor and linguistic decoder.

The acoustic processor part which is the first main part of ASR system converts the speech signal into set of either spectral or temporal features, such as Mel Frequency Cepstral Coefficients (MFCC's). Then the linguistic decoder part of the ASR, which is the second main part, performs a best maximum likelihood estimate of the words of the spoken sentence by using acoustic model, language model and word lexicon. Finally a confidence score is given for each recognized word.

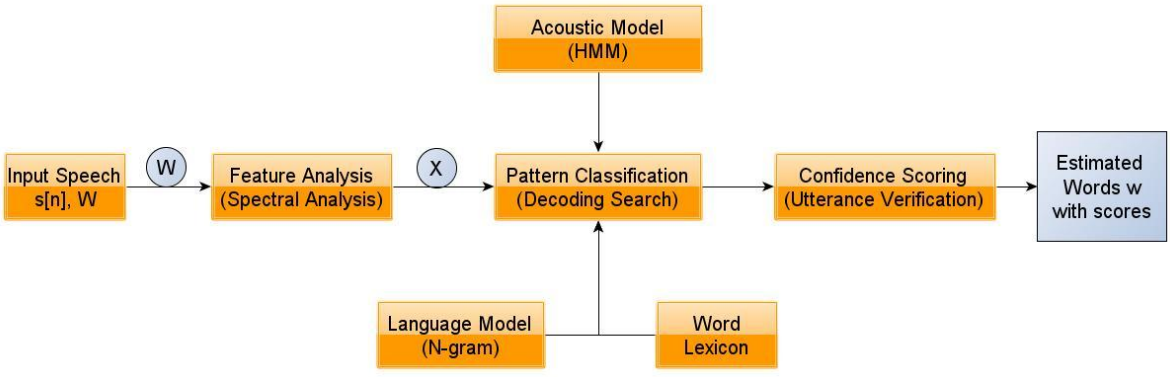


Figure 4.1 Block diagram of overall speech recognition system [2].

In Figure 4.1, the first block, which is also the first main part, performs short-time spectral analysis. Mel Frequency Cepstral Coefficients are used to represent short-time spectral

characteristics of the speech signal. Extracting MFCC by using HTK HCopy tool will be explained in detail at Chapter 5. Then the pattern classification block which is the first block of the linguistic decoder, decodes the sequence of feature vectors which are extracted by acoustic processor, into a symbolic representation that is maximum likelihood string. Pattern classification block uses the acoustic model which is defined by HMMs, an N-gram language model which computes probabilities of possible word strings and word lexicons which includes utterance words, i.e. phoneme transcription of each word in the dictionary. The final block represents confidence scoring, where the system scores each recognized words and it is possible to detect wrong recognized words by using that confidence scores [2].

4.2 Approaches of ASR

The overall ASR system has summarized above. In this section, different approaches of ASR system such as the acoustic-phonetic approach, the pattern recognition approach and the artificial intelligence approach will be explained with strengths and weakness of them briefly [6].

4.2.1 Acoustic-Phonetic Approach to Speech Recognition

In the Acoustic-Phonetic approach, it is assumed that there are finite different phonetic units in spoken language. The phonetic units are matched with little time segments according to the acoustic properties of speech. However the acoustic properties are depend on either the speaker or neighboring phonetic unit. Acoustic-Phonetic approach has two main steps. The first step is segmentation and labeling. In this step, the speech signal is divided into short segments and corresponding phonetic units are labeled. And at the second step, the utterance of phonetic units, i.e. utterance of phonemes is converted into a valid word.

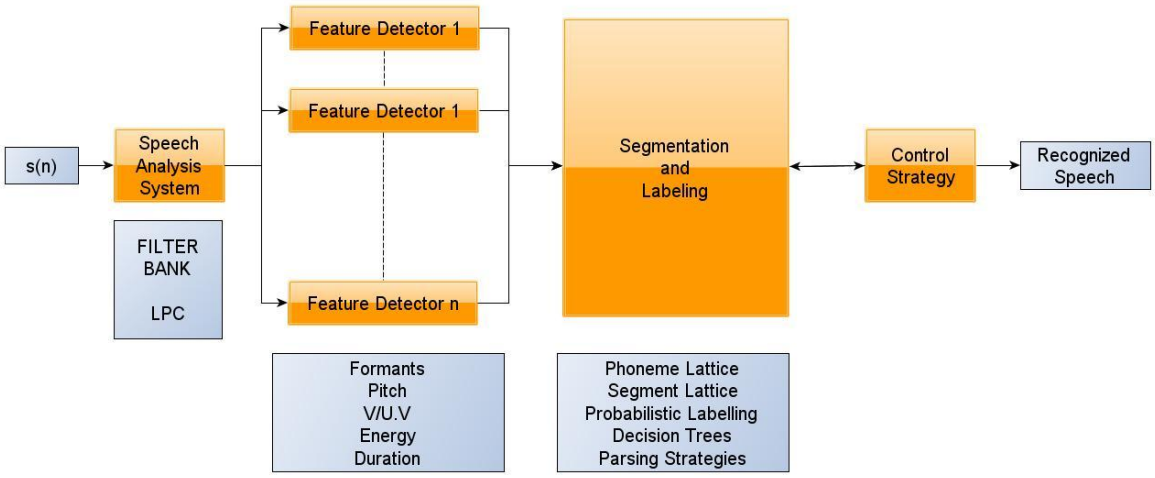


Figure 4.2 Block diagram of acoustic-phonetic speech recognition system [6].

As shown at the Figure 4.2 there are four main steps of this approach such as speech analysis, feature detection, segmentation and labeling, and finally control strategy. In the first step, by using either LPC analysis or other filter bank approaches such as MFCC, the speech signal is converted into a set of vectors. Where a short-time analysis of speech is used and for each frame a set of vectors is obtained. In the second step, the set of vectors which are extracted at the first step is converted into features such as formant frequencies, pitch frequencies, voiced and unvoiced parts, duration and energy features in general. In this step one may use more feature sets. In the third step, the system tries to find stable regions and match them with phonetic units and finally at the last step the utterance of phonetic units is compared with the words at the lexicon in order to find the best match.

The third step of the Acoustic-Phonetic approach is the most important part for a successful recognition. This step uses the features which were extracted at the previous part. In [6] Acoustic-Phonetic vowel classifier (Figure 4.3) and speech sound classifier (Figure 4.4) has shown.

In Figure 4.3 there is a 4-level binary classifier, where at each step different features of the vowel are used in order to classify. In the first step, the first formant frequency of the vowel is compared with average first formant frequency of all vowels, which is a threshold. If the

first formant frequency is high, the classifier decides that it is a compact vowel, else it is a diffuse vowel. In the next step, second formant frequencies are considered. For high F_2 it is an acute vowel, else it is a grave vowel. In the third step, the duration features are considered. If the duration is long, then the vowel is classified as long vowel, else it is a short vowel and finally the sum of first and second formant frequencies are compared with a threshold. If the sum of first two formant frequencies exceeds the threshold then the vowel classified as a flat vowel, else it is classified as a plain vowel.

Figure 4.3 shows the vowel classifier, however we should classify all of the 40 phonetic units in the case of working on English spoken language, or 29 phonetic units in the case of working on Turkish spoken language. Figure 4.4 shows more comprehensive classifier.

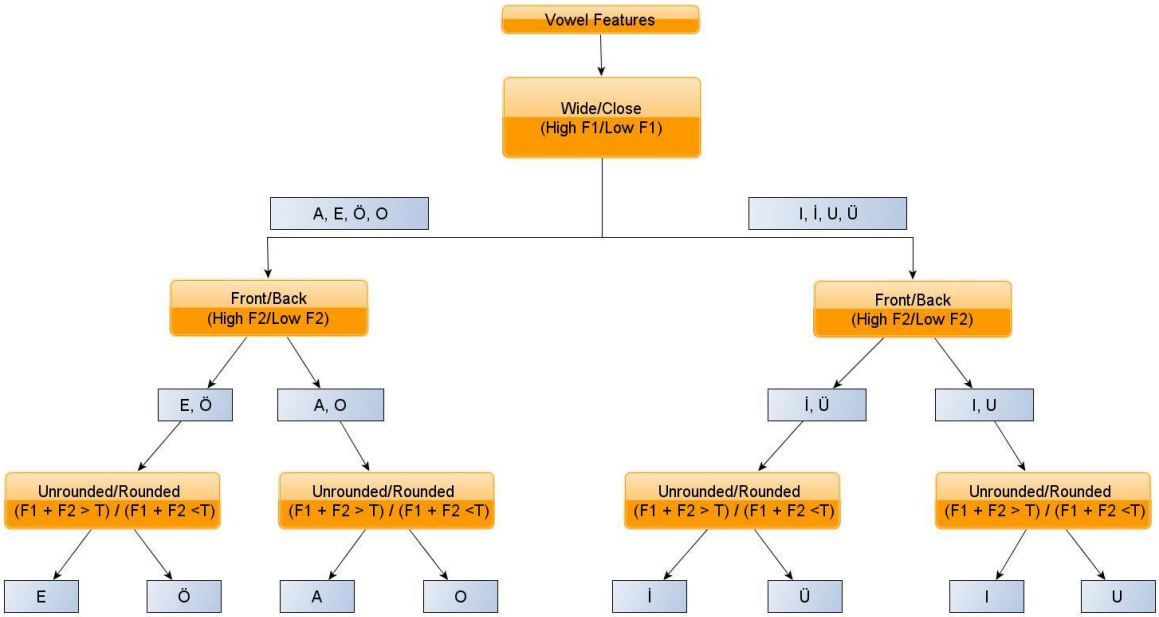


Figure 4.3 Acoustic-Phonetic vowel classifier for Turkish spoken language.

The first decision is if the input speech features are correspond either a sound or silence. If it is a sound, then the decision of if it is a voiced or unvoiced sound is given. Then by comparing the preceded frames features and the frequencies the final decision is given.

Note that, if the classifier decides that the segment of the signal is a vowel, then vowel classifier is used. Hence, the Acoustic-Phonetic vowel classifier which is shown at Figure 4.3 is a part of Figure 4.4.

There are several disadvantages of using Acoustic-Phonetic approach. Firstly, it requires extensive knowledge of acoustic properties of phonetic units. In addition the speaker should speak very clearly. The system can give wrong decisions if a few phonemes are spoken wrongly. Secondly the choice of features is mostly made ad hoc considerations, so choice of features is not optimal. Hence thirdly the design of sound classifiers is not optimal [6].

4.2.2 Pattern Recognition Approach to Speech Recognition

In pattern recognition approach [6], trained speech patterns are compared with the input speech signal. This method has two main steps such as training speech patterns and recognition of pattern by pattern comparison. In order to use this approach properly, enough versions of patterns should be trained. In recognition side, if enough versions of patterns are recognized the system will able to process the input speech, for instance consider the Isolated Digit Recognition task. In the training procedure, the words “zero, one ,…, nine” will be trained. Then if the input speech is the utterance of spoken numbers, for example “five four zero six” the recognizer side will recognize the words separately and combine them at the output. In other words, the machine learns which acoustic properties of speech class are reliable and repeatable across all training tokens of the pattern.

There are several advantages of using Pattern Recognition approach such as its simplicity, robustness and high performance. Furthermore it is easy to expand the vocabulary, and use different users and different feature sets.

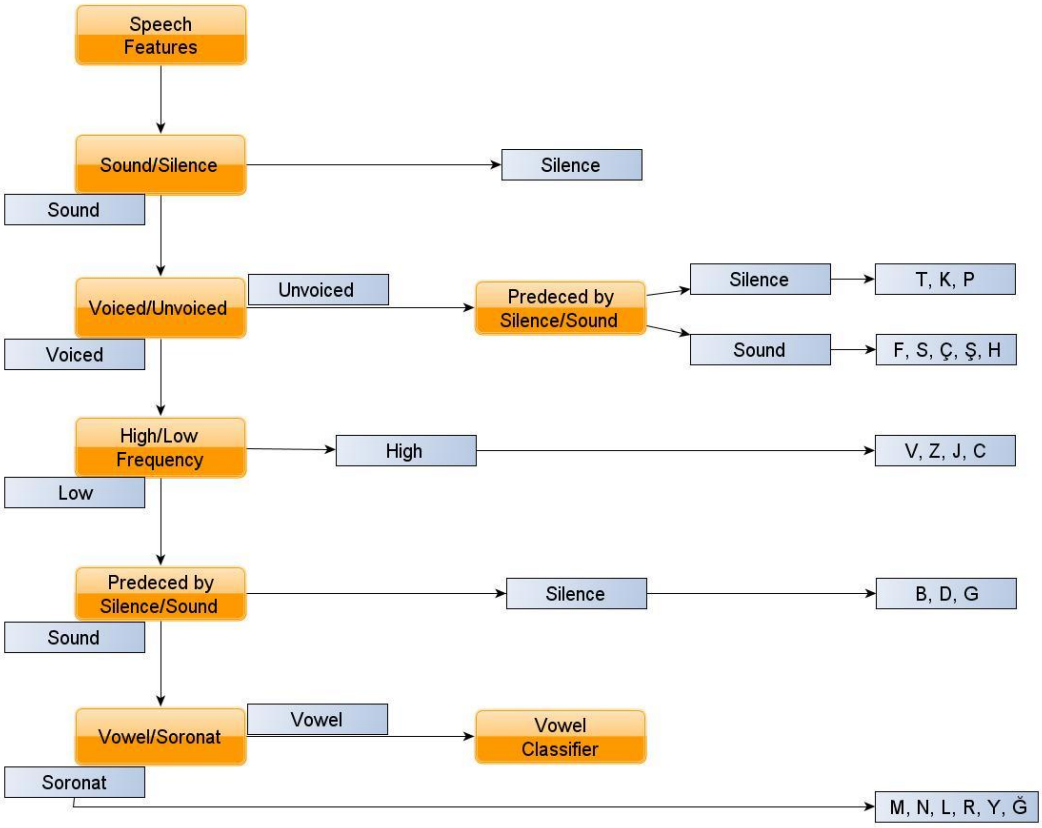


Figure 4.4 Binary tree speech sound classifier for Turkish spoken language [6].

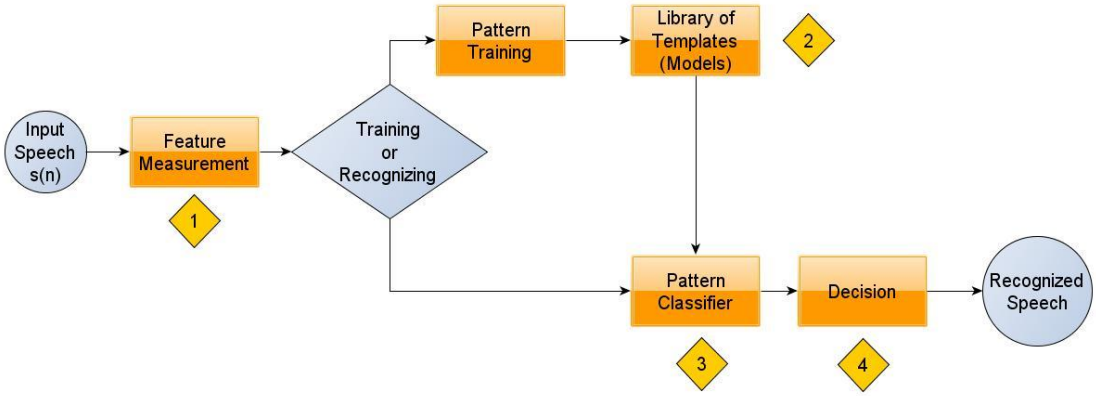


Figure 4.5 Block diagram of pattern-recognition speech recognizer [6].

Figure 4.5 shows the block diagram of pattern-recognition approach to speech recognition. There are four main steps of that approach which are analysis, pattern training, pattern classification and decision. In the first block, by using short-time analysis methods the feature measurements are performed by using spectral analysis techniques such as filter bank approach, linear predictive coding and Discrete Fourier Transform. Then there is a switch which represents if the system is at either training or classification mode. As usual firstly the system should be trained. So the second step is the pattern training block. In this step templates for different speech sounds are generated and stored. The third step is pattern classification, where the switch addresses to the classification block. In that step the feature measures which are extracted at the first step is compared with the templates, i.e. sound models in the library. And finally by using local distance measures such as dynamic time warping algorithm, the final decision about the unknown pattern is given.

The size of training patterns, speaking environment and the speech transmission channel affects the performance of the system. Furthermore the system is not affected by different vocabularies or specific words. But increasing the size of word library in the system causes additional computational load. However syntactic and semantic knowledge of the system reduces the computational load and improves the accuracy.

4.2.3 Artificial Intelligence Approach to Speech Recognition

The Artificial Intelligence Approach is a combination of Acoustic-Phonetic approach and Pattern Recognition Approach. The basic idea is to compile and incorporate knowledge from a variety of several knowledge sources and relate them with the problem at hand. The knowledge sources are acoustic knowledge, which is obtained by spectral measures, lexical knowledge, which is the dictionary of words and their corresponding phoneme utterance, the syntactic knowledge, which controls if the combination of words are grammatically correct, the semantic knowledge, which represents understanding the related task of the sentence or the couple of the recognized words and finally pragmatic knowledge which is the ability to do corrections if any word makes the word utterance semantically

meaningless. The Neural Network approaches are being used in the training of such knowledge sources which are explained in section 4.3.4.

4.2.4 Neural Networks

The general structure of a Neural Network is that, there are several nodes which are connected to each other, and according to the task to be implemented, there are several weights assigned to the paths which connects the nodes. For instance consider human neural system and the task is to shake our hand. Our brain sends some commands to our hands and fingers by using the most efficient path on whole Neural Network of the body. The fundamental element of a Neural Network is shown at Figure 4.6, where we can imagine whole Neural Network as the combinations of fundamental element of the Neural Network. The Figure 4.6 has modeled at Equation 4.1. Where i = 1... N represents the index of inputs x_i, w_i represents the weight of the path $x_i \rightarrow$y, ∅ represents the internal threshold and f(.) represents either a linear or non-linear function.

$$y = f\left[\sum_{i=1}^{N} w_i x_i - \emptyset\right] \tag{4.1}$$

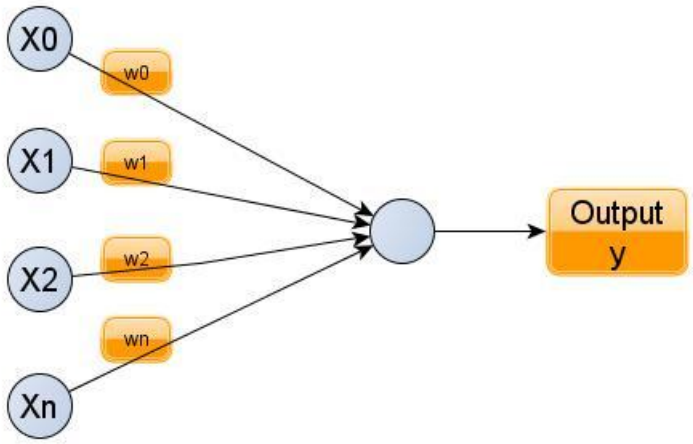


Figure 4.6 Fundamental element of a Neural Network [6]

There are three kinds of Neural Network topologies such as Single/Multilayer Topologies, Hopfield or Re-current Networks and Kohonen or Self-Organizing Networks.

In the first topology which is Single/Multilayer Topologies, in single layer topology N inputs are connected directly to M outputs, however in Multilayer Topology there are some hidden layers between N inputs and M outputs. Each layer, i.e. level of connections is related with some decisions. In the second topology which is Re-current Networks, each computational element includes both inputs and outputs. Consider Equation 4.2. The index "i" represents the index of nodes and index "j" represents the inputs coming through to each nodes whose index represented by "i", and the "t" represents the time instance and w_{ij} represents the weight of the connection to the node indexed by "i".

$$y_i(t) = f\left[x_i(t) + \sum_j w_{ij} y_i(t-1) - \emptyset\right] \tag{4.2}$$

4.3 Complexity of the ASR System

There are several factors which affect the complexity of the speech recognition system such as; number of the speakers to be recognized, the size of the vocabulary, if the speaker speaks continuously such as speaker of a broadcast news channel or discrete units such as specific commands, the acoustic confusability between the words which has similar pronunciations, the speaking environment of the recognized speaker and the linguistic knowledge built into the recognizer. Waibel and Lee (1990) have called them as "dimensions of difficulty" in speech recognition [27].

4.3.1 Speaker-Dependent and Speaker-Independent Recognition Systems

Using either speaker-dependent or speaker-independent recognition systems has several advantages and disadvantages when they are compared.

The strength of using speaker-dependent system instead of speaker-independent system is, because of the system is used by the speaker or speakers who are trained the system, the performance of speaker-dependent system is higher than speaker-independent systems. However, for each new speaker, the system should be re-trained by the new speaker. Those systems are preferred in security systems.

The strength of using speaker-independent system instead of using speaker-dependent system is that, the recognition system could be used by the speakers which are not in the group of speakers who have trained the system. Hence those systems could be accessible to the public such as call centers. However, speaker-independent systems performance is poor when it is compared with speaker-dependent recognizing systems.

4.3.2 Vocabulary Size

The vocabulary size is one of the factors that affect the efficiency and the performance of the recognition systems. As usual, increasing the size of the vocabulary will cause degrease on the efficiency and performance of the recognition system. According to [27] some speech researchers have estimated the difficulty of the recognition problem increased logarithmically with the size of the vocabulary. Furthermore increasing the size of vocabulary requires more memory, because the size of the lexicon should be increased.

According to the vocabulary size, speech recognition systems has basically divided into several classes such as, x-small vocabulary systems where the number of words are vary between 1-99 words, small vocabulary systems whose size is varies between 100-999 words, medium sized systems whose size varies between 1K-100K words, large and x-large sized systems whose sizes are more than 100K and 1000K words respectively.

The purpose of the recognition system generally defines approximate size of the system in general. For instance, if we want to design a speech-recognition system which will recognize only specific words or commands, then we will expect that the size of that system

will small. On the other hand, in our work we are interested in speech recognition system for Turkish broadcast news; in addition those are continuous speech, our lexicon should include all of the used words in Turkish language and additionally some special words such as foreign names and most widely used foreign words.

4.3.3 Isolated-Word and Continuous-Speech Recognition Systems

The Isolated-Word Recognition (IWR) and Continuous-Speech Recognition (CSR) models are used according to the structure of the speech. In the first case, the speech could be combination of utterances of speech units and generally those are word utterances and there are at least 200 milliseconds duration between two words. In that case IWR systems are used.

In the second case, the speaker is speaking naturally, so there are no duration rules such as at least 200 milliseconds pauses after each word. Furthermore, the speaker can pronounce some phonetic units different, where the pronunciations of phonemes were depended on neighbor sounds. In addition two separate words could be combined. For instance in Turkish language instead of speaking “geçen-sil-akşam”, where “sil” represents the silence between two words, it could be spoken as “geçenakşam”. Hence the recognizer should be able to handle the problems of unknown temporal boundaries in acoustical signal, coarticulatory effects and sloppy articulation.

In both IWR and CSR approaches, end point detection is essential. The goal of end point detection is to separate background noise and speech. Basically short-time zero crossing rate and short-time energy measures are performed at the first step. Then thresholds for both zero crossing rates and energy measures are defined for speech and non-speech regions. It is similar with detecting voiced and unvoiced parts of a speech, hence high energy and low zero crossing rates will be occurred at speech regions, and low energy, high zero crossing rates will be occur in non-speech regions.

4.3.4 Linguistic Constraints of Speech Recognition Systems

The linguistic constraints are defined the rules of how basic language units such as phones, phonemes, syllables and words will be concatenated in order to recognize a meaningful message by the recognizer. The complexity of linguistic constraints is also known as the perplexity of the ASR system. Adding more constrained rules to spoken language recognizer system will reduce the freedom of the speaker. Hence the perplexity measures the limitations of the speaker by the system. The perplexity of a speech recognizer will be analyzed more detailed at Chapter 5.

Peirce's model of language (Hartstone and Weirs, 1935) as described by Rabiner and Levinson (1981) includes four levels of the natural language code, which are symbolic, grammatical (syntax), semantic and pragmatic levels, which are shown at Figure 4.7[27].

The symbolic level is the most fundamental unit of a language, where for a spoken language we can assign either phonemes or words as symbols, and for written language we can assign letters as the symbols. If we model symbols as the phonemes, we will observe that they are highly correlated with the spectral properties of the signal. (See Chapter 3 for more detail.)

The grammatical level (also referred as syntax) of a language determines how symbols should be combined to obtain a meaningful message. If we think the symbols as the phonemes or letters, the grammatical component controls if the concatenated version of that symbols forms a valid word. That is the lexical sub-level of the grammatical component. Then in the next sub-level, the syntactic level controls if the concatenated utterance of recognized words forms a grammatically correct sentence.

The further levels are semantic and pragmatic levels, where semantic level controls if the recognized message is meaningful, and pragmatic level discerns various meanings of the output of semantic level and estimates the correct meaning.

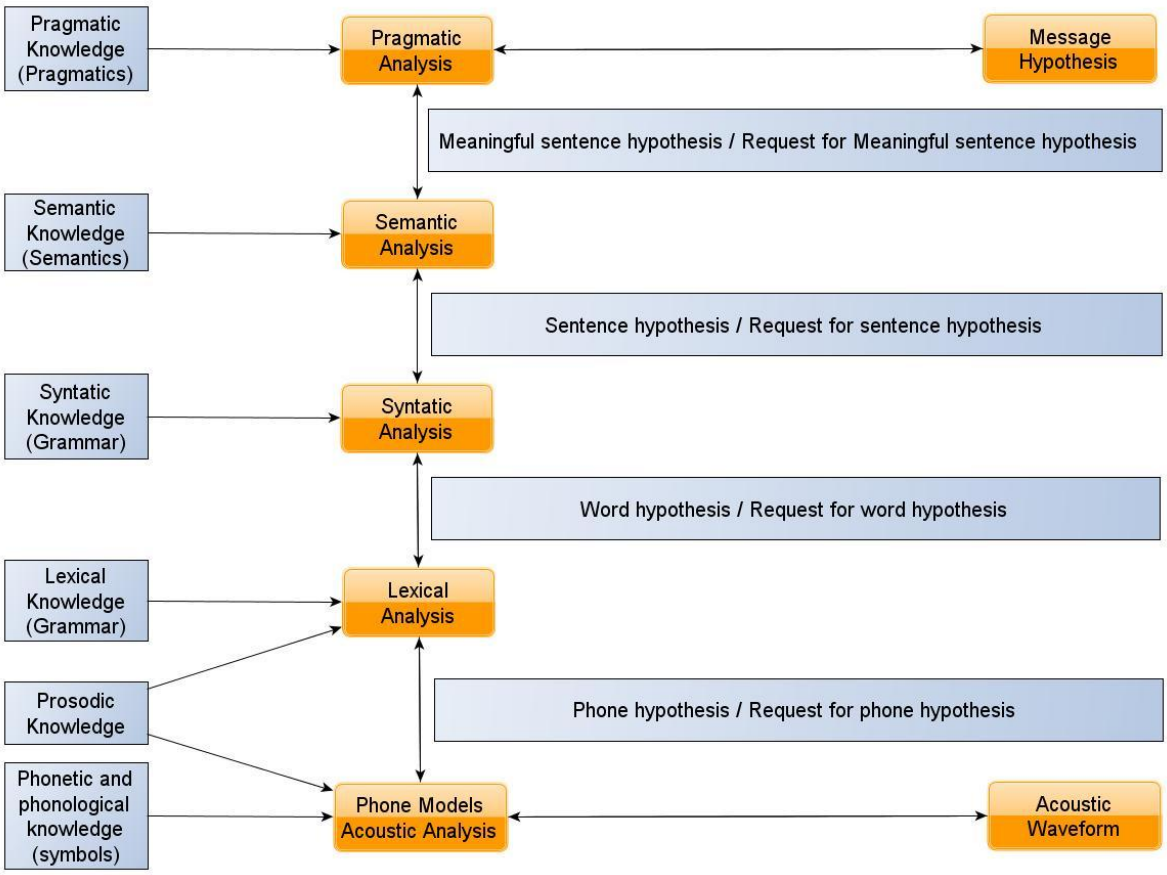


Figure 4.7 Block diagram of a general speech recognizer showing the acoustic and linguistic processors [27].

4.3.5 Acoustic Ambiguity and Confusability of Speech Recognizing System

In the spoken language there may be some similar pronounced words such as similarity between parts of the word, where confusability referred to this situation or same pronunciations for different words, where this situation is referred as acoustically ambiguous words. For instance the words “gelmek” (to come) and “gitmek” (to go) has similar ending so they may be confused. On the other hand the words “gül” (rose) and “kül” (ashes) are acoustically ambiguous words. The corrections of similar examples could be performed at pragmatic level of the language model according to the estimated meaning of the recognized message.

4.3.6 The effects of Environment to Speech Recognition System

The environment is important in speech recognition systems because the background noise which made in the environment of the speaker location affects the performance of the recognizer. However end-detection techniques which are introduced at IWR and CSR approaches, handles those problems. As usual, the performance of the recognizer increases in the case of reduced background noise.

4.4 Building a Speech Recognition System

There are four main steps of building a speech recognition system such as choosing the recognition task, choosing the feature set, training acoustic and language models and evaluating the performance of the speech recognition system.

In our work, the recognition task is to recognize words and sentences with boundaries in Turkish Broadcast news. The lexicon (dictionary) is used which is a vocabulary list includes the written word with phonetic utterances, and the syntax of the language is used as language model. In further applications such as topic segmentation and summarizing semantics and pragmatics should be also used.

The feature set, which are Mel Frequency Cepstral Coefficients (MFCC) are used in our ASR developed by using Hidden Markov Toolkit (HTK) in order to recognize the spoken words and perform forced alignment operation (see chapter 5) in the first step. The performance evaluation of the first system is given at section 4.5. At the second step prosodic features (see chapter 6) are used in order to train models which detects sentence boundaries (see chapter 7).

And finally the performance of the overall system corresponding several feature sets on trained model are evaluated in terms of F-measure score and Nist error rate (see chapter 7 and 8).

The recognition part where pattern classification and decision operations are done is the heart of the overall ASR system. A brief introduction to the Bayesian formulation of the ASR problem is given in [2].

4.4.1 Bayesian Formulation of the ASR Problem

The goal is to recognize the spoken words and concatenate them in order to recognize the spoken message. Hence the problem is modeled as a statistical decision problem which is to find the word string in the spoken language $\widehat{W}$, that maximizes the *a posteriori* probability, *P(W|X)*, of that string, given the sequence of feature vectors *X* as shown at Equation 4.3.

$$\widehat{W} = \arg\max_{W} P(W|X) \tag{4.3}$$

In Equation 4.3, the X represents the acoustic features of the spoken message. In Chapter 3, the short-time analysis of speech processing in time and frequency domain has introduced. As it mentioned, the speech has divided into little time segments i.e., frames. Hence the X is combination of feature vectors of corresponding frames as shown at Equation 4.4.

$$X = [\mathbf{X_1 X_2}, \dots, \quad \mathbf{X_T}] \tag{4.4}$$

In each of $\mathbf{X_t}$vectors includes 12 MFCC features, energy of the frame, 12 delta-mfcc values and delta-energy value and also acceleration values of the corresponding frame. (See chapter 5 for details) The W represents the optimal decoded word string such as $W = \{W_1 W_2, \dots, W_M\}$ where it is assumed that there are M words in the decoded message.

By using the Bayes' rule, we can re-write Equation 4.3 as

$$\widehat{W} = \arg\max_{W} \frac{P(X|W)P(W)}{P(X)} \tag{4.5}$$

Where P(W) represents the *a priori* probability of the word sequence W, $P(X|W)$ represents the likelihood that the word string W, produced the feature vector X, and finally P(X) represents the *a priori* probability of the feature vector. While P(X) is independent of the word sequence W being optimized we can discard P(X) in Equation 4.5 and we can re-write $\widehat{W}$ as shown at Equation 4.6.

$$\widehat{W} = \arg\max_{W} P_A(X|W)P_L(W) \tag{4.6}$$

Where the term $P(X|W)$ represents the acoustic model and shown as $P_A(X|W)$ and the term P(W) represents the language model and shown as $P_L(W)$ in equation 4.6. As shown in Equation 4.6 there are three main steps of the recognition and decoding process such as the first step is to compute the probability associated acoustic properties of the speech message with $P_A(X|W)$, in the second step $P_L(W)$ is computed which is the probability of word sequence associated with the language model of the spoken language and finally $\arg\max_W(.)$ operation searches through all possible valid word utterances in order to find the maximum likelihood sentence.

In Chapter 5, Hidden Markov Modeling, The Viterbi Algorithm, Acoustic and Language Modeling and the search problem which is the process of finding the maximum likelihood sentence will be explained.

4.5 Performance Evaluation of Speech Recognizers

The performance of a speech recognition system depends on how well the spoken words are recognized. Hence; the ratio of total word errors over all of the words, in the recognized speech. There are three types of word errors such as word insertions, word substitutions and word deletions. Word insertion error occurs when a word is recognized even it has not spoken. Generally they are short words and they may occur in long pause durations or hesitations and self-corrections of the speaker.

The word substitution error is occurred when the spoken words and the corresponding recognized words are different. This type of error generally occurs when those two words has similar soundings. And finally word deletion error is occurs when the spoken word is not recognized by the recognizer. This type of error may occur when two words are pronounced as a single word.

The calculation of word error rate (WER) is shown at Equation 4.8, Where N_I represents number of word insertions, N_S represents number of word substitutions and $_{ND}$ represents number of word deletions, and $|W|$ represents number of the words in the spoken word utterance [2].

$$WER = \frac{N_I + N_S + N_D}{|W|} \tag{4.7}$$

Word error rates of several speech recognizers are shown at Table 4.1 [2].

Corpus	Type of Speech	Vocabulary Size	WER
Connected digit strings (TI database) [28]	Spontaneous	11 (0-9, oh)	0.3%
Connected digit strings (AT&T mall recordings) [28]	Spontaneous	11 (0-9, oh)	2.0%
Connected digit strings (AT&T HMIYH © mall recordings) [28]	Conversational	11 (0-9, oh)	5.0%
Resource Management (RM)	Read Speech	1000	2.0%
Airline Travel Information System (ATIS) [29]	Spontaneous	2500	2.5%
North American Business (NAB & WSJ)	Read Text	64,000	6.6%
Broadcast News (CNBC)	Narrated News	210,000	~15%
Switchboard [30]	Telephone Conversation	45,000	~27%
Call-Home	Telephone Conversation	28,000	~35%

Table 4.1 Word Error Rates of several speech recognition systems

Chapter 5

Modeling a Speech Recognizer

5.1 Dynamic Time Warping

The Dynamic Time Warping (DTW) approach is one of the earliest speech recognition approaches. This approach can be applied into simple applications which require relatively straightforward algorithms. The major difference of DTW from pattern recognition algorithms is that, before applying the confidence scoring, the features of the test utterance should aligned temporally with the features of training utterance. [27]

At the beginning the dynamic programming will be introduced, then how DTW approach is used in Isolated Word Recognition (IWR) and Continuous Speech Recognition (CSR) will be shown.

5.1.1 Dynamic Programming

In section 4.2.4 the Neural Networks has introduced, and in Figure 4.6 the fundamental element of a Neural Network has shown. In addition it has mentioned that, whole Neural Network system was the concatenated combinations of the fundamental elements of the Neural Network.

The general framework of dynamic programing is defined in [27] which states that to find the most efficient distance, which is the shortest distance with minimum cost, in between

two nodes which are spreaded in an abstract space. To visualize the concept easily, a two dimensional *i-j* plane is preferred, where i=0,…,I (the horizontal axis) and j=0,…,J (the vertical axis).

The starting node is represented by *(s,t)* and the ending node is represented by *(u,v).* Also the notation *(s,t)* $*$ *(u,v)* represents the path. In addition if *(s,t)=(0,0) and (u,v)=(I,J)* then the path *(s,t)* $*$ *(u,v)* is called a complete path. Moreover there are several distances, in other words costs are assigned to the paths between two adjacent nodes such as (T-type) Transition cost, (N-type) Node cost and (B-type) which is the sum of T-type and N-type.

Firstly, the transition cost is the case when the cost is associated with only to the transition from the current node to adjacent node and the nodes themselves are free. For instance, a businessman travels from the office to the home by using a bus. The cost of total path is charged only at the travelling, in other words transition stage. The formal definition of the transition cost is shown at Equation 5.1. The transition cost $d_T[.]$ is always non negative and any transition $d_T[(i.j)|(0,0)]$ is defined costless, i.e. unit element of the operation which calculate the total cost. For instance it is zero for addition and one for multiplication.

$$d_T[(i_{K-1}, j_{K-1})|(i_K, j_K)] \stackrel{\text{def}}{=} Transition\ Cost \quad (5.1)$$
where (s,t) = (i_{K-1}, j_{K-1}) *and (u,v) =* (i_K, j_K)

Secondly the N-type cost (cost of the node) is the case when the cost is associated with only at the nodes themselves and the transition to that node is costless. For instance, the businessman walks to a hotel after finishing his work at the office. In that case, the cost will be charged only at the hotel, and because of he has walked, there is no cost of travelling. The formal definition of the Node cost is shown at Equation 5.2. The cost of the Node is always defined as non-negative and the cost of initial (0,0) node is always zero or unit element of the operation. (For instance it is one for multiplication)

$$d_N(i,j) \stackrel{\text{def}}{=} The\ cost\ associated\ with\ node\ (i,j) \quad (5.2)$$

Finally the B-type cost is defined as the sum of the transition cost and the cost associated with the node. In B-type costs, both of the T-type and N-type costs could be charged. The formal definition of the B-type cost has shown at Equation 5.3.

$$d[(i_{K-1},j_{K-1})|(i_K,j_K)] = d_B[(i_{K-1},j_{K-1})|(i_K,j_K)] = d_T[(i_{K-1},j_{K-1})|(i_K,j_K)] + d_N(i_K,j_K) \quad (5.3)$$

The distance associated to complete-path is the total cost of (s,n) $*$ (u,v). In Equations 5.3 and 5.4 the total cost is defined in terms of addition and multiplication operations respectively, where "K" is the number of all transitions.

$$D = \sum_{k=1}^{K} d[(i_{K-1},j_{K-1})|(i_K,j_K)] \quad (5.3)$$

$$D = \prod_{k=1}^{K} d[(i_{K-1},j_{K-1})|(i_K,j_K)] \quad (5.4)$$

The best path (s,t) $*$ (u,v) is the case when the total cost *D*is minimized. This path may pass through a node (w,x). In that case the total paths are the concatenated paths of (s,t) $*$ (w,x) and (w,x) $*$ (u,v) which is notated as (s,t) $*$ (w,x) $\oplus$ (w,x) $*$ (u,v) by using Bellman Optimally Principle (BOP) [47], where $\oplus$ is the concatenating symbol. That is also notated as$(s,t)\xrightarrow[*]{(w,x)}(u,v)$. The best path (0,0)$\vec{*}$ (i_K,j_K) is represented by $D_{min}(i,j)$, and this best path may pass through a predecessor node (i_{K-1},j_{K-1}). So $D_{min}[(i_K,j_K)|(i_{K-1},j_{K-1})]$ defines the distance (cost) from (0,0) to (i_K,j_K) which passes through the predecessor node (i_{K-1},j_{K-1}). Hence $D_{min}[(i_K,j_K)|(i_{K-1},j_{K-1})]$ represents the partial path of $(0,0)\xrightarrow[*]{(i_{K-1},j_{K-1})}(i_K,j_k)$ as shown at Equation 5.5.

$$D_{min}[(i_K,j_K)|(i_{K-1},j_{K-1})] = D_{min}(i_{K-1},j_{K-1}) + d[(i_{K-1},j_{K-1})|(i_K,j_K)] \quad (5.5)$$

There may be a set of best-paths corresponding to different predecessor nodes, in that case the optimum partial path is found by taking the minimum of those possible best-paths as shown at Equation 5.6.

$$\begin{aligned} D_{min}(i_k, j_k) &= \min_{(i_{K-1}, j_{K-1})} \{D_{min}[(i_K, j_K)|(i_{K-1}, j_{K-1})]\} \\ &= \min_{(i_{K-1}, j_{K-1})} \{D_{min}(i_{K-1}, j_{K-1}) + d[(i_{K-1}, j_{K-1})|(i_K, j_K)]\} \end{aligned} \tag{5.6}$$

In Equation 5.6 we have only information about the starting node and the ending node and distance (cost) of a path. By using Bellman Optimally Principle notation, we have also information about a node between the starting and the ending nodes about the path which has minimum distance. In most applications it is enough to know the distance of the shortest path, on the other hand in other problems we must not only know the distance of the shortest path but also the path itself with all nodes.

The offered method in [27] is that, once optimal partial path to (i_K, j_K) be found, to perform backtracking (i_K, j_K) to the starting node and each step recording the partial paths to a memory. While recording each step to a memory, the index notation of the each partial path and also the general idea of recording each step into a memory has shown Equation 5.7, where (i_K^*, j_K^*) represents the t[h] index pair on the *K* node long path $(0,0)^{\rightarrow}{}_*(I,J)$ and $\Psi(i_K, j_K)$ represents index of the predecessor node to (i_K, j_K) on $(0,0)\vec{*}\,(i_K, j_K)$and $\Psi(i_K, j_K) = (i_{K-1}, j_{K-1})$.

$$\begin{aligned} (i_K^*, j_K^*) &= (I,J) \\ (i_{K-1}^*, j_{K-1}^*) &= \Psi(I,J) & &= \Psi(i_K^*, j_K^*) \\ (i_{K-2}^*, j_{K-2}^*) &= \Psi\big(\Psi(I,J)\big) & &= \Psi(i_{K-1}^*, j_{K-1}^*) \\ \cdots & \quad \cdots \\ (i_0^*, j_0^*) &= \Psi\big(\Psi \ldots \Psi(I,J)\big) & &= \Psi(i_1^*, j_1^*) = (0,0) \end{aligned} \tag{5.7}$$

In addition, the transitions between states may depend on some defined rules, such that it the whole path corresponds to a word in the vocabulary of the spoken language and each node, in other words state is one of the phonetic units, then the lexical knowledge of the speech recognizer (see figure 4.7) will define the transition rules. Hence, while deciding the transition according those rules; it is also become dependent on past transitions. Hence it becomes a Markov transition.

5.1.2 Dynamic Time Warping Applied to Isolated Word Recognition

5.1.2.1 DTW problem and its solution with DP

The goal of Dynamic Time Warping (DTW) is to perform time-alignment on temporal regions of reference features and test features. The need of doing such operation is that, the difference on speaking speed may be different in test and reference utterances. For instance duration of a word, sub-word and even a phoneme may be different in reference and test records, which are represented by a string of features.

In Isolated Word Recognition systems, both the reference and test words are represented as string of features. Once an incoming unknown test word is recognized as a word in the reference word lists, i.e., best matching scored reference word, the feature strings of features belongs to test word should be mapped onto feature strings of reference word to prevent mistaken word recognitions.

Earlier, linear methods have been used to temporally align utterance of test features to reference features. In this method, the heart of the task was to determine all of the endpoints correctly but even detecting endpoints correctly, this method was not sensitive to differences of phonetic durations inside the words. Later, researchers found more efficient way to perform that alignment. In the new method, instead of using a linear alignment of test and reference feature utterances, a non-linear mapping strategy, in other words, warping has been used by considering energy measures. To perform this operation, Dynamic Programming has been used.

The representation of the test utterance features and reference word utterances have shown below at Equation 5.8, where the string *t*represents the test utterance of features and *r* represents the reference utterance of features [27].

Test features: **t**(1), **t**(2),…,**t**(i),…,**t**(I)
Reference features: **r**(1), **r**(2),…,**r**(j),…,**r**(J) (5.8)

The task is to find a function *j=w(i)* which aligns the test feature utterances onto reference feature utterances. In two dimensional space, if we place the test features **t**(i_k) on horizontal lines and **r**(j_k) on vertical lines, the problem is to find the best global match scored path through (1,1) to (I,J).The cost of matching is defined as positive value, to match **t**(i_k) with **r**(j_k), which is shown at Equation 5.9. The $d_I(.)$ term represents Itakura distance [27]. However in the case of using cepstral coefficients, the cost of matching is evaluated the Euclidean distance between $t(i_k)\ and\ r(j_k)$.

$$d_N(i_k, j_k) = d_I\big(t(i_k), r(j_k)\big) \geq 0 \quad (5.9)$$

The minimum-cost of total path is evaluated by taking the sum of all costs of matching. In this calculation, only N-type costs (see 5.1.1) have been used. So the minimum-cost searching problem has turned into minimum-path search problem. However, if B-type costs are used instead of N-type costs, the transition costs (T-type) will be also included in minimum-cost searching. Hence the distance between the connected nodes will be also considered and warped.

5.1.2.2 DTW search constraints

In previous section, we have stated the DTW problem and we have basically modeled the problem by using Dynamic Programming. However there are several search constraints which should be obeyed in order to obtain legal transitions in the final path such as

endpoint constraints of the path, monotonicity, global path constraints and local path constraints.

Firstly according to the endpoint constraint, the starting point of the path should be at the (t,r)=(1,1) point and the end point of the path should be at the (t,r)=(I,J) point as shown in the Figure 5.1 [27], where the starting and ending points are referring the location of the first and last node.

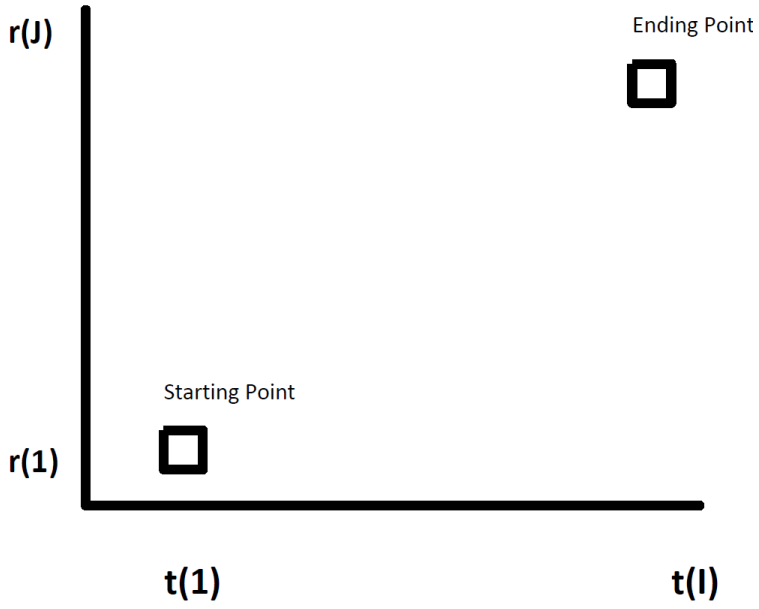


Figure 5.1 The locations of endpoints in the (i,j) plane according to the endpoint constraint.

Recall from section 5.1.1 that the cost of the (i,j) = (0.0) node and transitions from that node was costless, hence defining the starting point as (1,1) will not change the total distance (total cost) of the path.

Secondly, the path should be monotonic. In other words, in the (i,j) plane only the transitions to the north and east are allowed. In other words, it is forbidden to make a transition which will reduce the value of either “i” or “j” is forbidden as shown at Figure 5.2 where the forbidden transitions are shown with red lines on the transition lines.

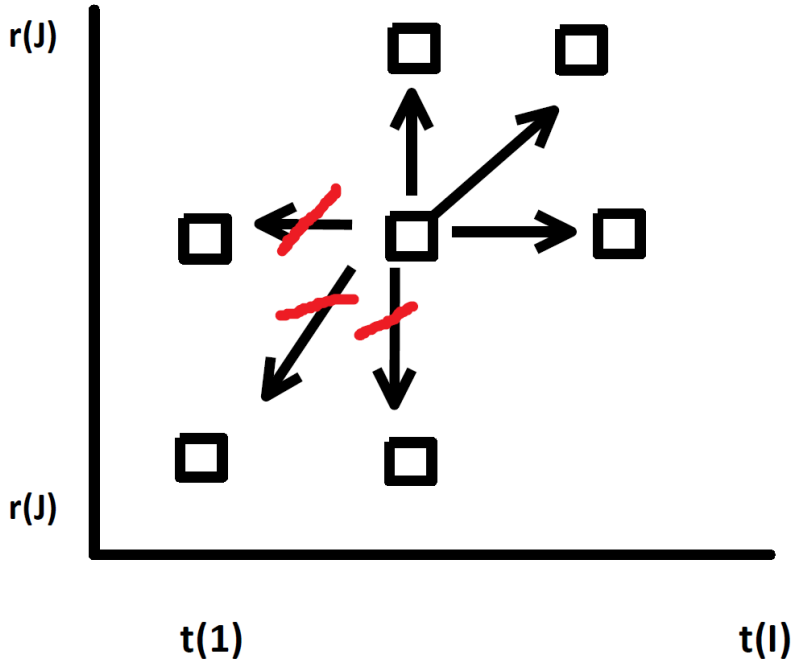


Figure 5.2 Legal and illegal transitions according to the monotonicity constraint [27]

Thirdly the global path constraint specifies the allowable amount of compression and extension in the path, where the compression occurs when the projection of the path to the reference axis is bigger than the projection of the path to the test axis and vice versa for expansion case. Hence global path constraint determines a kind of borders on the (i,j) plane, and the path cannot exceed that area. For instance as shown in the Figure 5.3 that, Itakura global search constraints permitted maximum compression and expansion factor as two. On the other hand, as shown in the Figure 5.4, a simple global search constraint has fixed the width of the path with a constant number W such as $|j_k - i_k| \leq W$.

Finally the local path constraint is related with the connection of each arrival node. All of the arrival nodes will have several constraints. Those constraints are related with the slope of the transition. As shown in the Figure 5.5, several valid transitions to arrival node are shown, and illegal transitions are hidden. The local path constraint limits either extension or compression of small neighborhood preceding (i_k, j_k).

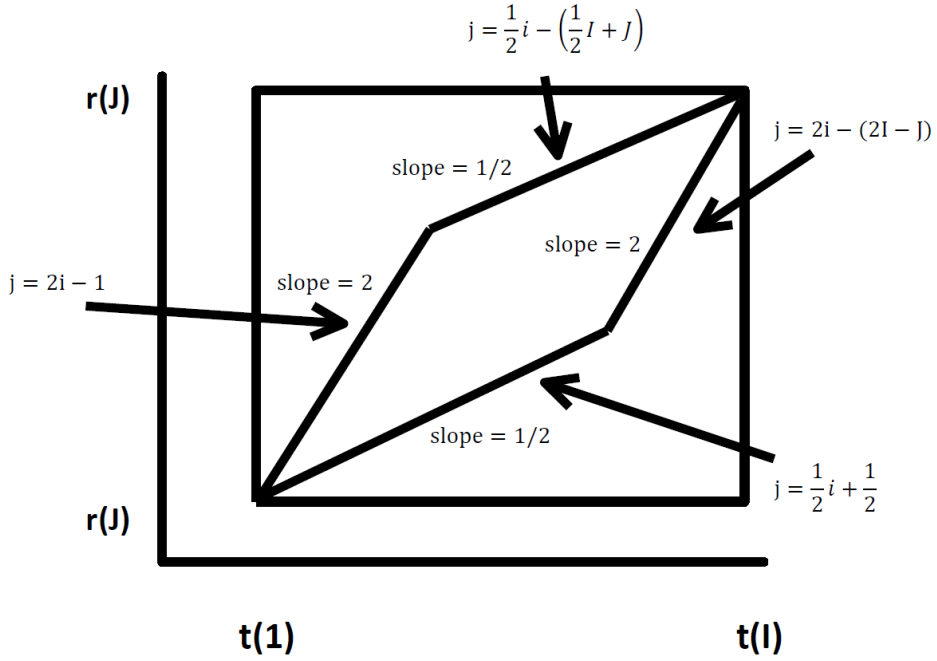


Figure 5.3 The Itakura global path search constraints [27].

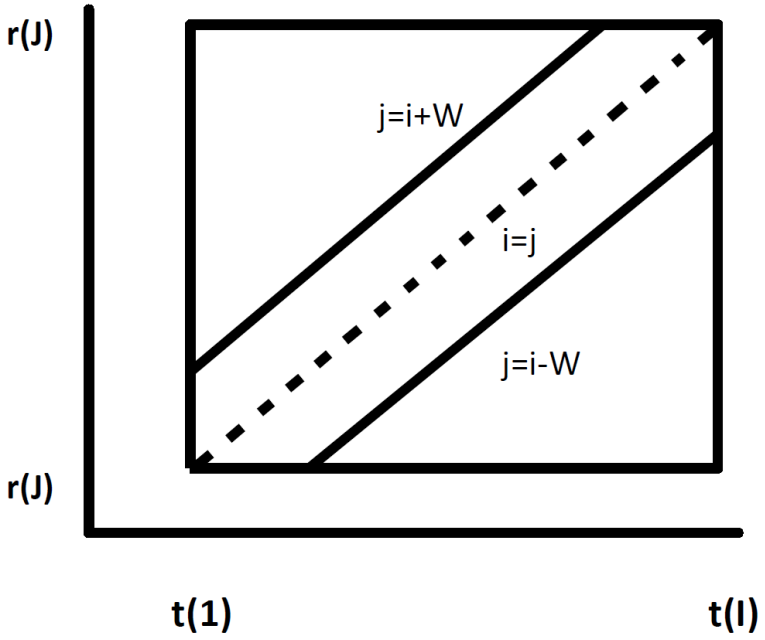


Figure 5.4 A simple global search region with fixed width [27].

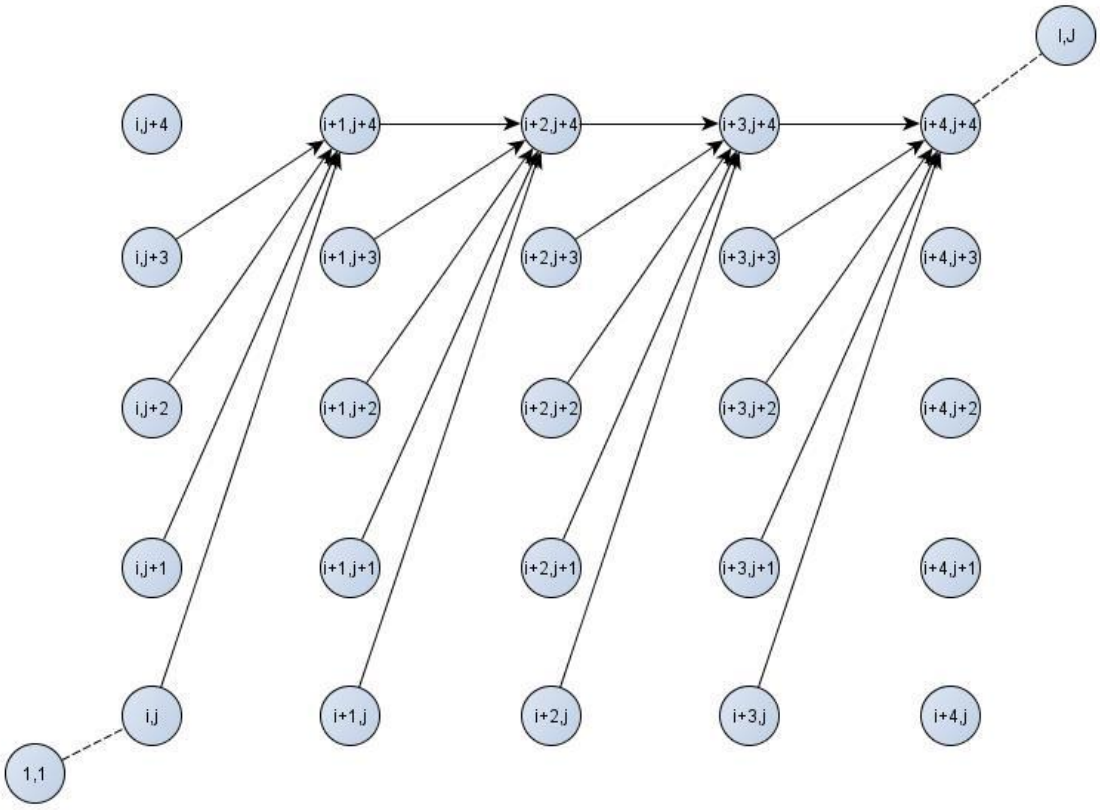


Figure 5.5 Example to a local path constraint, minimum allowed slope is 0.25 and maximum allowed slope is 4 [27].

As shown in the example given at the figure 5.5, the transitions to arrival node$(i_k j_k) = (i + 4, j + 4)$. However in this case, the T-type costs are no more Markovian, because the current transitions are depended on previous transitions. By using Bellman Optimally Principle (BOP) the Markov nature of the transitions can be preserved. In section 5.1.1, we have defined the Bellman Optimally Principle such that a path $(s,t) \xrightarrow[*]{(w,x)} (u,v)$ can be modeled as concatenate of two partial paths such as (s,t) $*$ (w,x) $\oplus$ (w,x) $*$ (u,v), without a requirement that (w,x) is the immediate predecessor node (i.e. adjacent) to node (u,v). In that case, the calculation of the minimum distance of the total path is shown at the Equation 5.10.

$$D_{min}\left[(i_k, j_k)|(i_{k-p}, j_{k-p})\right] = D_{min}\left[(i_{k-p}, j_{k-p})\right] + \hat{d}\left[(i_k, j_k)|(i_{k-p}, j_{k-p})\right]$$

where

$$\hat{d}\left[(i_k, j_k)|(i_{k-p}, j_{k-p})\right] \stackrel{\text{def}}{=} \sum_{m=0}^{p-1} d[(i_{k-m}, j_{k-m})|(i_{k-m-1}, j_{k-m-1})] \quad (5.10)$$

In order to find the optimal path to (i_k, j_k), the minimum over all distance predecessors are taken, which has shown at Equation 5.11.

$$D_{min}(i_k, j_k) = \min_{(i_{k-p}, j_{k-p})}\{D_{min}\left[(i_k, j_k)|(i_{k-p}, j_{k-p})\right]\} \quad (5.11)$$

5.1.3 Dynamic Time Warping Applied to Continuous Speech Recognition

In section 4.3.3 the major differences of input speech to IWR and CSR has explained. When the input speech signal to a CSR system compared with IWR, there will be much larger utterance of speech with varying speaking rate, prosodic variations and articulations. In addition the partitions of the input test speech and corresponding features might be compared with individually trained speech pieces. Hence relax constraints are required but the computational cost becomes unacceptable. For instance for a given task with K reference templates per word, V vocabulary size and contains L words, $O[(KV)^L]$ matches should be performed between reference strings and test utterances. Furthermore the number of required distance computations and DP solutions are $O[\mu IJ]$, where μ represents an integer with a typical value$1/3$, I represents the length of the test, and J represents the length of the reference string. The number of distance computations is $O[\mu(KV)^L I\bar{J}]$ where $\bar{J}$ represents average string length of reference single words.

As described above [27], there is unacceptable computational cost. Hence more efficient algorithms are needed such as Level Building (LB) Algorithm [48], and One-Stage (OS) Algorithm (known also as Bridle Algorithm) [49, 50] The algorithms with complexity and memory requirements are explained below.

5.1.3.1 Level Building (LB) Algorithm

The Level Building (LB) algorithm divides the global search region into levels, where the Itakura global path constraint (see Figure 5.3) is used. For a task with L-words, the reference axis is divided into L-levels as shown at Figure 5.6. It is also assumed that the reference templates have the same length$\bar{J}$, so there are $\bar{J}$ frames in j-axis.

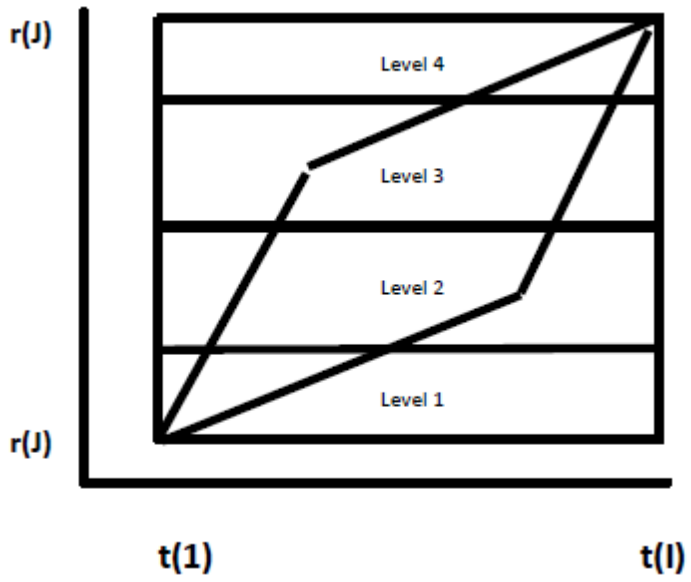


Figure 5.6 A four-level LB algorithm search region [27].

The Algorithm starts to proceed at Level 1 and goes through straightforward to remaining levels. The overview of the algorithm is shown at Table 5.1.

The DTW algorithms are compared by considering the complexity and memory requirements of the algorithm. The operations performed are local distance computations and DP searches. In addition three-level storage array associated with the top boundary of each level. $O(3LI)$ Memory size required for this task. In addition $O(vJ)$ sized local memory is required to memorize the distances and $O(vJ)$ sized global memory is required for backtracking task.

Overview of the LB Algorithm

LEVEL 1

STEP 1
Obtain the reference string $\boldsymbol{r}^v$ for word v.
Where , $\{\boldsymbol{r}^v(1), \boldsymbol{r}^v(2), \dots, \boldsymbol{r}^v(\bar{J})\}$, each $\boldsymbol{r}^v(j)$ corresponds the jth frame.

STEP 2
For $\boldsymbol{r}^1$, Algorithm proceeds to find the best path to each of the possible endings of level 1, considering the local path constraints.

STEP 3
Record the normalized minimum distance for the first found suitable path.
$D^1_{min}(i', J_1) = \frac{D^1_{min}(i,J_1)}{i}$, where the *i* at denominator is the normalizing factor.

STEP 4
Search for alternative path and if an alternative found which has lower cost, replace with the recorded path.

$$Replace\ if\ D^A_{min}(i', J_1) < D^1_{min}(i', J_1)$$

STEP 5

Record the following information for each node $(i, J_1)\ \epsilon\ E_1$, where E_1 represents endpoints of Level-1.

- $v^*(i, J_1)$: Index of the word associated with best path to (i, J_1)
- $D^{v^*}_{min}(i, J_1)$: Cost of the best path (i, J_1)
- (i_k, j_k), starting node of the best path to (i, J_1)

STEP 6

Go to Level 2 and further levels straightforward.

Table 5.1 Overview of the LB algorithm

5.1.3.2 One Stage (OS) Algorithm

In OS algorithm [27,52], instead of dividing the *i-j* plane into several levels, in OS algorithm the goal is to find an optimal path through a DP grid in one stage. However the

2D *i-j* plane is extended to 3D *i-j-v* space, where each word, i.e. template is represented by an *i-j* plane located on v-axis, as shown at Figure 5.7

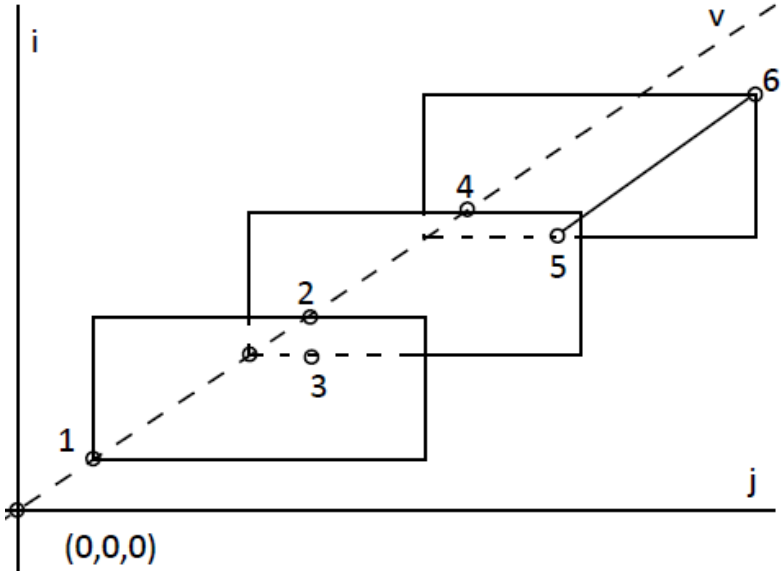


Figure 5.7 3D space for the OS algorithm [27].

The cost of the minimum-cost path is represented by $D_{min}(i,j,v)$, as shown at Equation 5.12.

$$D_{min}(i_k,j_k,v_k) = \min_{(i_{k-p},j_{k-p},v_{k-p})}\{D_{min}[i_k,j_k,v_k]|(i_{k-p},j_{k-p},v_{k-p})\}$$

$$= \min_{(i_{k-p},j_{k-p},v_{k-p})}\{D_{min}(i_{k-p},j_{k-p},v_{k-p}) + \hat{d}[D_{min}[i_k,j_k,v_k]|(i_{k-p},j_{k-p},v_{k-p})]\}$$

Where

$$\hat{d}[D_{min}[i_k,j_k,v_k]|(i_{k-p},j_{k-p},v_{k-p})] \stackrel{\text{def}}{=} \sum_{m=0}^{p-1} d[(i_{k-m},j_{k-m},v_{k-m})|(i_{k-m-1},j_{k-m-1},v_{k-m-1})] \quad (5.12)$$

There are two types of transitions such as within-template transitions and between-template transitions. For instance at the Figure 5.7, the transitions 1 to 2, 3 to 4 and 5 to 6 are within-

template transitions and the transitions 2 to 3 and 4 to 5 are between-template transitions. The cost of within-template transitions are modeled as Type-N costs and the cost of between-template transitions are modeled as Type-T costs.

The formulation of within-template cost is shown at Equation 5.13, where that is the reduced version of Equation 5.12, where the *v* is constant in the I X J_v plane.

$$D_{min}(i_k, j_k, v) = \min_{(i_k, j_k, v)} \{D_{min}[(i_k, j_k, v) | (i_{k-1}, j_{k-1}, v)]\} \tag{5.13}$$

In the calculation of between-template boundary, it is assumed that $(i, 1, v)$ may be preceded by $(i-1, 1, v)$ and $(i-1, J_{v'}, v')$ for any v' including *v'=v*. Hence,

$$\begin{aligned} D_{min}(i,1,v) &= \min \{D_{min}[(i,1,v) | (i-1,i,v)]\} \\ &= \min_{v'} \{D_{min}[(i,1,v) | (i-1, J_{v'}, v')]\} \\ &= \min \{D_{min}[(i-1,1,v)] + d[(i,1,v) | (i-1,1,v)]\} \\ &= \min_{v'} \{D_{min}[(i-1, J_{v'}, v')] + d[(i,1,v) | (i-1, J_{v'}, v')]\} \end{aligned} \tag{5.14}$$

The OS algorithm for connected-word recognition [27] is shown at Table 5.2. The size of required local memory is $O(V\bar{J})$ and the size of the required global memory for back-tracking is $O(I \times \bar{J} \times V)$. The complexity of the algorithm is $V \times I \times \bar{J}$, which represents number of distances and DP searches. As shown above, both memory and complexity of the algorithm is depends the size of the vocabulary. When it is compared with LB algorithm, it has a little more computational cost but on the other hand it is simpler algorithm and requires less memory. In addition at each boundary only one path survives, that property provides optimization over origins [27].

The OS Algorithm

STEP 1: Initialization

For all v $D_{min}(1,1,v) = d[(1,1,v)|(0,0,0)]$

STEP 2: Recursion

For i = 2,…,I

For v = 1,…,V

Focus on parallel grid corresponding to word v. Search grid points (i,j,v) with fixed v

For j = 2,…,J

Determine $D_{min}(i,j,v)$ for any i and for all j>1.

Store $D_{min}(i,j,v)$ into a column vector $\delta(j,v)$ whose size is [J_v 1].

Update $\delta(j,v)$ for each I, which are distances to the top boundaries of words in their location.

In this step, total amount of local storage is $O(V\bar{J})$

Total size of the matrix$\boldsymbol{\delta}$is [$\max_v J_v \times V$]

Required memory for $\boldsymbol{\delta}$is $O(v \times \max_v J_v \times V)$, where v is a small integer.

Update $\boldsymbol{\beta}(j,v)$which holds back-tracking information, where it is enough to know for any path reaches (i,j,v), what was (i',J_v',v').

$\boldsymbol{\beta}(j,v)$ is complement of the local memory vector $\delta(j,v)$

Next j

Next v

Set **w**(i)= $v^*(i) = \arg\min_v D_{min}(i,J_v,v) = \arg\min_v \delta(J_v,v)$

The row vector w(.) whose size is [I X 1] holds the word with best existing path at frame i.

Set **e**(i) = $\boldsymbol{\beta}\left[J_{v^*_{(i)}}, v^*(i)\right]$ which is frame of the preceding exit node on the path, where $\left[i,\ J_{v^*_{(i)}}, v^*(i)\right]$ represents best existing path.

Exit points are stored in an **e**(.) complementing **w**(.)

STEP 3: Backtracking

The optimal sequence of words in reverse order is **w**(J), **w**(**e**(J)),…, **w**(**e**(..**e**(J))) and so on.

Table 5.2 The OS Algorithm

5.2 Hidden Markov Modeling

In section 4.2.4 we have introduced the neural network topology and in section 5.1 we have introduced dynamic time warping (DTW) algorithm and we have also briefly described DTW applied either to IWR and CSR. Hidden Markov Modeling is a stochastic form of DTW and it is used in higher level problems compared with the DTW applications. In this section at first we will do a theoretical introduction to Hidden Markov Modeling and also we will observe the differences between Markov Modeling and Hidden Markov Modeling. And then we will briefly introduce the applications performed by using Hidden Markov Modeling and several algorithms.

5.2.1 Discrete Markov Models

Consider system which is combination of a sequence of N physical events and we are interested in to observe their occurrence in time. To model such system by using Markov Models, several essential information are required such as the number of states and state transition probabilities.

For instance, if Isolated Digit Recognition task is considered (see Table 4.1 [28]), there are 11 physical events, i.e. states, which are the digits zero to nine and an "oh" which means also zero, and there will be 121 state-transition probabilities, where each state has connection with each state. Let us now reduce the number of states for simplicity, where we have only three states N=3, which are the digits, S_1=zero, S_2= one and S_3= two respectively. Figure 5.8 shows a Markov Model for this case [6]. The state transition probabilities are shown on each transition line.

In Figure 5.8, the state transition probabilities are already given. However before going through let define the state transition probabilities in formal. Firstly consider Equation 5.15

$$P[q_t = j | q_{t-1} = i, q_{t-2} = k, \dots] = P[q_t = j | q_{t-1} = i] \quad (5.15)$$

Where, q_t represents the observed state is j in time instance t, and rest of the left-hand side is straightforward. It is shown that, the occurrence of state j in time instance t depends all previous occurred states. In the right-hand side of equation 5.15, the dependence is reduced to first order. Hence the state transition probability a_{ij} defined in Equation 5.16, with the properties of$a_{ij} \geq 0$, for all i and j, and also $\sum_{j=1}^{N} a_{ij} = 1$ for all i.

$$a_{ij} = P[q_t = j | q_{t-1} = i] \tag{5.16}$$

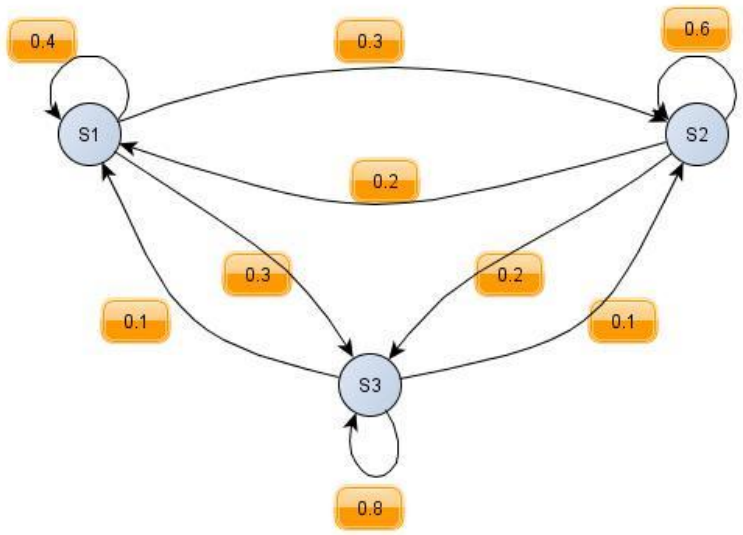


Figure 5.8 Three state ergodic Markov Model

By combining Equation 5.16 and state transitions given at Figure 5.8, for our model, state transition matrix, which includes the probabilities of all transitions are shown at Equation 5.17.

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \tag{5.17}$$

Once the model is given, it is easy to calculate the probability of an observed sequence of spoken digits. For instance the observed sequence $O = [zero,\ zero,\ two,\ one,\ zero,\ two]$ for time instances t = 1 to 5 respectively. Hence

$$\begin{aligned} P(O|Model) = \ & P[S_1 S_1 S_3 S_2 S_1 S_3] \\ = \ & P[S_1|S_1]P[S_1|S_3]P[S_3|S_2]P[S_2|S_1]P[S_1|S_3]P[S_3] \\ = \ & a_{11}a_{13}a_{32}a_{21}a_{13}\pi_3 \end{aligned}$$

where π_i represents initial state probability

$$\pi_i = P[q_1 = i], 1 \leq i \leq N \tag{5.18}$$

5.2.2 Hidden Markov Models (HMM)

In discrete Markov Models, the observations was also the states themselves in other words, each state corresponded to a deterministically observed event [6]. However in Hidden Markov Models, the observation becomes probabilistic function of the state and only the sequence of observations is given and any information about the states is hidden. Two examples to explain the concept of HMM [6, 53] which are, coin toss example and The Urn and Ball model are given below.

In coin toss example, two persons are separated with a wall in a room. There is no information about either the number of coins or the selected coin in the corresponding time instance. The only shared information is the observations which are Heads *H* or Tails *T*. Several assumptions can be performed in order to obtain the whole model. Firstly, it can be assumed that, there is only one coin and whole model is assumed as an discrete Markov Model whose states are the observations either heads or tails. Secondly it can be assumed that there are two coins. Hence the states will be the coins themselves. However the state transitions are unknown at the beginning. Thirdly, similar with second case, it can be assumed that there are three coins, and further assumptions can be made straightforward. The important questions are, which model among several assumptions best matches with actual observations and is the sequence of observations is long enough to specify a complex model.

The Urn and Ball example is similar with coin toss example. In that case, there are several urns which have red, blue, green and yellow colored balls. One urn is selected and one ball is picked from the selected urn, the color of the ball is recorded and the selected ball is released into its urn. In this example, any information about urns is hidden and the only given information is the sequence of observed ball colors.

An HMM, for a sequence of discrete observations, can be modeled by defining the characteristic parameters, which are shown at Table 5.3.

Characterizing a HMM

1. We assume that we have N states, i.e. for the coin toss experiment we assign how many coins are being used and for urn experiment we assign how many urns we have. The states are represented as S_i.
2. Discrete alphabet size M. For the coin toss experiment, the results are head or tail and for the urn example the results are the set of colors of the balls. The observations are represented as $V_i = \{v_1, v_2, \dots, v_M\}$.
3. The state transition matrix "A", which is the combination of Equation 5.16 and 5.17.
4. The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = p[O_t = v_k | q_t = S_j] \ , \ 1 \le j \le N and \ 1 \le k \le M$$

5. The initial state distribution π_i, where

$$\pi_i = p[q_1 = S_i], \ 1 \le i \le N.$$

The combination of probability measures A, B and π is denoted by $\lambda = (A, B, \pi)$

Table 5.3 Characterizing a HMM [6].

After defining the unknown parameters A, B, M, N and π, the generation of a HMM has shown at Table 5.4, by using the observation sequence $O = (o_1 o_2 \dots o_T)$, where each o_i is one of M observations and *T* represents the length of the observation sequence.

1. Choose initial state $q_1 = S_i$according to the initial state distribution π.
2. Set t=1.
3. Choose $O_t = v_k$ according to the symbol probability distribution in stateS_i, i.e., $b_i(k)$.
4. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_i, i.e., a_{ij}.
5. Set t=t+1; return to step 3 if $t < T$; otherwise exit.

Table 5.4 Generation of a HMM [6].

In order to obtain a useful model, three main problems [6] should be considered such as

1. Given the observation sequence $O = (O_1, O_{2,\dots}O_T)$ and the model $\lambda = (A, B, \pi)$ how the probability of the observation sequence $P(O|\lambda)$ computed efficiently? In this task, the goal is to compare the selected model with given observation sequence and score the model. In other words, to compute the probability of given observed sequence produced by the offered model.
2. Given the observation sequence $O = (O_1, O_{2,\dots}O_T)$ and the model $\lambda = (A, B, \pi)$ how a corresponding optimal state sequence $q = (q_1 q_2 \dots q_T)$ is selected which best explains the observation? In this task the goal is to estimate the hidden part of the model, in other words to estimate the optimal state sequence.
3. How the parameters $\lambda = (A, B, \pi)$ should be adjusted to maximize $P(O|\lambda)$. The task is to train the HMM. In other words the parameters of the models are estimated according to the given observations.

For instance in an Isolated Word Recognition system which is implemented by HMM, at first individual word models are built by using solution to problem 3. In second step, to understand the physical meaning of the model states, solution to problem 2 is used. In this stage, each of the word training sequences are segmented into states and properties of spectral vectors are analyzed. And finally an unknown word is recognized by the HMM

model by using the solution to problem. There are several algorithms to solve the three main problems above. Those algorithms are explained below.

5.2.2.1 Forward-Backward Algorithm

The first problem was to calculate probability of observation sequence $P(O|\lambda)$ for a given observation sequence $O = (O_1, O_{2,\dots} O_T)$ and the model $\lambda = (A, B, \pi)$. The straightforward calculation is shown at Equation 5.19.

$$P(\boldsymbol{O}|\lambda)=\sum_{\forall q} P(\boldsymbol{O}|\boldsymbol{q},\lambda)P(\boldsymbol{q}|\lambda) = \sum_{i=1}^{T} \pi_{qi} b_{qi}(o_i) a_{q_i q_{i+1}} b_{qi+1}(o_{i+1}) \quad (5.19)$$

Where, there are N^T state sequences$\boldsymbol{q} = (q_1 q_2 \dots q_T)$**,** the $b_{qi}(o_i)$ represents statistical independent observations, $a_{q_i q_{i+1}}$ represents state transition probabilities. According to Equation 5.19, the system requires $(2T-1)N^T$multiplications and $N^T - 1$ additions. Hence there are too many computations even for a small system. The forward-backward algorithm has developed in order to decrease the complexity.

The forward-backward algorithm is divided into two stages such as forward procedure and backward procedure. In forward procedure for any time instance *t,* $1 \le t \le T-1$the probability of partial observation sequence, which is from the beginning to the time instance, given the model λ is computed. The forward variable is defined in Equation 5.20.

$$\alpha_t(i) = P\left(\boldsymbol{o}_1 \boldsymbol{o}_2 \dots \boldsymbol{o}_{t,} q_t = i|\lambda\right) \quad (5.20)$$

Where the index *t* represents the time instance and the index *i* represent the state. The forward procedure is shown at Table 5.5.

The backward procedure is similar with forward procedure. The difference is that, in forward procedure it was interested in arrival state *j* in time instance *t+1*. In backward procedure it is interested in all possible ways from state *i* at time *t* to the end given the

HMM model. The process is shown at Table 5.6. By using forward-backward processes, N^2T calculations are enough in order to find $P(\boldsymbol{O}|\lambda)$.

STEP 1

Initialization: The joint probability of state i and initial observation o_1 is initialized.

$$\alpha_1(i) = \pi_i b_i(o_1)\,, 1 \leq i \leq N \tag{5.21}$$

STEP 2

Induction: In this step, the ways to reach state j in time instance t+1from N states is calculated. In Equation 5.22, the sum of all paths and transition probabilities to state j at time instance t+1 is multiplied with observation of state j.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \qquad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix} \tag{5.22}$$

STEP 3

Termination: Desired calculation is the sum of terminal forward variables.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{5.23}$$

Where $\alpha_T(i) = P\left(o_1 o_2 \ldots o_t, q_t = i|\lambda\right)$

Table 5.5 The forward procedure [6]

Backward Variable: $\beta_T(i) = P(\boldsymbol{o}_{t+1}\boldsymbol{o}_{t+2} \dots \boldsymbol{o}_T | q_t = i, \lambda)$

STEP 1

Initialization: The $\beta_T(i)$ is defined as 1 for all states.

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{5.24}$$

STEP 2

Induction:

$$\beta_t(i) = \sum_{j=2}^{N} a_{ij} b_j(\boldsymbol{o}_{t+1})\, \beta_{t+1}(j) \qquad \begin{matrix} t = 1, \dots, T-2, T-1 \\ 1 \leq i \leq N \end{matrix} \tag{5.25}$$

Table 5.6 The backward procedure [6]

5.2.2.2 The Viterbi Algorithm

The problem 2 was to find optimal state sequence $\mathbf{q} = (\mathbf{q_1 q_2} \dots \mathbf{q_T})$associated with given observation sequence.One way to perform this task is to choosing states $\boldsymbol{q_t}$ independently for most likely at time instance *t.* The posteriori probability variable is defined at Equation 5.26.

$$\gamma_t(i) = P(q_t = i | \mathbf{O}, \lambda) \tag{5.26}$$

This is the probability of being in state i in time instance t given, the observed sequence and the HMM model. The Equation 5.26 is also written as

$$\gamma_t(i) = P(q_t = i | \mathbf{O}, \lambda) = \frac{P(\boldsymbol{O}, q_t = i | \lambda)}{P(\boldsymbol{O} | \lambda)} = \frac{P(\boldsymbol{O}, q_t = i | \lambda)}{\sum_{i=1}^{N} P(\boldsymbol{O}, q_t = i | \lambda)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{i=1}^{N} \alpha_i(t)\beta_i(t)} \tag{5.27}$$

At the right side of Equation 5.27, it is seen that the posteriori probability variable depends on the forward and backward observation sequences which are explained in forward-backward algorithm. The most likely state is $q_t^* = \arg max[\gamma_i(t)]$, where $1 \leq i \leq N$ and

$1 \leq t \leq T$. However, the probabilities of occurrence of sequences of states are not considered while determining the most likely state.

The goal of Viterbi Algorithm is to find single best state sequence $\boldsymbol{q} = (q_1 q_2 \ldots q_T)$ for the given observation sequence $\boldsymbol{O} = (\boldsymbol{o}_1 \boldsymbol{o}_2 \ldots \boldsymbol{o}_T)$. The algorithm is shown at Table 5.7

5.2.2.3 Baum-Welch Algorithm

Baum-Welch method is used to adjust the parameters of the HMM model. The overview of Baum-Welch algorithm is follows

1. For each parameter, initialize locations which are referred as accumulators.
2. Calculate the forward and backward probabilities for all states j and times t.
3. For each states j and times t, use the probability $\gamma_t(i)$ and current observation vector to update accumulators for that state.
4. To calculate new parameters, use final accumulators.
5. Until finding the maximum $P(\boldsymbol{O}|\lambda)$ repeat above steps.

The initialize stage of the algorithm above is performed by H_{INIT} tool of HTK and the rest is performed by H_{REST} [54].

5.3 Acoustic Modeling

The main goal of acoustic modeling is that to by using audio recordings of speech and corresponding text transcriptions, obtain statistical representations of the sounds which make up a valid word in the spoken language, i.e., word in the used lexicon. Equation 4.3 and 4.6 represents the Bayesian approach to ASR problem.

In Equation 4.6 we have mentioned that the term $P_A(X|W)$ was acoustic properties of the speech message and this was the first main step of the decoding problem. The overview of

the method is that, obtaining the best acoustic HMM models for each sound that composes each word by training the system on an independent and orthographically labeled set of training data.

In the term $P_A(X|W)$ the X represents observed acoustic vector sequences and W represents word sequences. Hence, the goal of acoustic modeling is to assign probabilities to the acoustic realizations of a sequence of acoustic vectors, given the words [2], which has shown at Equation 5.35.

$$P_A(X|W) = P_A(\{\boldsymbol{X}_1\boldsymbol{X}_2, \dots, \boldsymbol{X}_T\}|\{W_1W_2, \dots, W_M\}) \tag{5.35}$$

STEP 1

Initialization: The highest probable single path for the first node and following node is initialized.

$$\delta_1(i) = \pi_i b_i(\boldsymbol{o}_1) \quad , \quad 1 \le i \le N \tag{5.28}$$

$$\psi_1(i) = 0 \tag{5.29}$$

STEP 2

Recursion: In this step, by considering state transition probabilities and the probability of being in a state in corresponding time instance, the most likely node is found.

$$\delta_t(j) = \max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right]b_j(\boldsymbol{o}_t) \quad , \quad 2 \le t \le T,\ 1 \le j \le N \tag{5.30}$$

$$\psi_t(j) = \arg\max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right] \quad , \quad 2 \le t \le T,\ 1 \le j \le N \tag{5.31}$$

STEP 3

Termination: In previous step, the algorithm finds the most probable states belongs to previous state. However, there may be more than one single path. In this step, the path which has the most probability is selected.

$$P^* = \max_{1\le i\le N}[\delta_T(i)] \tag{5.32}$$

$$q_T^* = \arg\max_{1\le i\le N}[\delta_T(i)] \tag{5.33}$$

STEP 4

Path backtracking: From the final state to the initial state, every states are determined in this stage.

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad , \quad t = T-1, T-2, \dots, 1 \tag{5.34}$$

Table 5.7 The Viterbi Algorithm [6]

The observation vector sequences *X* are the Mel Frequency Cepstral Coefficients and the main procedure to extract them is to at first pre-emphasizing the sampled speech signal according to Equation 5.35, then dividing the speech signal into frames to perform short-time analysis and applying hamming windowing (see Chapter 3), then performing Cepstral analysis to obtain the Cepstrum coefficients. In our work we have used HCopy tool of HTK (see section 5.5) to obtain that coefficients.

Before going further to the calculation of Equation 5.35 [2], there are several assumptions are done.

1. Each of the frame *t* has aligned with the word model *i*, and the HMM model state *j*.
2. Each frame is independent from each other.
3. Each $\boldsymbol{X}_t$, i.e. frame of *X* is an unique word associated with state w_j^i

By considering the assumptions above, the Equation 5.35 re-written as

$$P_A(X|W) = \prod_{t=1}^{T} P_A\left(X_t|w_j^i\right) \tag{5.36}$$

Hence the acoustic property of the speech message is expressed by local probabilities given the word and state of the corresponding frame. However to express the local probabilities, further assumptions are done.

1. Each word model also divided into sub-states $S_j, \ \ j = 1,2, \dots, N$
2. Feature vectors *X* is warped onto HMM word sub-states.
3. Left-Right HMM Model is being used, which has shown at Figure 5.9.
4. Within each sub-state in each word, there is a probability density which characterizes the statistical properties of the feature vectors in that state. Those probability densities are learned in the training phase.

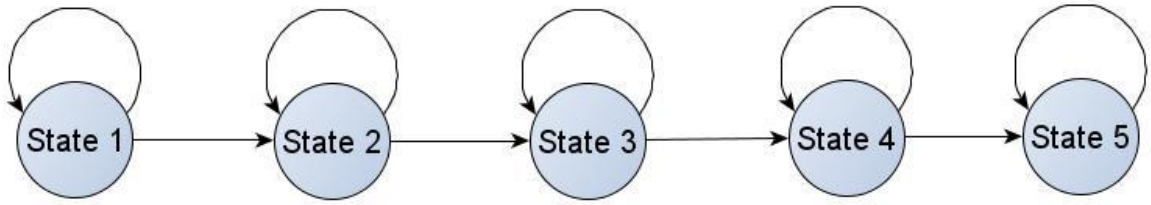


Figure 5.9 Left-right HMM model

Hence the term $P_A(X_t|w_j^i)$ is approximately equals to$b_j^i(X_t)$, where $b_j(X_t)$ represents density function for each corresponding state.

$$b_j(X_t) = \sum_{k=1}^{K} c_{jk} N[X_t, \mu_{jk}, U_{jk}] \tag{5.37}$$

Where $b_j(X_t)$ represents mixture of Gaussian normal densities, K represents the number of mixture components in the density function, c_{jk} represents the weight of the kth mixture component in state j which is always equal or greater than zero, and N represents Gaussian density function with mean μ_{jk} and covariance U_{jk} for each mixture k and state j. In addition there are two constraints which are shown at Equation 5.38-39.

1. $\sum_{k=1}^{K} c_{jk} = 1$, $1 \leq j \leq N$ (5.38)

2. $\int_{-\infty}^{+\infty} b_j(X_t) dX_t = 1, \quad 1 \leq j \leq N$ (5.39)

However in that computation the state transition probabilities a_{ij} are not considered and also it is not specified how to determine the within-word state j in the alignment of word and corresponding features.

In this task, HMM models are generated for each word in the dictionary; however for the tasks which require large vocabulary, there should be many HMMs should be trained. An alternative method is to build acoustic-phonetic models for all phonetic units in the spoken

language. For instance, in English it is enough to build 40 acoustic-phonetic models for each phonemes of the English language, and in Turkish, it is enough to build 29 acoustic-phonetic models, where there are 29 phonemes (also letters) in modern Turkish language. In this type of systems, a word lexicon, i.e. dictionary is used. In this dictionary, all words are listed with corresponding either mono-phones (utterance of corresponding words phonetic units) or tri-phones. For instance for a given Turkish word "kelime", the mono-phone representation is "k-e-l-i-m-e" and the tri-phone representation is "kel-eli-lim-ime".

5.4 Language Modeling

Recall that the second main step of building a speech recognizer was the computation of $P_L(W)$ which was the probability of word sequence associated with the language model of the spoken task. Also In section 4.3.4 we have explained the linguistic constraints of speech recognition systems. The definition of Language Modeling is that to assign probabilities to each possible word strings based on the likelihood of that word string occurred in the task performed. In that point, the task is, for small applications such as digit recognition is to recognize the sequence of the digits, hence the vocabulary will include only the digits. On the other hand in a complex application such as recognizing a spoken language, the vocabulary size is much larger than simple applications, i.e. the size of lexical knowledge is much larger. In addition the language model should also contain syntactic, semantic and pragmatic knowledge depend on the goal of the desired goal of recognition system such as either performing only sentence segmentation or going one step further to perform topic segmentation and summarization.

Developing a speech recognition system for any spoken language requires large vocabulary, and also from the previous discussions, the probability of a word followed by another word depends highly on the structure of the spoken language. In other words, for instance consider a segment of a speech which contains one or two sentence and includes Q words.

$$W = w_1 w_2 \dots w_Q$$

Hence the probability of word sequence associated with the spoken language model will be

$$P_L(W) = (w_1 w_2 \dots w_Q) = P(W_1)P(W_2|W_1)P(W_3|W_1W_2) \dots P(W_Q|W_1W_2 \dots W_{Q-1}) \quad (5.40)$$

In the right side of the Equation 5.40, it becomes impossible to estimate the large terms such as $P(W_Q|W_1W_2 \dots W_{Q-1})$ where there are too many word dependencies. To overcome that problem, N-gram word models are offered. In this technique, instead of calculating such large terms shown above it will be enough to calculate the most effective part, in other words, instead of going back to the initial word in the middle of a long speech, it is enough to go back only N words. Equation 5.41 expresses the idea in a formal way.

$$P(W_Q|W_1W_2 \dots W_{Q-1}) \cong P(W_Q|W_{Q-N+1} \dots W_{Q-1}) \quad (5.41)$$

Hence the Equation 5.41 can be re-written as

$$P_L(W) = \prod_{i=1}^{Q} P(W_i|W_{i-1}W_{i-2} \dots W_{i-N+1}) \quad (5.42)$$

The estimation of conditional probabilities in Equation 5.42 is done applying Equation 5.43

$$\hat{P}(W_i|W_{i-1}, \dots, W_{i-N+1}) = \frac{F(W_i, W_{i-1}, \dots, W_{i-N+1})}{F(W_{i-1}, \dots, W_{i-N+1})} \quad (5.43)$$

Where the function F(.) represents the number of occurrences of the word string given the training corpus. However if the *N* is selected large, the value of F(.) goes to zero hence Jelinek [55] defined N=3 and used trigram word models. The smoothing has done by interpolating trigram, bigram and unigram relative frequencies which shown at Equation 5.44.

$$\hat{P}(w_3|w_1,w_2) = p_1\frac{F(w_1,w_2,w_3)}{F(w_1,w_2)} + p_2\frac{F(w_1,w_2)}{F(w_1)} + p_3\frac{F(w_1)}{\sum_i F(w_i)} \tag{5.44}$$

Where p_1, p_2, p_3non-negative weights and sum of them are 1, and $\sum_i F(w_i)$ is the size of the corpus. The weights depends on $F(w_1,w_2)$ and $F(w_1)$ and they obtained by applying the principle of cross-validation [6, 55].

In language modeling, it is also important to know that how well the language model is represents the context of the speech [6]. In other words does the semantic knowledge of the language model is enough to recognize a speech in any context. This is measured by using source of information. The entropy of any given word sequence is shown at Equation 5.45.

$$H = -\lim_{Q\to\infty}\left(\frac{1}{Q}\right)\left\{\sum P(w_1 w_2 \dots w_Q) log P\{w_1 w_2 \dots w_Q\}\right\} \tag{5.45}$$

Where in the Equation 5.45 the term p(.) defines the probability of the given word string occurs in a language model which has vocabulary size of Q words.

If the sequences of words are considered as independent, then the term $P(w_1 w_2 \dots w_Q)$ becomes equal to $P(w_1)P(w_2), \dots, P(w_Q)$. Hence; equation 5.45 re-written at below, where *w* represents words and *v* the vocabulary.

$$H = -\sum_{\forall w \epsilon v} p(w)\log p(w) \tag{5.46}$$

There may be at total 2^H such word sequences. The entropy of a typical given word sequence which is sufficiently large is computed by using equation 5.47.

$$H = -\left(\frac{1}{Q}\right)\log P(w_1, w_2, \dots, w_Q) \tag{5.47}$$

However it is difficult to calculate the term $P(w_1, w_2, \dots, w_Q)$ in Equation 5.47. Hence the estimate of P(W) from N-gram model calculation is used. Thus

$$H_p = -\frac{1}{Q}\sum_{i=1}^{Q} \log P(w_i|w_{i-1}, w_{i-2}, \dots, w_{i-N+1}) = -\frac{1}{Q}\log \hat{P}(w_1, w_2, \dots, w_Q) \tag{5.48}$$

Where $\hat{P}(w_1, w_2, \dots, w_Q)$ is an estimate of $P(w_1, w_2, \dots, w_Q)$.Similarly H_p is estimated entropy of H and it represents average difficulty or uncertainty in each word based on the language model [6].where if the source is ergodic and as Q goes to infinity, then $H_p \geq H$. The perplexity of the language model is computed by using Equation 5.49.

$$B = 2^{H_p} = \hat{P}(w_1, w_2, \dots, w_Q)^{-1/Q} \tag{5.49}$$

5.5 Hidden Markov Model Toolkit

In speech recognition and several applications Hidden Markov Toolkit (HTK) is used to build Hidden Markov Models (HMMs). There are two main tasks performed by HTK such as training the recognizer and use trained recognizer in order to perform recognition operation. In this section the four main stages of the HTK toolkit which are data preparation, training the recognizer, testing the recognizer and performance analysis of the recognizer will be introduced and the usage of HTK in our work will be explained.

The Hidden Markov Toolkit [54] has developed by researchers in Cambridge University. By login to the system, one can download this toolkit and the referred HTK tutorial. The HTK has sub toolkits, and each toolkit is used for a specific purpose. In our work, I have used HTK Turkish speech recognition system which has developed in Boğaziçi University BUSIM laboratory [44] in UNIX operating system environment. Figure 5.10 shows the architecture of the HTK system.

5.5.1 Data Preparation Stage

In Figure 5.9, at the top the process begins with data preparation step. There are two types of files to be processed. The speech file and the transcriptions of corresponding speech files. At the top of the figure, speech signal processing analysis is performed by the tools HSLab, HCopy, HList and HQuant. The HSLab tool can be used in order to record data. The HCopy is used to parameterize the data. For instance, in our work HCopy is used to obtain Mel Frequency Cepstral Coefficients (MFCC) of the speech signal.

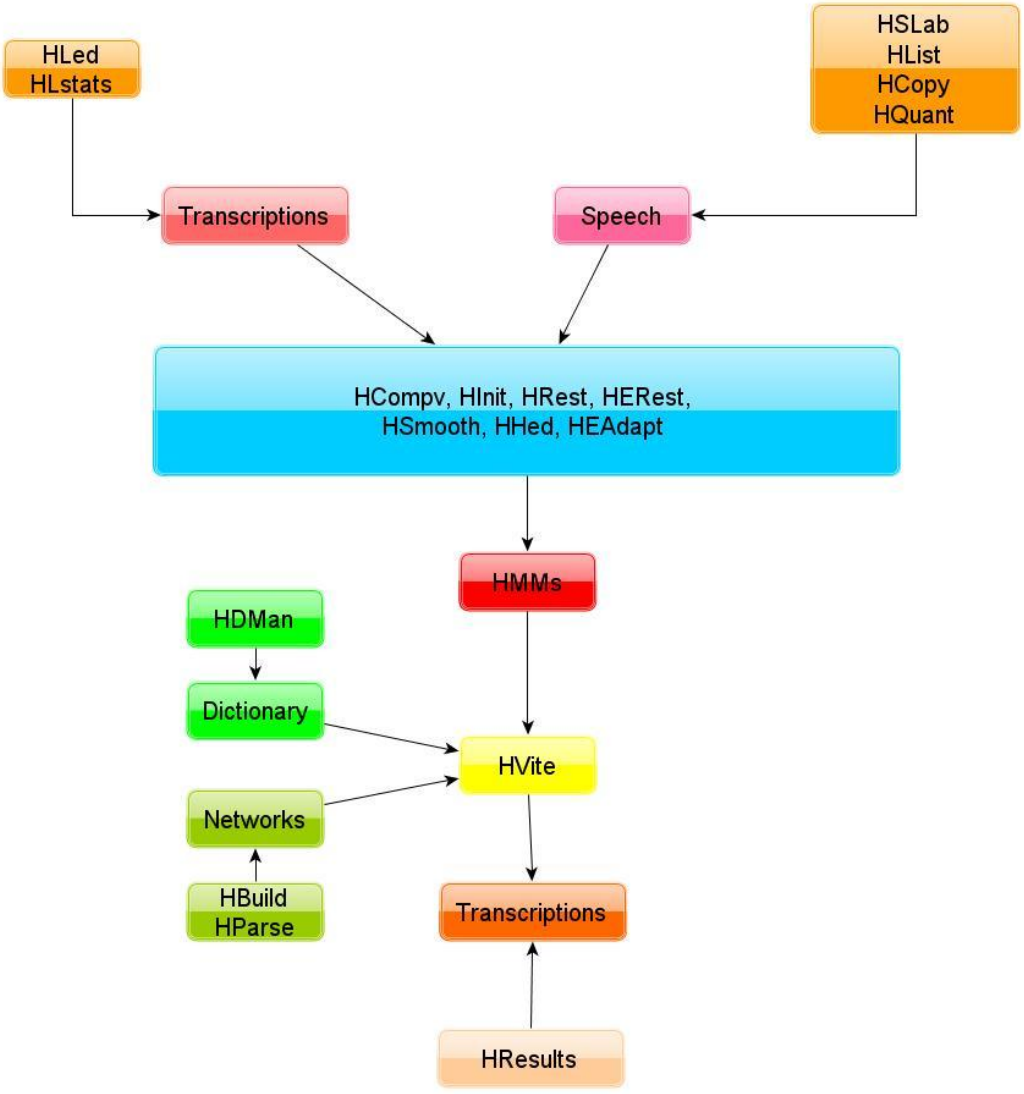


Figure 5.10 Architecture of the HTK processing stages.

HList is used to view the parameterized data obtained by HCopy, where the output of HCopy is a binary file and when it cannot be viewed. The HQuant data is used in order to build VQ codebook. On the other hand at the left-side, the files contains the transcriptions of the corresponding speech are processed by using HLed and HLstats tools. The HLed is works as label editor and constructs the *Master Label Files.* For instance the segment time marks (STM) files are example of Master Label Files, where the timing information of each speech segment is shown at those files. The HLStats is used to display statistics on label files if it is necessary.

In our work, we have used VOA Turkish broadcast news [43] and corresponding segment time marks (STM) files, which has prepared by the BUSIM [44] group. However, because of Prosodic Feature Extraction Tool [31, 35] works speaker based, I have divide the speech data and corresponding STM files into speaker based files. In this stage, we have speaker based speech files and the corresponding transcriptions STM files.

At the first step, by using HCopy tool I have extracted the MFCC of the speech signal. The procedure of extracting MFCC by using HCopy is explained below.

In order to extract MFCC by using HCopy tool, at first step there should be a configuration file. Such configuration file was already prepared at BUSIM [44], at first this configuration file will be explained.

STEP 1: Defining the input/output formats and sampling rates.

SOURCEFORMAT = WAV	# the input source format.
SOURCERATE = 625	# 16KHz sampled input waveform.
TARGETRATE = 100000	# 100 parameters for each output block.
WINDOWSIZE = 250000	# 25msec window.
SOURCEKIND = WAVEFORM	# the type of the source.
TARGETFORMAT = HTK	# the type of the output.

Table 5.8 I/O part of the HCopy configuration file

In Table 5.8; "SOURCEFORMAT" determinates the input waveform format, "SOURCERATE" determines the sample rate of the input waveform file, "TARGETRATE" determinates the sample rate of the output file. "WINDOWSIZE" determinates the window duration while we are dividing the signal into frames. Those parameters are in terms of 100 nanosecond units. So we have divided 16 kHz input speech signal into 25 milliseconds long frames. The "SOURCEKIND" is the input parameter kind where it is defined as waveform. It could be also LPC (Linear Prediction Coefficients) or ANON. The "TARGETFORMAT" is the type of the output.

STEP 2: Performing pre-processes to extract MFCC

ZMEANSOURCE = FALSE	# Removes the DC mean
PREEMCOEF = 0.97	# Pre-emphasizing
USEHAMMING = TRUE	# Hamming windowing

Table 5.9 Pre-processes to extract MFCC

In Table 5.9, the pre-processing steps are shown before extracting MFCC coefficients of the data. The first step is either removing the DC mean or pre-emphasizing the speech signal according to Equation 5.50, where s_n' are the pre-emphasized signal, s_n original signal and *k* is the pre-emphasizing coefficient. After pre-emphasizing the signal, hamming windowing is used, where it has explained in Chapter 3.

$$s_n' = s_n - k s_{n-1} \tag{5.50}$$

STEP 3: Extracting MFCC

TARGETKIND = MFCC_E_D_A	#Energy, delta coefficients and acceleration coefficients will be calculated
DELTAWINDOW = 2	
ACCWINDOW = 2	
ENORMALISE = FALSE	# Normalizing energy values inactive.
SILFLOOR = 50.0	# Removes minimum energy components under the threshold defined in terms of dB
ESCALE = 1.0	# Energy scaling factor
LOFREQ = 300	# Minimum frequency in the speech signal.
HIFREQ = 8000	# Maximum frequency in the speech signal.
NUMCHANS = 24	# 24 triangular Mel filters
NUMCEPS = 12	# Number of Cepstral coefficient at the output for each frame.

Table 5.10 Extracting MFCC

The "TARGETKIND" will be used by HList tool to view extracted MFCC coefficients. The "_E" stands for the energy measure. The energy is computed as the log normal energy as shown in the Equation 5.51.

$$E = log \sum_{n=1}^{N} s_n^2 \tag{5.51}$$

Energy values can be normalized by setting "ENORMALISE = T" which will normalize and give the values in the range of the minimum energy value and 1. This normalization is done by subtracting the maximum energy value and adding 1. We cannot use "ENORMALIZE" in the live audio applications because of the computational complexity. We can also get rid of the minimum energy components by using "SILFLOOR" where we can give a ratio in terms of dB between the maximum and minimum energy. The overall log energy can be also scaled by the "ESCALE" command. After calculating the energy parameter, now we will also put the delta coefficients (first-order regression coefficients) by typing "_D." By using the delta coefficients (the first order differences) our speech

recognition system is enhanced. We have also typed "_A" which are acceleration coefficients (second order regression coefficients). At the output, there will be 12 Cepstral coefficients for each frame, 1 Energy measure, 12 delta Cepstral coefficients, and 1 delta energy coefficient and 12 acceleration Cepstral coefficients and 1 energy acceleration coefficient, hence for each frame there will be 39 parameters. The calculation of the delta coefficients are shown at Equation 5.52.

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{5.52}$$

Where, d_t is delta coefficient computed in terms of the corresponding static coefficients $c_{t+\theta}, c_{t-\theta}$. The parameter Θ represents DELTAWINDOW and ACCWINDOW (Acceleration window), where the computation of acceleration coefficients are similar with computation of delta coefficients.

As shown in Table 2.1, there are 24 filters according to the bark-scale. The locations of the filters are defined by the Equation 2.1. However HTK provides a simple Fourier Transform based filterbank design, which has shown at Equation 5.53 and approximately equal to the resolution in Mel-scale.

$$Mel(f) = 2595 log_{10}\left(1 + \frac{f}{700}\right) \tag{5.53}$$

We have typed "TARGETKIND = MFCC" at the Table 5.10 to obtain Cepstral parameters. The Cepstral parameters are calculated from the log filter bank amplitudes $\{m_j\}$ using Discrete Cosine Transform as shown in the Equation 5.54.

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \tag{5.54}$$

At the Equation 5.38 the N is the number of the filter bank channels, which is set by typing "NUMCHANS = N". The number of output Cepstral coefficients are defined by typing "NUMCEPS = M".

STEP 4: Outputs

```
NATURALREADORDER = T
NATURALWRITERORDER = T
```

Table 5.11 Output configuration of the file

"NATURALREADORDER" and "NATURALWRITEORDER" commands as shown in the Table 5.11 which controls the HTK reading and writing format waveform files.
The following commands are used to extract and view MFCC coefficients.

```
HCopy -C hcopy2.conf inputfile.wav outputfile.mfc

HList -C mfcc.conf -i 39 -h -o inputfile.wav
```

Table 5.12 Extracting and viewing MFCC coefficients by using HTK

The corresponding STM files should be also adjusted. In this stage, while dividing speech signals in speaker based, also the corresponding STM files are divided. And both speech and corresponding STM files are concatenated. By using Perl scripts, we have obtained STM files corresponding single speaker speech records.

5.5.2 Training Stage

In the training stage, the speech and transcriptions are used in order to obtain HMMs. In this stage, the main problem is to build phone models.

STEP 1: Initialization

If there are labeled utterances and corresponding speech signals i.e. forced aligned utterances are given as input into H_{INIT} tool. The H_{INIT} tool creates initial set of models. The time labels of the labeled utterance are used as *bootstrap* data. Then H_{REST} re-estimates the isolated words. The initial set of mean and variance parameters is computed by using segmental k-means procedure. Where at the beginning the parameters are initialized uniformly and further iterations the parameters are defined by using Viterbi alignments. In the re-estimation stage performed by $H_{REST,}$ the segmental k-means procedure is replaced by Baum-Welch re-estimation procedure.

In the case of, the forced aligned data is not given, i.e. there are no time labels in the input data, phone models are initialized identical and have equal means and variances. This initialization is performed by HCompV tool.

STEP 2: Embedded Training

Recall that, the goal of Baum-Welch algorithm was to adjust the HMM parameters. Once the initialization step is performed, HER_{EST} tool performs a single Baum-Welch re-estimation of the whole set of HMM phone models. In addition forward-backward probabilities are used in order to accumulate the statistics of concatenated phone models.

5.5.3 Recognition Stage

The recognition stage is very important for our work. We have used H_{VITE} tool to obtain phoneme based and word based labeled segment time marks, where this kind of data is required in order to extract prosodic features by using the Prosodic Feature Extraction Tool which has explained in detail at Chapter 6. At the beginning there are several inputs to the H_{VITE} tool such as; the master label file (MLF) which contains a list of words in a specified

time segment, the corresponding MFCC vectors of the word utterances in the MLF file, those files were prepared in data preparation stage, a configuration file for the H_{VITE} tool, the monograms, i.e. phones in Turkish Language, Turkish language model which has trained and prepared by BUSIM researchers and a dictionary which includes the utterance of phonemes for each word. The main command to perform forced alignment is shown at Table 5.13

```
HVite -o S -T 1 -C config.hd -a -b "<s>" -m -I test.word.mlf -S test.mfc.list -H htkmono/MODELS -
iout.mlf dictionary htkmono/mono.list
```

Table 5.13 Performing Forced Alignment

At First, “config.hd” is a configuration file which is similar with the configuration file prepared for the H_{COPY} tool. This file is prepared by BUSIM researchers.

At Second, the “<s>” is the used silence model.

At Third, “test.word.mlf” is a master label file, which contains the spoken words in a specified time interval. I have used simple Perl scripts in order to convert the segment time marks (STM) into such master label files (MLF). One example is shown below at Table 5.14

At Fourth, “test.mfc.list” is a list file of the MFCC vectors for each segment time marked speech. This file is also obtained by using simple Perl scripts, and shown at Table 5.14

At fifth, the "MODELS" is the Turkish Language Model developed by BUSIM researchers. The required phoneme and word based networks are found inside this file.

At sixth, the “out.mlf” is the output of the HVite tool, i.e. the forced aligned data, which is shown at Table 5.16.

At seventh, the dictionary is the list of the words with spoken utterance of phonemes, which is shown at Table 5.15.

At eighth, the mono.list includes the monograms, i.e. phones of Turkish language which are shown at Table 3.1.

STEP 1: Part of an STM file. The first and third column is speaker information, second column is a kind of channel information (mono) the fourth and fifth columns are starting times and ending times of the segment in terms of seconds, the sixth column is information about the speaker and the speech environment. We have a male speaker who has spoken in studio "f0" Each following word is placed in a new column, and a new line starts at concatenated segment, where the beginning of each line is bolded. Finally the "." at the end represents silence model between two segments.

AlparslanesmerA 1 AlparslanesmerA 0.00 50.62 <of0maleunknown> amerikanIn en yakIn mUttefiklerinden biri olan suudi arabistanIn amerikayI eleStirmesi sUpriz yarattI Ote yandan telaferde sUnnilerin oturduGu bir mahalleye giriSilen saldIrIda yetmiS kiSi OldU saGlIk gOrevlileri OldUrUlen kiSilerin baSlarIndan vurulduklarInI bildiriyor IraklI yetkililer saldIrIya Sii polislerin de katIldIGInI aCIklarken Irak ordusu kentte sokaGa CIkma yasaGI ilan etti saldIrInIn birgUn Once Siileri hedef alan bombalI saldIrIlara misilleme olduGu sanIlIyor bombayUklU iki araCla dUzenlenen saldIrIlarda en azaltmIS kiSi hayatInI kaybetmiS yUzU aSkIn kiSi de yaralanmIStI Ote yandan fellucede araClarIna zehirli gaz yUkleyen intihar bombacIlarI hUkUmet merkezine saldIrdI amerikan ordusu saldIrInIn pUskUrtUldUGUnU ancak CatISmada on beS amerikalI ve IraklI askerin yaralandIGInI aCIkladI ayrIca gazdan zehirlenen Cok sayIda IraklI asker ve polis tedavi altIna alIndI .

AlparslanesmerA 1 excluded_region 50.62 50.62 <ounknown> .

STEP 2: Converting into MLF files by using a simple Perl script. The output is at below.

#!MLF!#

"*/data_excluded_region_0_0.lab"

.

"*/data_AlparslanesmerA_0_5062.lab"

amerikanIn

en

yakIn

mUttefiklerinden

biri

olan

suudi

arabistanIn

amerikayI

eleStirmesi

sUpriz

yarattI

Ote

yandan

telaferde

sUnnilerin

oturduGu

bir

mahalleye

giriSilen

saldIrIda

yetmiS

kiSi

OldU

saGlIk

gOrevlileri

OldUrUlen

kiSilerin

baSlarIndan

vurulduklarInI

bildiriyor

IraklI

yetkililer

saldIrIya

Sii

polislerin

de

katIldIGInI

aCIklarken

Irak

ordusu

kentte

sokaGa

CIkma

yasaGI

ilan

etti

saldIrInIn

bir

gUn

Once

Siileri

hedef

alan

bombalI

saldIrIlara

misilleme

olduGu

sanIlIyor

bomba

yUklU

iki

araCla

dUzenlenen

saldIrIlarda

en

az

altmIS

kiSi

hayatInI

kaybetmiS

yUzU

aSkIn

kiSi

de

yaralanmIStI

Ote

yandan

fellucede

araClarIna

zehirli

gaz

yUkleyen

intihar

bombacIlarI

hUkUmet

merkezine

saldIrdI

amerikan

ordusu

saldIrInIn

pUskUrtUldUGUnU

ancak

CatISmada

on

beS

amerikalI

ve

IraklI

askerin

yaralandIGInI

aCIkladI

ayrIca

gazdan

zehirlenen

Cok

sayIda

IraklI

asker

ve

polis

tedavi

altIna

alIndI

.

"*/data_excluded_region_5062_5062.lab"

.

STEP 3: Obtaining MFCC list files. Each ".lab" file which is shown above should also have an ".list" file which includes corresponding MFCC vectors.

data_AlparslanesmerA_0_5062.mfc=AlparslanesmerA.mfc[0,5062]
data_excluded_region_5062_5062.mfc=AlparslanesmerA.mfc[5062,5062]
data_AlparslanesmerA_5062_6815.mfc=AlparslanesmerA.mfc[5062,6815]

The meaning of above three lines is that, the first line addresses that the corresponding MFCC vectors form t=0 second to t=50.62 second is the parameters 0 to 5062. Where the TARGETRATE in HCopy configuration file was 100, so 100 parameters are constructed for each second.

Table 5.14 MLF and MFCC index files

bookbuksil
bookbuksp
cumhuriyetCicumhuriyetC1isil
cumhuriyetCicumhuriyetC1isp
ikiikisil
ikiikisp
saraysaraysil
saraysaraysp
<s>sil

Table 5.15 A part of dictionary is shown for a couple of words. Each word has corresponding pronunciations, i.e. phone utterances.

5.5.4 Testing Stage

The performance of the trained HMMs can be measured by using HResults tool. The recognized outputs are compared with reference inputs. The error is evaluated in terms of word error rates.

#!MLF!# "data_AlparslanesmerA_0_5062.rec" 0 700000 sil<s> 700000 1600000 a amerikanIn 1600000 1900000 m 1900000 2500000 e 2500000 2800000 r 2800000 3100000 i 3100000 3900000 k 3900000 4500000 a 4500000 5100000 n 5100000 5400000 I1 5400000 5700000 n 5700000 5700000 sp	5700000 6000000 e en 6000000 6600000 n 6600000 6600000 sp 6600000 7200000 y yakIn 7200000 7600000 a 7600000 8600000 k 8600000 9000000 I1 9000000 9400000 n 9400000 9400000 sp 9400000 9900000 m mUttefiklerinden 9900000 10300000 U1 10300000 10800000 t 10800000 11500000 t 11500000 11900000 e	11900000 12700000 f 12700000 13100000 i 13100000 14100000 k 14100000 14400000 l 14400000 15100000 e 15100000 15400000 r 15400000 15700000 i 15700000 16000000 n 16000000 16300000 d 16300000 17400000 e 17400000 17900000 n 17900000 17900000 sp 17900000 18300000 b biri	18300000 18800000 i 18800000 19100000 r 19100000 19400000 i 19400000 19400000 sp 19400000 20600000 o olan 20600000 21000000 l 21000000 21900000 a 21900000 22200000 n 22200000 22200000 sp 22200000 23700000 s suudi 23700000 24100000 u 24100000 24500000 u 24500000 25100000 d	25100000 25600000 i 25600000 25600000 sp 25600000 25900000 a arabistanIn 25900000 26200000 r 26200000 27000000 a 27000000 27300000 b 27300000 27700000 i 27700000 28600000 s 28600000 29200000 t 29200000 29500000 a 29500000 29800000 n 29800000 30100000 I1 30100000 30700000 n	30700000 30700000 sp 30700000 31700000 a amerikayI 31700000 32300000 m 32300000 32600000 e 32600000 32900000 r 32900000 33300000 i 33300000 34400000 k 34400000 34700000 a 34700000 35700000 y 35700000 36000000 I1 36000000 36000000 sp 36000000 36300000 e eleStirmesi 36300000 36700000 l	36700000 37300000 e 37300000 38800000 S1 38800000 39100000 t 39100000 39400000 i 39400000 39700000 r 39700000 40300000 m 40300000 40700000 e 40700000 41800000 s 41800000 42100000 i 42100000 42100000 sp 42100000 43200000 s sUpriz 43200000 43700000 U1 43700000 44600000 p	44600000 44900000 r 44900000 45300000 i 45300000 46400000 z 46400000 46400000 sp 46400000 46800000 y yarattI 46800000 47100000 a 47100000 47400000 r 47400000 48300000 a 48300000 49000000 t 49000000 49600000 t 49600000 50200000 I1 50200000 55100000 sil

Table 5.16 Word and Phoneme based timings in terms of 100 Nano seconds.

Chapter 6

Prosody and Prosodic Feature Extraction

6.1 The Definition of Prosody

The speech and the speech signal is more than utterance of words in a given transcript of a speech [32]. The speech signal contains more information such as structural, semantics, functional and emotional information. In addition, prosody includes duration information such as pause duration between words, even though between phonemes [32]. Furthermore prosody includes also information related changes in pitch range and amplitude, global pitch declination, melody and tone distribution and speaking rate variation [33]. By using prosodic features with ASR output, we are able to do sentence segmentation, topic segmentation and summarization, which will be discussed at Chapter 8. For instance longer pause duration between two words is one of the most important clues while detecting sentence boundary. The prosodic features and extraction of prosodic features will be discussed below.

6.2 The Prosodic Features

There are three main types of features which are basis features, statistical features and derived features. In the following sections, the features corresponding to their types are

given below with definitions [31][35]. The used feature sets for event detection is discussed at Chapter 8.

6.2.1 Basic Features

The basis features are divided into three sub classes such as duration, energy and F_0 features.

6.2.1.1 Base Features

Base features include the basic information about the waveform and phone/word time alignments of corresponding waveforms. Before running the program to extract prosodic feature, we fill information about the session, speaker, gender and the location of waveform which has shown at Table 6.n. Basic features include that information which are shown at Table 6.1. The following tables are prepared by considering the definitions of the features of the Prosodic Toolkit [31, 35].

WAV:	The location i.e. path of the current audio waveform.
SPK_ID:	Identification label, for instance name of the speaker.
SPK_GEN:	The gender of the speaker. Male/Female.

Table 6.1 The base features of Praat outputs.

6.2.1.2 Duration Features

The duration features are based on either word and phone alignments of human transcripts or ASR output [34] and related with pauses and duration of phone and rhymes. There are two types of duration features. The first one is pause features and second is phone and rhyme duration features. Firstly, the pause features are related with the elapsed time between two word boundaries, which is a very important cue to detect semantic units such as sentences or topics [33] which will be discussed more detailed at Chapter 8. Secondly the phone and rhyme duration features are related with the phone durations in the last

rhyme of a word. Similarly the phone and rhyme features are also other well-known cue in sentence segmentation [33], which will be also discussed at Chapter 8. Examples to duration features are; word duration, following word duration, last rhyme duration, the pause between the current word and the following word and last phoneme duration. The following tables are prepared by considering the definitions of the features of the Prosodic Toolkit [31, 35].

6.2.1.3 F_0 Features

The F_0 feature are related with the pitch, where pitch is defined as the relative highness or lowness of a tone as perceived by the ear and depends on the vibrations per second produced by the vocal cords [35]. The following tables are prepared by considering the definitions of the features of the Prosodic Toolkit [31, 35].

MIN_F0:	The minimum raw F_0 value of the current word.
MAX_F0:	The maximum raw F_0 value of the current word.
MEAN_F0:	The mean raw F_0 value of the current word.
- **MIN_F0_NEXT:** - **MAX_F0_NEXT:** - **MEAN_F0_NEXT:**	Those features are similar with the features which are without "_NEXT". The difference is that, those are computed for the following word, i.e. word after a boundary.
- **MIN_F0_WIN:** - **MAX_F0_WIN:** - **MEAN_F0_WIN:** - **MIN_F0_NEXT_WIN:** - **MAX_F0_NEXT_WIN:** - **MEAN_F0_NEXT_WIN:**	Those features are similar with the features are without "_WIN". The difference is that, those features are computed over N frames before a boundary. However if there are not enough data, than maximum number of frames are considered.

Table 6.2 Features computed using the raw F_0 extracted by Praat.

WORD:	The word preceding a boundary.
WORD_START:	The start time of the preceding word, i.e. current word.
WORD_END:	The end time of the current word.
FWORD:	The word following a boundary, i.e. next word.
FWORD_START:	The start time of the next word.
FWORD_END:	The end time of the next word.
PAUSE_START:	The start time of the pause between sequences of word. If there are a word exists at following boundary, then the value is equal to FWORD_END.
PAUSE_END:	The end time of the pause. If there are a following word i.e. FWORD then it is set as FWORD_START. If there is no following word then this is set to end of the waveform.
PAUSE_DUR:	The duration of the pause between WORD and FWORD.
WORD_PHONES:	The phones in the word with their duration.(phone1:duration1_phone2:duration2_...)
FLAG:	It compares the word duration and sum of the phone durations. If the durations are not reliable, then the value is set to "SUSP" i.e. suspicious word. Furthermore if there are missing phones, then the value is set to "?", else the value is set to zero.
LAST_VOWEL:	The last vowel of the word preceding a boundary, i.e. last vowel of the current word. If it doesn't exist then all of the LAST_VOWEL related features will be assigned to "?".
LAST_VOWEL_START:	The start time of the current words last vowel.
LAST_VOWEL_END:	The end time of the current words last vowel.
LAST_VOWEL_DUR:	The duration of the current words last vowel.
LAST_RHYME_START:	The start time of the current words last rhyme. Where last rhyme is defined as the sequence of phones starting with the last vowel to the end of the word.
LAST_RHYME_END:	Same as WORD_END.
NORM_LAST_RHYME_DUR:	$\sum_{every\ phone\ in\ word} \frac{dur(phone) - mean(phone)}{std\_dev(phone)}$ Where dur(phone) is the duration of the phone in current time, mean(phone) and std_dev(phone) are average and standard deviation of the duration of that phone in the training data.
PHONES_IN_LAST_RHYME:	The total number of phones in the last rhyme.

Table 6.3 The duration features.

Stylizing F_0:

The stylizing F_0 algorithm has described below [35].

1. The raw values of F_0 for each frame are obtained by using get_f0 [36], where the fundamental frequency (F0) estimated by using the normalized cross correlation function and dynamic programming.
2. Extracting and labeling a sequence of voiced frames which are longer than a certain number of frames.
3. LTM Modeling. (Lognormal tied mixture model of pitch.) Where pitch is modeled as combination of three lognormal distributions with tied means and variances [37]. The output of Step 1 has two main noise sources which will be minimized in the next step [33]. F0 halving and doubling is being estimated by using a Lognormal tied mixture model.
4. By using the LTM parameters, the probability of halving, doubling and integral pitch for each frame are computed.
5. Median filter F_0in voiced frames.
6. Stylize F_0 with a piecewise linear model in voiced frames. Use a greedy algorithm which detects discontinuities where the mean square error exceeds a normalized threshold.
7. Compute slopes such as rises and falls.

Table 6.4 A brief algorithm to obtain stylized F_0 [35]

MIN_STYLFIT_F0:	The minimum stylized F_0 value for the current word.
MAX_STYLFIT_F0:	The maximum stylized F_0 value for the current word.
MEAN_ STYLFIT_F0:	The mean stylized F_0 value for the current word.
FIRST_ STYLFIT_F0:	The first stylized F_0 value for the current word.
LAST_ STYLFIT_F0:	The last stylized F_0 value for the current word.
- **MIN_STYLFIT_F0_NEXT:** - **MAX_STYLFIT_F0_NEXT:** - **MEAN_ STYLFIT_F0_NEXT:** - **FIRST_ STYLFIT_F0_NEXT:** - **LAST_ STYLFIT_F0_NEXT:**	Those features are similar as the features without "_NEXT"; the difference is that, they are computed for the following word, i.e. word after a boundary.
- **MIN_STYLFIT_F0_WIN:** - **MAX_STYLFIT_F0_WIN:** - **MEAN_ STYLFIT_F0_WIN:** - **FIRST_ STYLFIT_F0_WIN:** - **LAST_ STYLFIT_F0_WIN:** - **MIN_STYLFIT_F0_NEXT_WIN:** - **MAX_STYLFIT_F0_NEXT_NEXT_WIN:** - **MEAN_ STYLFIT_F0_NEXT_WIN:** - **FIRST_ STYLFIT_F0_NEXT_WIN:** - **LAST_ STYLFIT_F0_NEXT_WIN:**	Those features are similar as the features without "_WIN"; the difference is that, they are computed over N frames before a boundary. However if there are not enough data, then maximum number of frames will be considered.

Table 6.5 Features computed using stylized F_0

PATTERN_WORD:

This feature detects *falling slope*, *unvoiced section* and *rising slope* in the word preceding a boundary. The feature is combinations of *"f", "u"* and *"r"* which represent *falling slope*, *unvoiced section* and *rising slope* respectively.

PATTERN_WORD_COLLAPSED:	Similar with PATTERN_WORD, where a sequence of f's or r's are represented as one f or r.
PATTERN_SLOPE:	Similar to PATTERN_WORD, except instead of printing f or r's the slope values are printed. (value1:value2:…)
- **PATTERN_WORD_NEXT:** - **PATTERN_WORD_CALLAPSED_NEXT:** - **PATTERN_SLOPE_NEXT:**	Those features are similar with the features which are without "_NEXT". The difference is that, those are computed for the following word, i.e. word after a boundary.
- **PATTERN_WORD_WIN:** - **PATTERN_WORD_CALLAPSED_WIN:** - **PATTERN_SLOPE_WIN:** - **PATTERN_WORD_NEXT_WIN:** - **PATTERN_WORD_CALLAPSED_NEXT_WIN:** - **PATTERN_SLOPE_NEXT_WIN:**	Those features are similar as the features without "_WIN"; the difference is that, they are computed over N frames before a boundary. However if there are not enough data, then maximum number of frames will be considered.

Table 6.6 Stylized F_0 contour slope features.

PATTERN_BOUNDARY:

The value of PATTERN_WORD is combined with the value of PATTERN_NEXT_WORD.

SLOPE_DIFF:

The difference between the last non-zero slope of the current word and first non-zero slope of the next word. The length of slopes which exceed minimum frame length. If that is not found then a "?" is assigned by default.

Table 6.7 Features extracted considering word boundaries.

NO_PREVIOUS_SSF:

Counts number of previous cascade frames which are bigger than minimum frame length in the current word which have the same slope as last voiced frame of previous word.

NO_PREVIOUS_VF:

Counts the number of cascade "voiced" frames inside the word. Where, the counting starts at the last voiced frame and goes through backwards.

NO_FRAMES_LS_WE:

Counts the number of cascade frames between last voiced frame and end of the word. The last voiced frame is selected by considering the sequence of voiced frames is bigger than the minimum frame length.

NO_SUCCESSOR_SSF:

Counts a group of frames inside the current word, where each group of frames includes cascade frames has same slope with the first voiced frame of the current word and the length of each sequence of frame is bigger than minimum frame length.

NO_SUCCESSOR_VF:

Number of cascade "voiced" frames inside the current word.

NO_FRAMES_WS_FS:

Number of cascade frames between the first frame of the current word and the first voiced frame, where sequence of voiced frames which are larger than minimum frame length are considered.

The following features are similar with the features which are without "_NEXT". The difference is that, those are computed for the following word, i.e. word after a boundary.

- **NO_PREVIOUS_SSF_NEXT**
- **NO_PREVIOUS_VF_NEXT**
- **NO_FRAMES_LS_WE_NEXT**
- **NO_SUCCESSOR_SSF_NEXT**
- **NO_SUCCESSOR_VF_NEXT**
- **NO_FRAMES_WS_FS_NEXT**

The followingfeatures are similar as the features without "_WIN"; the difference is that, they are computed over N frames before a boundary. However if there are not enough data, then maximum number of frames will be considered.

- **NO_PREVIOUS_SSF_WIN**
- **NO_PREVIOUS_VF_WIN**
- **NO_PREVIOUS_LS_WE_WIN**
- **NO_SUCCESSOR_SSF_WIN**
- **NO_SUCCESSOR_VF_WIN**
- **NO_FRAMES_WS_FS_WIN**
- **NO_PREVIOUS_SSF_NEXT_WIN**
- **NO_PREVIOUS_VF_NEXT_WIN**
- **NO_FRAMES_WE_NEXT_WIN**
- **NO_SUCCESSOR_SSF_NEXT_WIN**
- **NO_SUCCESSOR_VF_NEXT_WIN**
- **NO_FRAMES_WS_FS_NEXT_WIN**

Table 6.8 Features involve counting.

6.2.1.4 Energy Features

Short-time energy calculations for each frame is performed in order to extract the Energy Features which are shown at Table 6.12

MIN_ENERGY	**MIN_ENERGY_WIN**
MAX_ENERGY	**MAX_ENERGY_WIN**
MEAN_ENERGY	**MEAN_ENERGY_WIN**
MIN_ENERGY_NEXT	**MIN_ENERGY_NEXT_WIN**
MAX_ENERGY_NEXT	**MAX_ENERGY_NEXT_WIN**
MEAN_ENERGY_NEXT	**MEAN_ENERGY_NEXT_WIN**

Table 6.9 Features computed using the raw Energy extracted by Praat.

MIN_STYLFIT_ENERGY	**MIN_STYLFIT_ENERGY_WIN**
MAX_STYLFIT_ENERGY	**MAX_STYLFIT_ENERGY_WIN**
MEAN_STYLFIT_ENERGY	**MEAN_STYLFIT_ENERGY_WIN**
FIRST_STYLFIT_ENERGY	**FIRST_STYLFIT_ENERGY_WIN**
LAST_STYLFIT_ENERGY	**LAST_STYLFIT_ENERGY_WIN**
MIN_STYLFIT_ENERGY_NEXT	**MIN_STYLFIT_ENERGY_NEXT_WIN**
MAX_STYLFIT_ENERGY_NEXT	**MAX_STYLFIT_ENERGY_NEXT_WIN**
MEAN_STYLFIT_ENERGY_NEXT	**MEAN_STYLFIT_ENERGY_NEXT_WIN**
FIRST_STYLFIT_ENERGY_NEXT	**FIRST_STYLFIT_ENERGY_NEXT_WIN**
LAST_STYLFIT_ENERGY_NEXT	**LAST_STYLFIT_ENERGY_NEXT_WIN**

Table 6.10 Features computed using stylized Energy

ENERGY_PATTERN_WORD	**ENERGY_PATTERN_WORD_WIN**
ENERGY_PATTERN_WORD_CALLAPSED	**ENERGY_PATTERN_WORD_CALLAPSED_WIN**
ENERGY_PATTERN_SLOPE	**ENERGY_PATTERN_SLOPE_WIN**
ENERGY_PATTERN_WORD_NEXT	**ENERGY_PATTERN_WORD_NEXT_WIN**
ENERGY_PATTERN_WORD_	**ENERGY_PATTERN_WORD_**
CALLAPSED_NEXT	**CALLAPSED_NEXT_WIN**
ENERGY_PATTERN_SLOPE_NEXT	**ENERGY_PATTERN_SLOPE_NEXT_WIN**

Table 6.11 Features involve counting.

ENERGY_PATTERN_BOUNDARY	**ENERGY_SLOPE_DIFF**

Table 6.12 Features extracted considering word boundaries.

6.2.2 Statistical Tables

The data inside the tables are not output features, but they are used to compute the derived features

a. **Phone Duration Statistics (phone_dur.stats):** Mean phone duration, the standard deviation phone duration, the number of occurrences of that phone in training database and phone duration threshold (shown at Equation 6.1) is computed for each phone.

$$Phone_{treshold} = \mu_{phone} + 10\ \sigma_{phone} \tag{6.1}$$

b. **Pause Duration Statistics (pause_dur.stats):** Mean, standard deviation of pauses is listed for each audio.

c. **Speaker Feature Statistics (spkr_feat.stats):** In this table, statistics related with cascade voiced and unvoiced frames F_0, F_0 slope, Energy and Energy slope. One row is dedicated to one speaker. The values are calculated in logarithm (base e). The detailed list has shown at Table 6.13.

d. **Speaker Phone Duration Statistics (spkr_phone_dur.stats):** Similar with "phone_dur_stats" the difference is considers all of the audio files corresponding to the selected speaker.

e. **Last Rhyme Duration Statistics (last_rhyme_dur.stats)**

f. **Last Rhyme Phone Duration Statistics (last_rhyme_phone_dur.stats)**

g. **Pause Duration Statistics (pause_dur.stats) :**Mean and standard deviation of duration of the pauses in the audio, logarithm of (base e) mean and standard deviation of the durations and number of pauses in the audio is counted.

Voiced Frames	**MEAN_VOICED**
	STDEV_VOICED
	COUNT_VOICED
Unvoiced Frames	**MEAN_UNVOICED**
	STDEV_UNVOICED
	COUNT_UNVOICED
F_0Value Related	**MEAN_PITCH**
	STDEV_PITCH
	COUNT_PITCH
F_0Slope Related	**MEAN_SLOPE**
	STDEV_SLOPE
	COUNT_SLOPE
Energy Related	**MEAN_ENERGY**
	STDEV_ENERGY
	COUNT_ENERGY
Energy Slope Related	**MEAN_ENERGY_SLOPE**
	STDEV_ENERGY_SLOPE
	COUNT_ENERGY_SLOPE

TABLE 6.13 Speaker Feature Statistics.

6.2.3 Derived Features

By using the basic features which were described in 6.2.1 and statistics which are shown at 6.2.2 the derived features are computed. Generally derived features are either computed by using two basis features or using the computed statistics. There are several derived features described below such as normalized durations, F_0 and Energy derived features, average phone durations and speaker specific normalization features.

6.2.3.1 Normalized Word Duration

- **WORD_DUR:** The duration of the current word.

WORD_DUR = WORD_END – WORD_START (6.2)

Where WORD_END and WORD_START are basic features.

- **WORD_AV_DUR:** Average word duration.

WORD_AV_DUR= $\sum_{evey\ phone\ in\ word} mean(phone)$ (6.3)

Where *mean(phone)* is a statistics which obtained from pause duration statistics and phones are obtained from the basic feature WORD_PHONES

- **NORM_WORD_DUR:** Normalized word duration.

NORM_WORD_DUR = $\frac{WORD\_DUR}{WORD\_AV\_DUR}$ (6.4)

6.2.3.2 Normalized Pause

- **PAU_DUR_N:** Normalized pause duration.

PAU_DUR_N = $\frac{PAU\_DUR}{PAUSE\_MEAN}$ (6.5)

Where PAUSE_MEAN comes from pause duration statistics.

6.2.3.3 Normalized Vowel Duration

- **LAST_VOWEL_DUR_Z:**

$$\text{LAST\_VOWEL\_DUR\_Z} = \frac{LAST\_VOWEL\_DUR - ALL\_PHONE\_DUR\_MEAN}{ALL\_PHONE\_DUR\_STDEV} \quad (6.6)$$

Where LAST_VOWEL_DUR is a basic duration feature, ALL_PHONE_DUR_MEAN and ALL_PHONE_DUR_STDEV are statistics taken from phone duration statistics and they are related to another basic feature LAST_VOWEL.

- **LAST_VOWEL_DUR_N:**

$$\text{LAST\_VOWEL\_DUR\_N} = \frac{LAST\_VOWEL\_DUR}{ALL\_PHONE\_DUR\_MEAN} \quad (6.7)$$

- **LAST_VOWEL_DUR_ZSP:**

$$\text{LAST\_VOWEL\_DUR\_ZSP} = \frac{LAST\_VOWEL\_DUR - SPKR\_PHONE\_DUR\_MEAN}{SPKR\_PHONE\_DUR\_STDEV} \quad (6.8)$$

Where SPKR_PHONE_DUR_STDEV and SPKR_PHONE_DUR_MEAN are speaker related statistics. The phone duration statistics and the basic features SPKR_ID and LAST_VOWEL are considered.

- **LAST_VOWEL_DUR_NSP:**

$$\text{LAST\_VOWEL\_DUR\_NSP} = \frac{LAST\_VOW\_DUR}{SPKR\_PHONE\_DUR\_MEAN} \quad (6.9)$$

6.2.3.4 Normalized Rhyme Duration

The normalized rhyme features are shown at Table 6.14 with equations.

LAST_RHYME_DUR_PH= $\frac{LAST\_RHYME\_DUR}{PHONES\_IN\_LAST\_RHYME}$	(6.10)
LAST_RHYME_DUR_PH_ND= $\frac{LAST\_RHYME\_DUR - PHONES\_IN\_LAST\_RHYME}{LAST\_RHYME\_PHONE\_DUR\_MEAN}$	(6.11)
LAST_RHYME_DUR_PH_NR = $\frac{LAST\_RHYME\_DUR \,/\, PHONES\_IN\_LAST\_RHYME}{LAST\_RHYME\_PHONE\_DUR\_MEAN}$	(6.12)
LAST_RHYME_NORM_DUR_PH = $\frac{NORM\_LAST\_RHYME\_DUR}{PHONES\_IN\_LAST\_RHYME}$	(6.13)
LAST_RHYME_NORM_DUR_PH_ND=$\frac{NORM\_LAST\_RHYME\_DUR \,/\, PHONES\_IN\_LAST\_RHYME}{NORM\_LAST\_RHYME\_PHONE\_DUR\_MEAN}$	(6.14)
LAST_RHYME_NORM_DUR_PH_NR = $\frac{NORM\_LAST\_RHYME\_DUR \,/\, PHONES\_IN\_LAST\_RHYME}{NORM\_LAST\_RHYME\_PHONE\_DUR\_MEAN}$	(6.15)
LAST_RHYME_DUR_WHOLE_ND = $LAST\_RHYME\_DUR - LAST\_RHYME\_WHOLE\_DUR\_MEAN$	(6.16)
LAST_RHYME_WHOLE_DUR_ND = $\frac{LAST\_RHYME\_DUR}{LAST\_RHYME\_WHOLE\_DUR\_MEAN}$	(6.17)
LAST_RHYME_WHOLE_DUR_Z = $\frac{LAST\_RHYME\_DUR - LAST\_RHYME\_WHOLE\_DUR\_MEAN}{LAST\_RHYME\_WHOLE\_DUR\_STDEV}$	(6.18)

Table 6.14 Normalized Rhyme Duration features.

In Table 6.14; LAST_RHYME_DUR, PHONES_IN_LAST_RHYME, and NORM_LAST_RHYME_DUR are duration features, LAST_RHYME_PHONE_DUR_MEAN is a statistics which taken from mean column of last rhyme phone duration statistics table, NORM_LAST_RHYME_PHONE_DUR_MEAN is a statistics and taken from mean column of normalized last rhyme phone duration statistics table, and finally LAST_RHYME_WHOLE_DUR_MEAN and LAST_RHYME_WHOLE_DUR_STDEV are statistics taken from mean and standard deviation columns and audio rows of last rhyme duration statistics table.

6.2.3.5 F_0 Derived Features

The F_0 characteristics of the speaker, where the related features are shown at Table 6.15, are computed by using *Praat's* built-in function for stylization and pitch statistics.

SPKR_FEAT_F0_MODE = $\exp(SPKR\_F0\_MEAN)$	(6.19)
SPKR_FEAT_F0_TOPLN =$0.75\ x\ \exp(SPKR\_F0\_MEAN)$	(6.20)
SPKR_FEAT_F0_BASELN = $1.50\ x\ \exp(SPKR\_F0\_MEAN)$	(6.21)
SPKR_FEAT_F0_STDLIN = $\exp(SPKR\_F0\_STDEV)$	(6.22)
SPKR_FEAT_F0_RANGE = $SPKR\_FEAT\_F0\_TOPLN\ -\ SPKR\_FEAT\_F0\_BASELN$	(6.23)

Table 6.15 F_0 characteristics of the speaker.

The following features shown at Table 6.16 are computed between previous and next word. The log differences of minimum, maximum and mean values of stylized F_0 values of the words are used.

F0K_WORD_DIFF_HIHI_N = $\log\ \left(\frac{MAX\_STYLFIT\_FO}{MAX\_STYLFIT\_FO\_NEXT}\right)$	(6.24)
F0K_WORD_DIFF_HILO_N = $\log\ \left(\frac{MAX\_STYLFIT\_FO}{MIN\_STYLFIT\_FO\_NEXT}\right)$	(6.25)
F0K_WORD_DIFF_LOLO_N = $\log\ \left(\frac{MIN\_STYLFIT\_FO}{MIN\_STYLFIT\_FO\_NEXT}\right)$	(6.26)
F0K_WORD_DIFF_LOHI_N = $\log\ \left(\frac{MIN\_STYLFIT\_FO}{MAX\_STYLFIT\_FO\_NEXT}\right)$	(6.27)
F0K_WORD_DIFF_MNMN_N = $\log\ \left(\frac{MEAN\_STYLFIT\_FO}{MEAN\_STYLFIT\_FO\_NEXT}\right)$	(6.28)

Table 6.16 All of the features used in computation are basic F_0 features.

The following features shown at Table 6.17 are computed between previous and next word. The log differences of minimum, maximum and mean values of normalized pitch range of the words are used.

F0K_WORD_DIFF_HIHI_NG = $\frac{\log\left(\frac{MAX\_STYLFIT\_F0}{MAX\_STYLFIT\_F0\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE}$	(6.29)
F0K_WORD_DIFF_HILO_NG = $\frac{\log\left(\frac{MAX\_STYLFIT\_F0}{MIN\_STYLFIT\_F0\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE}$	(6.30)
F0K_WORD_DIFF_LOLO_NG = $\frac{\log\left(\frac{MIN\_STYLFIT\_F0}{MIN\_STYLFIT\_F0\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE}$	(6.31)
F0K_WORD_DIFF_LOHI_NG = $\frac{\log\left(\frac{MIN\_STYLFIT\_F0}{MAX\_STYLFIT\_F0\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE}$	(6.32)
F0K_WORD_DIFF_MNMN_NG = $\frac{\log\left(\frac{MEAN\_STYLFIT\_F0}{MEAN\_STYLFIT\_F0\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE}$	(6.33)

Table 6.17 Formant frequency differences between adjacent words.

- The features shown at Table 6.18 are computed between previous and next window. The log differences of minimum, maximum and mean values of stylized F_0 values of the windows are used.
- The features shown at Table 6.19 are computed between previous and next window. The log differences of minimum, maximum and mean values of normalized pitch range of the windows are used.
- The features shown at Table 6.20 are the difference, log difference between last, mean and minimum of the stylizedF_0values in a window and the baseline of F_0 values.

F0K_WIN_DIFF_HIHI_N =

$$\log\left(\frac{MAX\_STYLFIT\_FO\_WIN}{MAX\_STYLFIT\_FO\_WIN\_NEXT}\right) \tag{6.34}$$

F0K_WIN_DIFF_HILO_N =

$$\log\left(\frac{MAX\_STYLFIT\_FO\_WIN}{MIN\_STYLFIT\_FO\_WIN\_NEXT}\right) \tag{6.35}$$

F0K_WIN_DIFF_LOLO_N =

$$\log\left(\frac{MIN\_STYLFIT\_FO\_WIN}{MIN\_STYLFIT\_FO\_WIN\_NEXT}\right) \tag{6.36}$$

F0K_WIN_DIFF_LOHI_N =

$$\log\left(\frac{MIN\_STYLFIT\_FO\_WIN}{MAX\_STYLFIT\_FO\_WIN\_NEXT}\right) \tag{6.37}$$

F0K_WIN_DIFF_MNMN_N =

$$\log\left(\frac{MEAN\_STYLFIT\_FO\_WIN}{MEAN\_STYLFIT\_FO\_WIN\_NEXT}\right) \tag{6.38}$$

Table 6.18 Formant frequency differences between two adjacent windows.

F0K_WIN_DIFF_HIHI_NG =

$$\frac{\log\left(\frac{MAX\_STYLFIT\_FO\_WIN}{MAX\_STYLFIT\_FO\_WIN\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE} \tag{6.39}$$

F0K_WIN_DIFF_HILO_NG =

$$\frac{\log\left(\frac{MAX\_STYLFIT\_FO\_WIN}{MIN\_STYLFIT\_FO\_WIN\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE} \tag{6.40}$$

F0K_WIN_DIFF_LOLO_NG =

$$\frac{\log\left(\frac{MIN\_STYLFIT\_FO\_WIN}{MIN\_STYLFIT\_FO\_WIN\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE} \tag{6.41}$$

F0K_WIN_DIFF_LOHI_NG =

$$\frac{\log\left(\frac{MIN\_STYLFIT\_FO\_WIN}{MAX\_STYLFIT\_FO\_WIN\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE} \tag{6.42}$$

F0K_WIN_DIFF_MNMN_NG =

$$\frac{\log\left(\frac{MEAN\_STYLFIT\_FO\_WIN}{MEAN\_STYLFIT\_FO\_WIN\_NEXT}\right)}{SPKR\_FEAT\_F0\_RANGE} \tag{6.43}$$

Table 6.19 Normalized formant frequency differences between two adjacent windows.

F0K_DIFF_LAST_KBASELN =

$$LAST\_STYLFIT\_FO\ -SPKR\_FEAT\_F0\_BASELN \tag{6.43}$$

F0K_DIFF_MEAN_KBASELN =

$$MEAN\_STYLFIT\_F0\ -SPKR\_FEAT\_FO\_BASELN \tag{6.44}$$

F0K_DIFF_WINMIN_KBASELN =

$$MIN\_STYLFIT\_F0\_WIN\ -SPKR\_FEAT\_F0\_BASELN \tag{6.45}$$

F0K_LR_LAST_KBASELN =

$$\log\left(\frac{LAST\_STYLFIT\_FO}{SPKR\_FEAT\_F0\_BASELN}\right) \tag{6.46}$$

F0K_LR_MEAN_KBASELN =

$$\log\left(\frac{MEAN\_STYLFIT\_FO}{SPKR\_FEAT\_F0\_BASELN}\right) \tag{6.47}$$

F0K_LR_WINMIN_KBASELN =

$$\log\left(\frac{MIN\_STYLFIT\_FO\_WIN}{SPKR\_FEAT\_F0\_BASELN}\right)$$

Table 6.20 Differences of stylized f0 frequencies.

- The features shown at Table 6.21 are the normalization of the stylized F_0 values in the current word and the following word. The baseline, topline and range of F_0 are used.
- The features shown at Table 6.22 are the difference and the log difference between mean and maximum stylized F_0 values of next word and the topline of F_0values.
- The features shown at Table 6.23 are normalization of the maximum stylized F_0 values in the current word and next word using the pitch of F_0values and pitch mode.
- The features shown at Table 6.24 are log differences between the stylized F_0values in the word extremes.

F0K_ZRANGE_MEAN_KBASELN =

$$\frac{MEAN\_STYLFIT\_F0 \; - SPKR\_FEAT\_F0\_BASELN}{SPKR\_FEAT\_F0\_RANGE} \tag{6.48}$$

F0K_ZRANGE_MEAN_KTOPLN =

$$\frac{SPKR\_FEAT\_F0\_TOPLN \; - \; MEAN\_STYLFIT\_F0}{SPKR\_FEAT\_F0\_RANGE} \tag{6.49}$$

F0K_ZRANGE_MEANNEXT_KBASELN =

$$\frac{MEAN\_STYLFIT\_F0\_NEXT \; - SPKR\_FEAT\_F0\_BASELN}{SPKR\_FEAT\_F0\_RANGE} \tag{6.50}$$

F0K_ZRANGE_MEANNEXT_ KTOPLN =

$$\frac{SPKR\_FEAT\_F0\_TOPLN \; - \; MEAN\_STYLFIT\_F0\_NEXT}{SPKR\_FEAT\_F0\_RANGE} \tag{6.51}$$

Table 6.21 Normalization difference of the stylized f0 frequencies.

F0K_DIFF_MEANNEXT_KTOPLN =

$$MEAN\_STYLFIT\_F0\_NEXT \; - SPKR\_FEAT\_FO\_TOPLN \tag{6.52}$$

F0K_DIFF_MAXNEXT_KTOPLN =

$$MAX\_STYLFIT\_F0\_NEXT \; - SPKR\_FEAT\_F0\_TOPLN \tag{6.53}$$

F0K_DIFF_WINMAXNEXT_KTOPLN =

$$MAX\_STYLFIT\_F0\_NEXT\_WIN \; - SPKR\_FEAT\_F0\_TOPLN \tag{6.54}$$

F0K_LR_MEANNEXT_KTOPLN =

$$\log\left(\frac{MEAN\_STYLFIT\_F0\_NEXT}{SPKR\_FEAT\_F0\_TOPLN}\right) \tag{6.55}$$

F0K_LR_MAXNEXT_KTOPLN =

$$\log\left(\frac{MAX\_STYLFIT\_F0\_NEXT}{SPKR\_FEAT\_F0\_TOPLN}\right) \tag{6.56}$$

F0K_LR_WINMAXNEXT_KTOPLN =

$$\log\left(\frac{MAX\_STYLFIT\_F0\_NEXT\_WIN}{SPKR\_FEAT\_F0\_TOPLN}\right) \tag{6.57}$$

Table 6.22 Differences of f0 frequencies between current word and next word.

F0K_MAXK_MODE_N	
$$\log\left(\frac{MAX\_STYLFIT\_F0}{SPKR\_FEAT\_F0\_MODE}\right)$$	(6.58)
F0K_MAXK_NEXT_MODE_N	
$$\log\left(\frac{MAX\_STYLFIT\_F0\_NEXT}{SPKR\_FEAT\_F0\_MODE}\right)$$	(6.59)
F0K_MAXK_MODE_Z	
$$\frac{MAX\_STYLFIT\_FO\ -SPKR\_FEAT\_F0\_MODE}{SPKR\_FEAT\_F0\_RANGE}$$	(6.60)
F0K_MAXK_NEXT_MODE_Z	
$$\frac{MAX\_STYLFIT\_FO\_NEXT\ -SPKR\_FEAT\_F0\_MODE}{SPKR\_FEAT\_F0\_RANGE}$$	(6.61)

Table 6.23 Normalized differences of f0 frequencies between current word and next word.

F0K_WORD_DIFF_BEGBEG =	
$$\log\left(\frac{FIRST\_STYLFIT\_FO}{FIRST\_STYLFIT\_FO\_NEXT}\right)$$	(6.62)
F0K_WORD_DIFF_ENDBEG =	
$$\log\left(\frac{LAST\_STYLFIT\_FO}{FIRST\_STYLFIT\_FO\_NEXT}\right)$$	(6.63)
F0K_INWRD_DIFF =	
$$\log\left(\frac{FIRST\_STYLFIT\_FO}{LAST\_STYLFIT\_FO}\right)$$	(6.64)

Table 6.24 Differences of stylized f0 frequencies in word extremes.

- The following features shown at Table 6.25 are slope patterns and the normalizations.

LAST_SLOPE:	The last slope "f" or "r" in the PATTERN_SLOPE.
FIRST_SLOPE_NEXT:	The first slope "f" or "r" in the PATTERN_SLOPE_NEXT
SLOPE_DIFF_N =	
$$\frac{SLOPE\_DIFF}{SPKR\_FEAT\_F0\_SD\_SLOPE}$$	(6.65)
LAST_SLOPE_DIFF_N =	
$$\frac{LAST\_SLOPE}{LAST\_STYLFIT\_F0}$$	(6.66)

Table 6.25 Slope patterns and normalizations.

6.2.3.6 Energy Derived Features

Energy derived features are derived similarly with F_0 derived features. Table 6.26 shows list of the derived energy features.

ENERGY_WORD_DIFF_HIHI_N	**ENERGY_WORD_DIFF_HILO_N**
ENERGY_WORD_DIFF_LOLO_N	**ENERGY_WORD_DIFF_LOHI_N**
ENERGY_WORD_DIFF_MNMN_N	**ENERGY_WORD_DIFF_HIHI_NG**
ENERGY_WORD_DIFF_HILO_NG	**ENERGY_WORD_DIFF_LOLO_NG**
ENERGY_WORD_DIFF_LOHI_NG	**ENERGY_WORD_DIFF_MNMN_NG**
ENERGY_WIN_DIFF_HIHI_N	**ENERGY_WIN_DIFF_HILO_N**
ENERGY_WIN_DIFF_LOLO_N	**ENERGY_WIN_DIFF_LOHI_N**
ENERGY_WIN_DIFF_MNMN_N	**ENERGY_WIN_DIFF_HIHI_NG**
ENERGY_WIN_DIFF_HILO_NG	**ENERGY_WIN_DIFF_LOLO_NG**
ENERGY_WIN_DIFF_LOHI_NG	**ENERGY_WIN_DIFF_MNMN_NG**
ENERGY_DIFF_LAST_KBASELN	**ENERGY_DIFF_MEAN_KBASELN**
ENERGY_DIFF_WINMIN_KBASELN	**ENERGY_LR_LAST_KBASELN**
ENERGY_LR_MEAN_KBASELN	**ENERGY_LR_WINMIN_KBASELN**
ENERGY_ZRANGE_MEAN_KBASELN	**ENERGY_ZRANGE_MEAN_KTOPLN**
ENERGY_ZRANGE_MEANNEXT_KBASELN	**ENERGY_ZRANGE_MEANNEXT_KTOPLN**
ENERGY_DIFF_MEANNEXT_KTOPLN	**ENERGY_DIFF_MAXNEXT_KTOPLN**
ENERGY_DIFF_WINMAXNEXT_KTOPLN	**ENERGY_LR_MEANNEXT_KTOPLN**
ENERGY_LR_MAXNEXT_KTOPLN	**ENERGY_LR_WINMAXNEXT_KTOPLN**
ENERGY_MAXK_MODE_N	**ENERGY_MAXK_NEXT_MODE_N**
ENERGY_MAXK_MODE_Z	**ENERGY_MAXK_NEXT_MODE_Z**
ENERGY_WORD_DIFF_BEGBEG	**ENERGY_WORD_DIFF_ENDBEG**
ENERGY_INWRD_DIFF	**ENERGY_LAST_SLOPE**
ENERGY_SLOPE_DIFF_N	**ENERGY_LAST_SLOPE_N**

Table 6.26 Energy derived features.

6.2.3.7 Average Phone Duration

Average phone duration features are shown at the Table 6.27.

AVG_PHONE_DUR_Z (6.67)

$$\frac{\sum_{every\ phone\ in\ word} phone_z\ [phone]}{number\ of\ phones}$$

MAX_PHONE_DUR_Z (6.68)

$$\max_{every\ phone\ in\ word} phone_z\ [phone]$$

AVG_PHONE_DUR_N (6.69)

$$\frac{\sum_{every\ phone\ in\ word} phone_n[phone]}{number\ of\ phones}$$

MAX_PHONE_DUR_N (6.70)

$$\max_{every\ phone\ in\ word} phone_n[phone]$$

Where,

$$phone_z[phone] = \frac{phone\_dur[phone]\ - phone\_dur\_mean[phone]}{phone\_dur\_stdev[phone]} \quad (6.71)$$

$$phone_n[phone] = \frac{phone\_dur[phone]}{phone\_dur\_mean[phone]} \quad (6.72)$$

phone_dur[phone] is phone duration for phone which has obtained from the feature WORD_PHONES and phone_dur_mean[phone] and phone_dur_stdev[phone] are computed at phone duration statistics table.

Table 6.27 Average phone durations

6.2.3.8 Speaker Specific Normalization

Speaker specific normalization features are shown at the Table 6.28.

AVG_PHONE_DUR_ZSP (6.73)

$$\frac{\sum_{every\ phone\ in\ word} phone_{zsp}\ [phone]}{number\ of\ phones}$$

MAX_PHONE_DUR_ZSP (6.74)

$$\max_{every\ phone\ in\ word} phone_{zsp}\ [phone]$$

AVG_PHONE_DUR_NSP (6.75)

$$\frac{\sum_{every\ phone\ in\ word} phone_{nsp}[phone]}{number\ of\ phones}$$

MAX_PHONE_DUR_NSP (6.76)

$$\max_{every\ phone\ in\ word} phone_{nsp}[phone]$$

Where,

$$phone_{zsp}[phone] = \frac{spkr\_phone\_dur\_mean[phone]}{spkr\_phone\_dur\_stdev[phone]} \quad (6.77)$$

$$phone_{nsp}[phone] = \frac{phone\_dur[phone]}{spkr\_phone\_dur\_mean[phone]} \quad (6.78)$$

phone_dur[phone] is phone duration for phone which has obtained from the feature WORD_PHONES and phone_dur_mean[phone] and phone_dur_stdev[phone] are computed at phone duration statistics table.

Table 6.28 Speaker specific normalization features

The features below are similar with phone duration features the difference is they are computed only over the vowels instead of every phone in the word.

AVG_VOWEL_DUR_Z	**MAX_VOWEL_DUR_Z**
AVG_VOWEL_DUR_N	**MAX_VOWEL_DUR_N**
AVG_VOWEL_DUR_ZSP	**MAX_VOWEL_DUR_ZSP**
AVG_VOWEL_DUR_NSP	**MAX_VOWEL_DUR_NSP**

Table 6.29 Vowel duration features.

6.3 Prosodic Feature Extraction

The Praat [6.10] and The Purdue Prosodic Feature Extraction Tool on Praat [31], [34] has been used to extract prosodic features. The input/output of that tool, the structure of the tool and the manipulations which has done in order to use the tool for Turkish spoken language will be discussed.

6.3.1 The Inputs and Outputs of the Prosodic Feature Extraction Tool

The Prosodic Feature Extraction Tool requires us three main inputs such as the audio file, phone and word alignments of the corresponding audio files with time labels. Either WAV or AIFF audio file formats can be used. We have used 16 bit 16 kHz WAV files. Additionally word and phone alignments should be (in Praat's *".TextGrid"* format) in the same directory with audio file. The word and phone alignments must be very sensitive, because of that the use of an ASR program has offered. We have used HTK, HVite tool (Details are at Chapter 5) in order to obtain 100 nanosecond sense alignments. Figure 6.1 shows either aligned phonemes, words and corresponding audio waveform signal and Table 6.30 shows the *".TextGrid"* format.

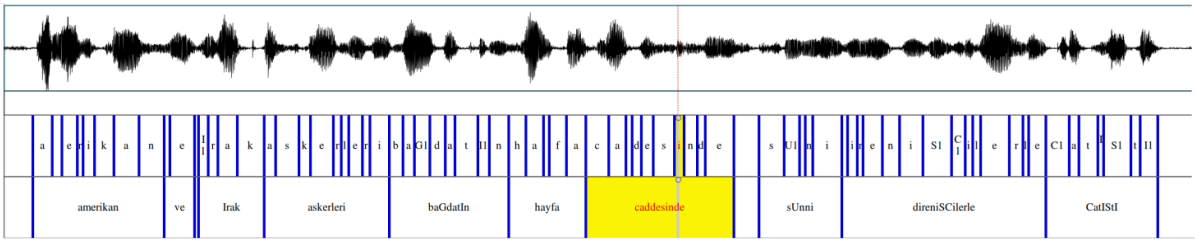


Figure 6.1 Audio waveform and corresponding phone and word alignments.

In phone alignments, one may use a pronunciation dictionary to represent words instead of vowel phone sets. By default CMU's dictionary or ISIP's dictionary is being used in the original file and capitalized phone labels are being supported. However for Turkish spoken

language, we have used a different suitable pronunciation dictionary which contains also lower phone labels. (See Chapter 5) In order to use our pronunciation dictionary and lower phone labels, we have modified the code *code/routine.praat.*

The pitch tracking and stylization functions of Praat are the most functional properties for the Prosodic Feature Extraction tool and pitch information is represented as raw pitch, voiced-unvoiced, stylized pitch and pitch slope. Autocorrelation based pitch tracking algorithm is being used in Praat in order to extract raw pitch values, using gender dependent pitch ranges. Furthermore, the pitch values can be further smoothed, stored and accessed to pitch values of each frame. [31]

The Prosodic Feature Extraction tool extracts the prosodic features in word based, i.e. all features of all single words.

However some of the features for some of the words cannot be extracted. In that case a question mark is printed. The output file seems to be a matrix where each row includes the features of the corresponding word, and each column includes the selected feature for all words. The detailed list of the features is given at section 6.2 [31].

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 7
tiers? <exists>
size = 1
item []:
   item [1]:
      class = "IntervalTier"
      name = ""
xmin = 0
xmax = 7
      intervals: size = 14
      intervals [1]:
xmin =  0
xmax =  0.15
            text =  ""
      intervals [2]:
xmin =  0.15
xmax =  0.83
            text =  "amerikan"
      intervals [3]:
xmin =  0.83
xmax =  0.99
            text =  "ve"
```

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 7
tiers? <exists>
size = 1
item []:
   item [1]:
      class = "IntervalTier"
      name = ""
xmin = 0
xmax = 7
      intervals: size = 76
      intervals [1]:
xmin =  0
xmax =  0.15
            text =  ""
      intervals [2]:
xmin =  0.15
xmax =  0.25
            text =  "a"
      intervals [3]:
xmin =  0.25
xmax =  0.3
            text =  "m"
      intervals [4]:
xmin =  0.3
xmax =  0.38
            text =  "e"
      intervals [5]:
xmin =  0.38
xmax =  0.41
            text =  "r"
      intervals [6]:
xmin =  0.41
xmax =  0.47
            text =  "I"
      intervals [7]:
xmin =  0.47
xmax =  0.57
            text =  "k"
      intervals [8]:
xmin =  0.57
xmax =  0.7
            text =  "a"
      intervals [9]:
xmin =  0.7
xmax =  0.83
            text =  "n"
      intervals [10]:
xmin =  0.83
xmax =  0.86
            text =  "v"
      intervals [11]:
xmin =  0.86
xmax =  0.99
            text =  "e"
```

Table 6.30 "Word.TextGrid" format shown at the left side and "Phone.TextGrid" format shown at the right side. Only first two words of figure 6.1 have shown.

6.3.2 Data Preparation

As it mentioned at the previous section, the audio waveform and corresponding *TextGrid* files should be at the same directory. The location of the files is generally *../demo/data/*and the waveform is saved as *demo_exapmle1.wav*, and the word and phone aligned files are saved as *demo_example1-phone.TextGrid* for the phoneme based aligned file and *demo_example1-word.TextGrid* for the word based aligned file. The time labels should be written instead of seconds. (ASR output has given us time marks in terms of 100 nanoseconds, so we converted them into seconds and also the ASR output converted into Praat TextGrid format by using a Perl script.) Additionally a metadata which is called *demo_wavinfo-list.txt* includes the information about sessions. An example is given at Table 6.1.

SESSION	SPEAKER	GENDER	WAVEFORM
demo_Speaker1	Speaker1	Female	../demo/data/demo_Speaker1.wav
demo_Speaker2	Speaker2	Female	../demo/data/demo_Speaker2.wav
demo_Speaker3	Speaker3	Male	../demo/data/demo_Speaker3.wav
demo_Speaker4	Speaker4	Male	../demo/data/demo_Speaker4.wav

Table 6.31 The Metadata. demo_wavinfo-list.txt [31]

The working procedure of the Prosodic Feature Extraction tool includes two main steps which are called *"Global Statistics Computation"* and *"Feature Extraction"*. There are two different statistics are computed which are called global and local statistics. There are several global statistics such as speaker dependent or independent statistics, specific phone duration statistics, pitch energy related statistics and the global phone duration statistics. All of the speaker independent statistics are computed for all speakers at all of the sessions before feature extraction step. The local statistics are session dependent statistics such as means and variances of the last rhyme duration, the last rhyme phone duration, the normalized last rhyme duration, and the pause duration. Those statistics are computed at feature extraction step. The statistics tables with definitions are shown at section 6.2.2 [31].

Figure 6.2 shows the procedure of feature extraction [34]. At the first step, the basic features and statistics are calculated, then by using the combination of the basic features and statistics, the derived features are also calculated.

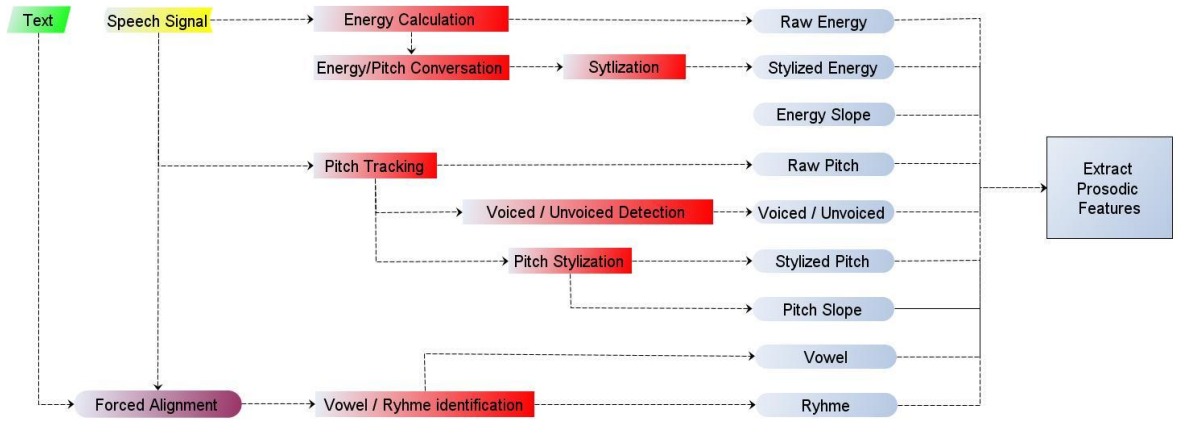
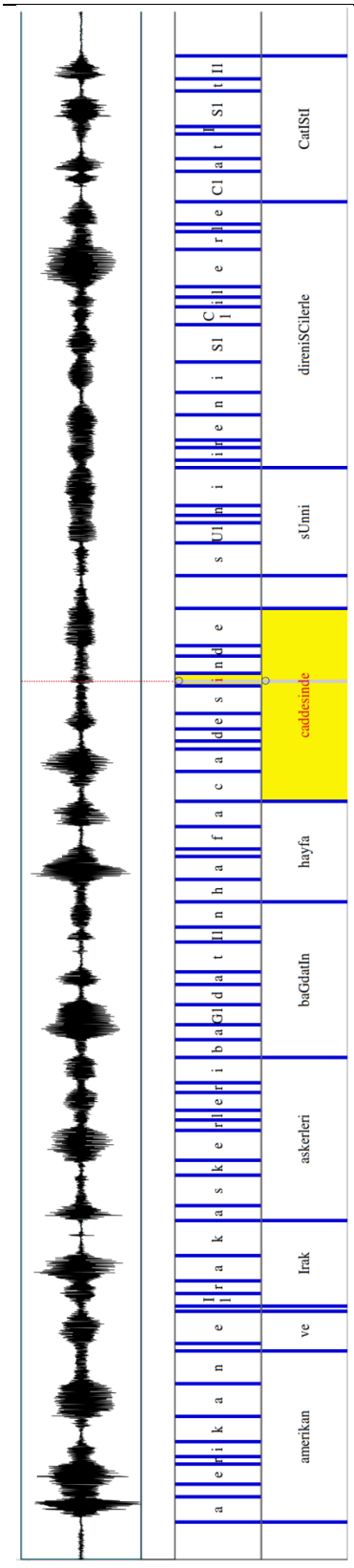


Figure 6.2 Prosodic Feature Extraction step [34].

6.3.3 Adapting the Prosodic Feature Extraction Tool for Turkish Spoken Language

The original version of the prosodic feature extracting tool is designed for English spoken language. Hence there were 40 phonetic units of the ARPAbet was defined on the prosodic feature extracting tool. Because of we have used the Turkish alphabet and phonetic representations for Turkish alphabet shown at Table 3.1, which is also used in ASR system, we have changed defined ARPAbet with phonetic unit representations shown in Table 3.1.



File type = “ooTextFile”

Object class = “TextGrid”

xmin = 0

xmax = 7

tiers? <exists>

size = 1

item []:

item [1]:

class = “IntervalTier”

name = “”

xmin = 0

xmax = 7

intervals: size = 14

intervals [1]:

xmin = 0

xmax = 0.15

text = “”

intervals [2]:

xmin = 0.15

xmax = 0.83

text = “amerikan”

File type = “ooTextFile”

Object class = “TextGrid”

xmin = 0

xmax = 7

tiers? <exists>

size = 1

item []:

item [1]:

class = “IntervalTier”

name = “”

xmin = 0

xmax = 7

intervals: size = 76

intervals [1]:

xmin = 0

xmax = 0.15

text = “”

intervals [2]:

xmin = 0.15

xmax = 0.25

text = “a”

intervals [3]:

xmin = 0.25

xmax = 0.3

6.3.4 Architecture of the Prosodic Feature Extraction Tool

There are two main steps at prosodic feature extraction procedure of the tool. The first step is called Global Statistics Computation, where the basic features (See section 6.2.1.) and statistic tables (See section 6.2.2.) are computed. And at the second step prosodic features are extracted by using the computed basic features and statistic tables. (See section 6.2.3.) The code organization of the tool is given at Table 7.1 and data flow diagram of the tool is given at Figure 6.3.

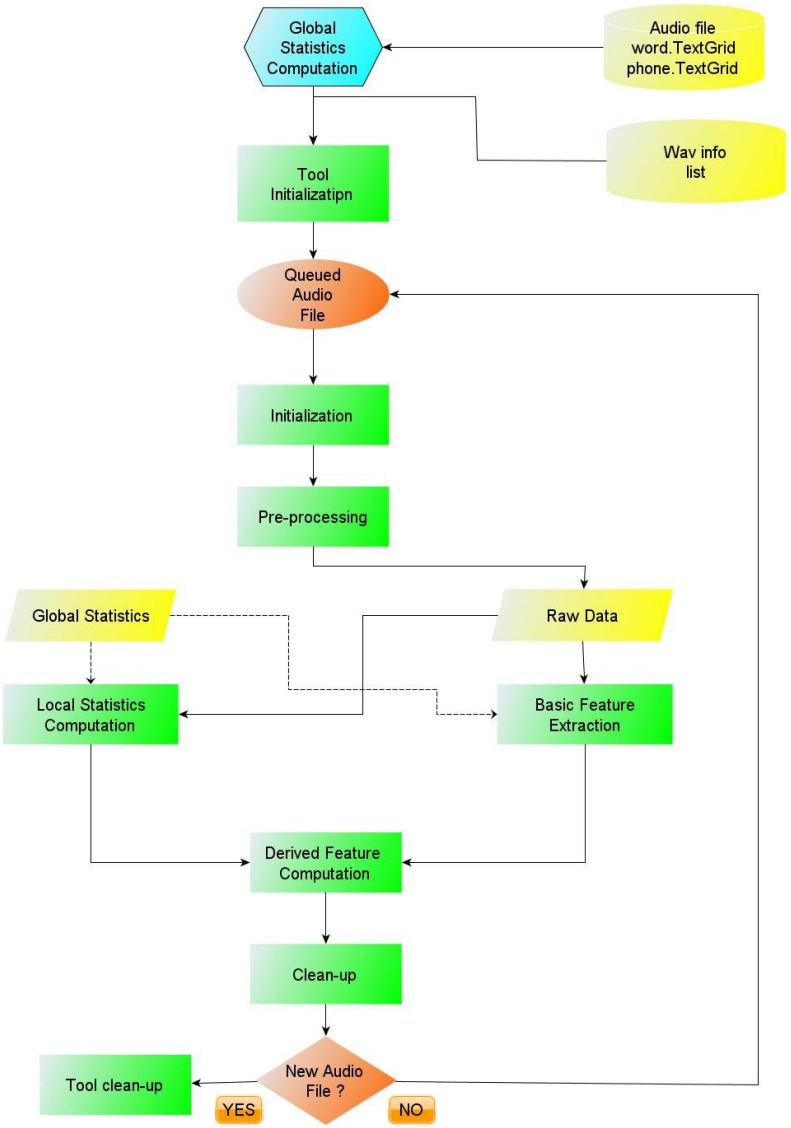


Figure 6.3 Data flow diagram of the Prosodic Feature Extraction tool. [31]

Scripts for Computing Global Statistics	
The codes below are found at the path: "code/stats/"	
stats_batch.praat:	The interface which accepts the inputs and controls the Global Statistics Computation operation.
operations.praat:	Highest level of the operation flow.
io.praat:	Controls input/output files.
table.praat:	Controls Praat Table operations, where various intermediate values and various operations such as table creation, value updating, etc. are handled.
stats.praat:	Routines for computing statistics.
routine.praat:	Routines for obtaining various basic elements.
utils.praat:	Contains some miscellaneous utility routines.
config.praat:	Configuration of the pre-defined parameter values, such as frame and window size.
Scripts for Extracting Prosodic Features	
The codes below are found at the path: "code/"	
main_batch.praat	The interface which accepts the inputs and controls the Feature Extraction operation.
operations.praat:	Highest level of the operation flow.
io.praat:	Controls input/output files.
table.praat:	Controls Praat Table operations, where various intermediate values and various operations such as table creation, value updating, etc. are handled.
fetch.praat	Higher level routines for extracting basic prosodic features by calling routines in routine.praat.
routine.praat:	Routines for obtaining various basic elements and lower level routines that extract prosodic features and supports higher level routines.
derive.praat	Routines to compute derived features.
utils.praat	Contains some miscellaneous utility routines.
config.praat:	Configuration of the pre-defined parameter values, such as frame and window size.
pf_list_files/feature_name_table.Tab	List of feature names.

Table 6.32 Code Organization of the Prosodic Feature Extraction tool. [31]

6.3.5 Using the Prosodic Feature Extraction Tool

At the first step, the wave file and corresponding word and phone aligned files which are shown at Figure 6.1 (See section 6.3 for details) should be loaded into the path: "*demo/work_dir/"*. Additionally an audio information table which is shown at Table 6.31 should be prepared which includes names of all of the audio files and corresponding word and phone aligned files. After finishing those preparations, the program can be run either by using command window or user interface. Table 7.2 shows the steps to run the program.

The table which includes the extracted prosodic features can be found at the following directory. "*demo/work_dir/pf_files/"* By using notepad, the file can be viewed. The structure of the table is; there are rows for each word where each row includes each prosodic feature for the given word. One can also view the output table file by using *open office calc*or by using *Microsoft office excel* programs. Table 6.34 shows an example of an output prosodic features table.

Table 6.33 shows the structure of the output Prosodic Feature Table. There are 266 features are extracted, the definition of the features are given at Chapter 6, at each word boundary.

WORD	WAV	SPKR_ID	GEN	WORD_START	Feature Names	Last Feature
Word 1	location	ID	gender	feature	feature	feature
Word 2	location	ID	gender	feature	feature	feature
Word 3	location	ID	gender	feature	feature	feature
Word n	location	ID	gender	feature	feature	feature

Table 6.33 Prosodic Feature Table

- **Global Statistics Computation**

Code:

praatstats_batch.praat ../demo-wavinfo_list.txt ../demo/work_dir yes

Using Interface

1. Run *Praat*
2. *Praat Objects / Read / Read from file /* select *stats_batch.praat*
3. Click *Run* on the menu of *Script Editor.*
4. Type *../demo-wavinfo_list.txt* and *../demo/work_dir* into the boxes and click *yes* if you want to use existing parameter files, *no* to generate parameter files from the beginning.
5. Click *OK*
6. Process is displayed in the *Praat Info* Window.

- **Prosodic Feature Extraction**

Code:

praatmain_batch.praat ../demo-wavinfo_list.txt user_pf_name_table.Tab
../demo/work_dir/stats_files ../demo/work_dir yes

1. Run *Praat*
2. *Praat Objects / Read / Read from file /* select *main_batch.praat*
3. Click *Run* on the menu of *Script Editor.*
4. Type *../demo-wavinfo_list.txt, user_pf_name_table.Tab, ../demo_work_dir/stats_files, and ../demo/work_dir*respectively into dour boxes.
5. Click *OK*
6. Process is displayed in the *Praat Info* Window.

Table 6.34 The use of theProsodic Feature ExtractionTool [31]

Chapter 7

Sentence Boundary Detection

By

Using Prosodic Features and Learning Algorithms

7.1 Sentence Segmentation Problem

The output of the Automatic Speech Recognizer system is just the utterance of the words. Even for humans, it is very hard to understand a long message which includes only utterance of words. However when there is punctuation at the written text, it is easier to understand the meaning of the written text. For instance two examples below with and without sentence boundary signs are given for English written language. Without denoting sentence boundaries it is hard to understand the message but with sentence boundary notations it is easier to understand the meaning of the written text.

Because of that reasons sentence segmentation is the fundamental steps of the speech processing applications such that topic segmentation and summarizing applications.

In the previous chapter, the prosodic feature extraction procedure and the structure of the output data which is called prosodic feature table (Table 6.33) has briefly explained. At table 6.33, the first column shows the words and the rest columns show the corresponding prosodic feature sets. Suppose that we have *N* words. We can denote each of the words as

"w_i", where "$i=1,...,N$", and each of corresponding feature sets i.e. feature observations "f_i", where "$i=1,...,N$". The sentence segmentation problem is at each word boundary "b_i" ,which is between the words w_i and w_{i+1} , to estimate if the boundary is a sentence boundary or not, given feature observation sets. A posterior probability of the word is either sentence or non-sentence given feature sets are computed and compared to obtain final decision. Table 7.1 formulizes the sentence segmentation problem.

Example 1: Without sentence boundaries.
the curta is a small hand cranked mechanical calculator introduced in nineteen forty eight it has an extremely compact design a small cylinder that fits in the palm of the hand it can be used to perform addition subtraction multiplication division and with more difficulty square roots and other operations this little article taken from wikipedia in this article the calculator curta is described

Example 1: With sentence boundaries.
the curta is a small hand cranked mechanical calculator introduced in nineteen forty eight **(s)** it has an extremely compact design **(s)** a small cylinder that fits in the palm of the hand **(s)** it can be used to perform addition subtraction multiplication division and with more difficulty square roots and other operations **(s)** this little article taken from wikipedia**(s)** in this article the calculator curta is described **(s)**

7.2 Supervised and Semi-Supervised Learning Algorithms

The sentence segmentation approach has introduced at the previous section. The aim is to detect the sentence boundaries i.e. automatically label each word boundaries whether they are sentence boundary or not. For each word, the posterior probabilities of either if it is a word boundary or not are computed and compared. Hence basically at that point a binary classification has performed. Nevertheless we have only one classifier and unless it is built with a high accurate prediction rule, which is a very difficult task, it outperforms alone.

On the other hand the main idea of using learning algorithm is that, obtaining a strong classifier by combining several classifiers each performs a little better than random

guessing will be easier and perform better. In other words, this is called a weak learning algorithm. During many rounds at each round, different subsets of training examples are taken and a new prediction rule is generated. At the end those weak rules are combined to obtain a single and strong prediction rule [40]. Figure 7.1 shows the idea very basically.

w_i	$i=1,\ldots,N$	The word sequence.
f_{ij}	$i=1,\ldots,N$ and $j=1,\ldots,M$	"f*i*" denotes the feature set of the corresponding word and "f_j" denotes a feature for all words.
b_i	$i=1,\ldots,N$	$b_1,\ldots,b_{N-1}$ is the word boundary between w_i and w_{i+1} and b_N is the last word boundary.
y_i	$y_i \in \{+1,-1\}$	When $k = 1$, it is a sentence boundary. It is labeled by "*s*" and $k = -1$ is not a sentence boundary and it is labeled by "*n*".
$P(b_i = y_i \mid f_i)$	Posterior probability of sentence boundary/non sentence boundary where y = 1 / y = -1.	
$b_i = s$	$P(b_i = +1 \mid f_i) > P(b_i = -1 \mid f_i)$	Sentence Boundary
$b_i = n$	$P(b_i = -1 \mid f_i) > P(b_i = +1 \mid f_i)$	Non-sentence Boundary.

Table 7.1 Sentence Segmentation Approach[38]

At below, at first the method of dividing whole data set into three main subsets such as training set, development set i.e. validation set and test set. Then several weak learning algorithms will be discussed and different techniques will be used about how to select a subset of training examples and how to combine weak prediction rules.

7.2.1 Dividing the Data into Subsets

At the beginning, we should divide all of our data sets into three main subsets; such as training set, development set i.e. validation set and test set. All of three main subsets have their own purposes. The training set is used to optimize the parameters such as weights of used learning algorithm. The validation set is used to optimize the hyper parameters such as

number deciding number of iteration and finally test set is used to measure final score of the optimized model [42]. To divide whole dataset into three main subsets, classification algorithms are being used. Classification algorithms are divided into two categories such as cross-validation and resampling methods.

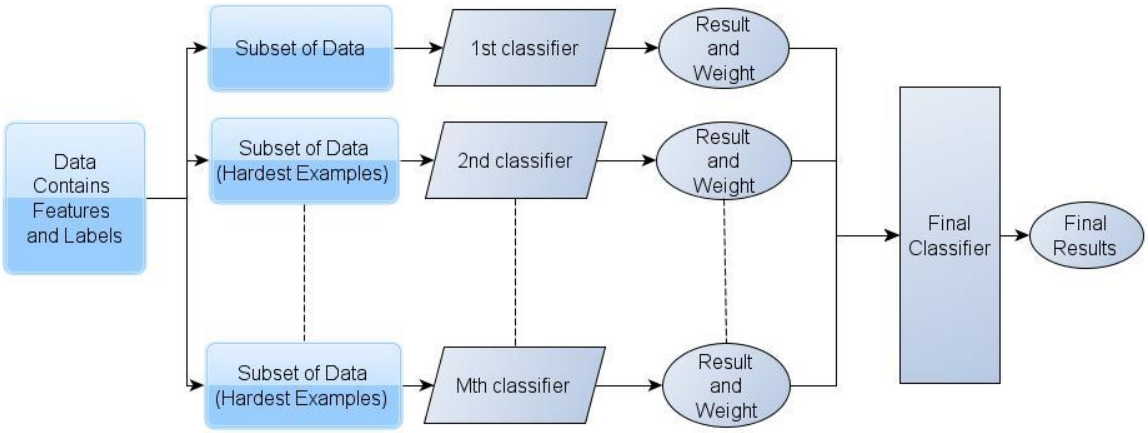


Figure 7.1 Basically the idea of a Learning Algorithm [39].

At the first step, we obtain the train set and development set. In our dataset we have P speakers. For each speaker we have N words and for each word we have M features. At the beginning, we are dividing our dataset into the speaker based sub datasets and we obtain P sub datasets. Then for each P sub datasets we select training/development set pairs,$\{T_i, D_i\}_{i=1}^{L}$, where L≤N. Furthermore while selecting the training/development set pairs, we do not spoil order of the words in a complete sentence. Hence we obtain independent and identical distributed train and development sets at the end. For speaker based tests, we do not combine train, development and test sets belongs to each speaker at the end, but for non-speaker based tests we will combine them.

There are some other techniques to select the train set and test set such as K-Fold Cross-Validation, 5×2 Cross-Validation, which are Cross-Validation techniques and Bootstrapping which is a resampling method.

K-Fold Cross-Validation method offers to divide whole dataset into K subsets at first. $X_i = \{f_i, y_i\},\ i = 1, \dots, K$. Then select $X_i$as development set $D_{i,}$ and combine rest of subsets

as training set T_i. This method has two disadvantages such as this method keeps the development set small and causes overlap between training and development sets.

5×2 Cross-Validation method Dietterich, et. al. similar with K-Fold Cross-Validation method. The data is divided into K subsets at first such that each subset has a sequence of word features and corresponding labels. $X_i = \{f_i, y_i\},\ i = 1, \dots, K$. However selecting training and development set differs. 5×2 Cross-Validation method offers to divide X_i into two groups such as $X_i^{(1)}$and $X_i^{(2)}$. For each i th fold $X_i^{(1)}$ becomes the test set and $X_i^{(2)}$ becomes the development set. Then those sets changes roles for each i+1 th fold where $X_i^{(1)}$ becomes the development set and $X_i^{(2)}$ becomes the training set. Dietterich points that it would be enough to perform 5 folds because after 5 folds there would be overlap between the sets and error rates will become dependent. Hence no further information will be added [42].

7.2.2 Boosting Algorithm

The boosting algorithm was first proposed in 1989 by Freund and Schapire. The algorithm has three main steps and a final evaluation step as follows. At the first step a subset of the training set is selected randomly and first weak learner is trained. At the second step, the samples which were not selected in the first subset and half of the misclassified samples of first weak learner are selected and second weak learned trained. At third step, all samples that first and second weak learners disagree on are selected and third weak classifier is trained. Finally at last step, those three weak learners vote equally to obtain final classifier. Table 7.2 shows the algorithm.

- **Boosting Algorithm**

Input:	$X = \{x_1, x, \dots, x_N\},\ where\ \ x_i = (f_i, y_i)$as training set.
Output:	$H(x)$, final strong classifier.

- $N_1 < N$ samples are selected randomly from *X* to obtain X_1.
- Train weak learner $h_1$on X_1.
- $N_2 < N$ samples with half of misclassifies samples by h_1 are selected to obtain X_2.
- Train weak learner h_2 on X_2.
- Select all samples *X* that h_1 and $h_2$disagree on to obtain X_3.
- Train weak learner $h_3$on X_3.
- Final classifier is obtained by equal voting of three classifiers as shown in Equation 7.1

$$H(X) = sign(\sum_{n=1}^{3} h_n) \tag{7.1}$$

Table 7.2: Boosting Algorithm [39]

7.2.3 AdaBoost Algorithm

The Adaptive Boosting (AdaBoost) algorithm has developed by Freund and Schapire in 1996 [40], shown at Table 7.3. There are two major differences from the first boosting algorithm such as choosing samples for the next iteration and weighted combination of weak classifiers are combined to obtain final strong classifier.

7.2.3.1 The Algorithm

At the beginning we assume that all of the samples are equally important. Hence as shown at first step of the algorithm we have a uniform distribution of weights corresponding to the samples at the beginning. At the second step of the algorithm, we train a weak classifier

which performs a little bit better than random guessing. The maximum error of the first and all classifiers are shown at Equation 7.2 which is calculated at the third step.

$$\epsilon_t = 0.5 - \gamma_t, \qquad where, \qquad \lim_{\gamma_t} \to 0 \tag{7.2}$$

At the fourth step, Equation 7.6 and Equation 7.4,5 are performed respectively. At Equation 7.4,5, the weight of the weak classifier is being calculated. Figure 7.2 shows the relation between the weight and error based on Equation 7.5

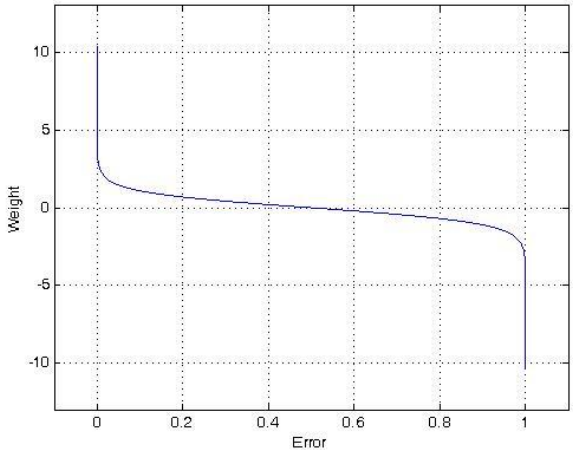


Figure 7.2 The weights versus errors where$\gamma_t = 10^{-9}$.

At Equation 7.7 we see that while final strong classifier is obtained, the weak classifiers which performs better gets higher scores than the weak classifiers which outperforms. On the other hand, at the each iteration the importance i.e. weights of correctly classified samples decreases and the weights of misclassified samples are increased. Equation 7.4,5 shows the weights of the samples for the next iteration. The reason of that, once a set of sample is classified correctly, at the next iteration instead of those samples, hardest samples which were misclassified should be considered and following weak learner should focus on to classify that samples.

- **AdaBoost Algorithm**

Input:	$X = \{x_1, x, \dots, x_N\},\ where\ \ x_i = (f_i, y_i)$as training set.
Output:	$H(x)$, final strong classifier.
Index "i"	Index of samples, i.e. subset of words in training set.
Index "t"	Index of iteration, i.e. number of combined weak classifiers.

➢ For t=1,…,T Do:

1. $D_1(i) = \frac{1}{n}$, where D_t is current distribution.

At the beginning all of the samples weights are uniformly distributed.

2. Train weak learner h_t on *X*.

$h_t : X = \{-1, +1\}$

3. Calculate the error probability, i.e. the probability of a sample has misclassified.

$$\epsilon_t = P_{Dt}(h_t(x_i) \neq y_i) \tag{7.3}$$

4. Update distribution of the next weak learner. Reduce the weight of correctly classified samples and increase the weight of misclassified samples.

$$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} \ if\ y_i = h_t(x_i) \\ e^{\alpha t} \ if\ y_i \neq h_t(x_i) \end{cases} \tag{7.4}$$

$$= \frac{D_t(i) exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{7.5}$$

Where

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{7.6}$$

And Z_t *is normalization constant.*

5. Repeat steps 2, 3, 4 until t = T.

6. Obtain the final strong classifier.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha t \times h_t(x)\right) \tag{7.7}$$

Table 7.3 AdaBoost Algorithm [40]

While doing more and more iterations on boosting algorithm, we observe two types of errors such as train error and test error, i.e. generalization error. The main difference between the train and test error is that, while increasing the number of iterations, train error will decrease and after certain iteration it will be always zero. But test error is different. While increasing the number of iteration actually we are combining same number of weak classifiers. After a certain number of classifiers, i.e. T classifiers, adding more iteration, i.e. combining T + t classifiers causes overtraining, i.e. we have too complex classifier. Hence the test error starts to increase after that certain number of iterations.

7.2.3.2 Analyzing the Training Error

AdaBoost has ability to reduce its training error [40], i.e. the mistakes on the training set reduced while adding more iteration. At Equation 7.4 and 7.5 we have defined the Z_t as a constant. The value Z_t shown at Equation 7.8 stands for an upper bound, which assumes at the current iteration all of the samples are misclassified.

$$Z_t = \sum_{i=1}^{N} D_t(i)\ exp\left(-\alpha_t y_i h_t(x_i)\right) \quad (7.8)$$

where

$$exp\left(-\alpha_t y_i h_t(x_i)\right) \geq 1\ if\ y_i \neq H(x_i)$$

The training error of final classifier is shown with an upper bound at Equation 7.9. The left side of the equation shows the actual error and right side of the equation bounds the actual error.

$$\frac{1}{m}|i : H(x_i) \neq y_i| \leq \frac{1}{m}\sum_{i=1}^{m} exp\left(-y_i f(x_i)\right) = \prod_t Z_t \quad (7.9)$$

where

$$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t \quad (7.10)$$

When we consider the Equation 7.8, we see that adding more iteration will decrease the value ofZ_t. Hence while adding more iteration the right side of the Equation 7.9 will converge to zero. Moreover the product of Z_t values is bounded by combining equations 7.2, 7.6, 7.9 and 7.10 which is shown at Equation 7.11 [40].

$$\prod_t Z_t = \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4{\alpha_t}^2} \leq exp\left(-2\sum_t {\gamma_t}^2\right) \tag{7.11}$$

7.2.3.3Analyzing the Test Error (Generalization Error)

While adding more iteration, it is shown above that the train error increases and finally goes to zero. However, test error i.e. generalization error behaves different. The reason is that, the complexity of the final classifier increases while increasing number of iteration. After a certain point the final classifier becomes a too complex classifier and it causes more errors. Robert Schapire, et al. described the generalization error based on margins and an upper bound based on VC-dimension (Vapnik Chernonenkis Dimension) of generalized error.

The margins are defined as the difference between the weighted sum of hypotheses voting for the right label and weighed sum of hypotheses voting for the wrong label as shown in the Equation 7.12. The left side of the final subtraction is the sum of hypotheses for the right label and right side of the subtraction is the sum of hypotheses for the wrong label.

$$\begin{aligned} margin(x,y) = y.f(x) = y.\sum_{t=1}^{T}\alpha_t h_t(x) = \sum_{t=1}^{T}\alpha_t y h_t(x) \\ = \sum_{t:h_t(x)=y}\alpha_t - \sum_{t:h_t(x)\neq y}\alpha_t \end{aligned} \tag{7.12}$$

Schapire, et. al. showed an upper bound of generalization error based on VC-dimension. The final classifier is constructed by weighted votes of weak classifiers.

Let Ȟ = $\{all\ functions\ of\ form\ sign(\sum_{t=1}^{T}\alpha_t h_t(x))\}$

If final classifier $H(x)$ is consistent and $\left|\check{\mathrm{H}}\right|$ finite, then with probability $1-\delta$

$$err\big(H(x)\big) \leq \widehat{err}\big(H(x)\big) + O\left(\sqrt{\frac{ln\prod_{\check{\mathrm{H}}}(2m) + \ln(1/\delta)}{m}}\right) \tag{7.13}$$

Where the left side of the inequality represents the generalization error of the final classifier and right side represents a bound. The first term of the bound is defined as the upper bound of the training error which has shown at Equation 7.11. The second term in the bound related with the complexity of the final classifier. As shown in that expression, number of features in each sample which has denoted by m, and number of iteration which has represented by $\check{\mathrm{H}}$ affect the complexity of the final classifier.

The complexity expression of Equation 7.13 derived as follows. It is assumed that the value of weak hypothesis space **H**does not diverges and all of either feature sets x_t or weak classifiers h_t are fixed. The final strong classifier is defined as follows.

$$H(\boldsymbol{x}_i) = sign(\boldsymbol{\alpha}.\boldsymbol{z}_i)$$
$$where\ \boldsymbol{\alpha} = \langle \alpha_1\alpha_2, \dots, \alpha_T\rangle\ and\ \boldsymbol{z}_i = \langle h_1(x), h_2(x), \dots, h_T(x)\rangle \tag{7.14}$$

Since there are m different inputs, i.e. features, and $sign(\boldsymbol{\alpha}.\boldsymbol{z}_i)$ has VC-dimension *T*, by Sauer's Lemma, the number of conflicts between weak classifiers are bounded as follows.

$$number\ of\ conflicts \leq \sum_{t=0}^{T}\binom{m}{i} \leq \left(\frac{em}{T}\right)^T \tag{7.15}$$

According to the Equation 7.15, *T* weak classifiers have no more than $\left(\frac{em}{T}\right)^T$ conflicts and there are $|\mathbf{H}|^T$ choices for all of the weak classifiers. Hence,

$$\prod\nolimits_{\check{\mathrm{H}}}(m) \leq |\mathbf{H}|^T\left(\frac{em}{T}\right)^T \tag{7.16}$$

By combining Equations 7.13 and 7.16 the boundary expression in Equation 7.17 is obtained.

$$err\big(H(x)\big) \leq \widehat{err}\big(H(x)\big) + O\left(\sqrt{\frac{Tln|H(x)|+Tln\left(\frac{m}{T}\right)+\ln\left(\frac{1}{\delta}\right)}{m}}\right), \forall H(x) \in \check{\mathrm{H}} \qquad (7.17)$$

With probability at least $1-\delta$

In Equation 7.17 it is clearly shown that while *T* increases, the complexity of *H(x)* is also increasing.

7.3 Model Training Procedure

There are four input files of Icsiboost [41] such as training set, development set, test set and a file which points the features. The key idea is, at the first step training a model by using the test set, then testing the trained model at development set to find optimum iteration number by doing and comparing performance evaluations and finally using the trained model with optimum number of iteration and evaluating the performance at test set. The details of the given procedure will be discussed with details but first the inputs outputs and usage of Icsiboost will be explained.

7.3.1 Labeling the Sentence Boundaries

As mentioned above at the beginning we must have a train set, development set and test set which includes the selected features according to the test and corresponding sentence boundary labels. To obtain those data sets at the first step we are labeling the word boundaries of prosodic feature output table (Table 6.33). We are adding one more column to Table 6.33 at first, then we are putting “n” to word boundaries and “s” to sentence boundaries while listening the original audio file and we obtain Table 7.4.

Words	Feature Names (f_{ij}) Columns 1 to M	Label (b_i) Column M+1
Word 1	Features f_{1j} *where j=1,…,M*	n
Word 2	Features f_{2j} *where j=1,…,M*	n
…	...	…
Word i	Features f_{ij} *where j=1,…,M*	n
…	...	…
Word n	Features f_{nj} *where j=1,…,M*	s

Table 7.4 Labeled Prosodic Feature Table

At the second step we are removing the first column which includes the words and we obtain our data by taking only interested features, i.e. the columns corresponding to interested features and word labels. At the third step we are dividing our new data into three parts such as train set, development set and test set respectively. Furthermore we also divide the train set into subsets.

Finally we save those three data sets as follows. Suppose for the first test we have given the name of the test as "example1". So we save the training, development at test sets as "example1.data", "example1.dev" and "example1.test" respectively which shown at Table 7.5

7.3.2 Addressing the Features

In previous section, it is mentioned that one could train a model by using only interested features. However the train, development and test sets are not only enough to train a model at Icsiboost but also a file which includes the names of the features and possible values of the corresponding features.

Selected Features (f_{ij}) Columns 1 to M	Labels (b_i) Column M+1	Data Sets
Features f_{ij} *where j=1,…,M and i=1,…,q*	*s or n*	Training set Size = (q, M+1) Example1.data
Features f_{ij} *where j=1,…,M* *and i=q,…,q+p*	*s or n*	Development set Size = (p, M+1) Example1.dev
Features f_{ij} *where j=1,…,M* *and i=q+p,…,q+p+r*	*s or n*	Test set Size = (r, M+1) Example1.test

Table 7.5 Icsiboost input data formats.

There are three types of features such as continuous valued features, label valued features such as pointing a rising slope or falling slope in energy and a sequence of features. In our tests, generally we are considering either continuous valued features or label valued features. Hence we are preparing a file which guides the features and we save it as "Example1.names" to the same directory with data sets of that experiment. The structure of the guide file is shown at Table 7.6.

s,n.
Feature1: ,continuous.
Feature2: ,continuous.
Feature3: f,r.
…

Table 7.6 Structure of Experiment1.names.

The first line of the ".names" (Table 7.6) file is the b_i where it is an "s" if the word is a sentence boundary or an "n" if the word is a non-sentence i.e. only word boundary. In this line we define trainer to label what. At the following lines, we are listing the features

according to the actual order in data sets and if a feature is continuous valued, we define it as continuous and if it is discrete or choices of some labels, we put all possible outcomes.

7.3.3 Training a Model by Using Icsiboost

In previous sections we have defined the inputs of Icsiboost. At this section training a model by using Icsiboost will be described. The input files should be in the same directory and model is trained by typing the code shown at the upper part of Table 7.7. Once the code has run, the following lines with similar parameters will appear in the "parameters.txt" file. In the lines after command shown at Table 7.7 there are five types of parameters such as weighted error, theoretical error, development error, test error and train error. At the first column, weighted error is stands for the weights of each trained weak classifier. The calculations of weights are shown at Equation 7.4 and 7.5. At the second column there is theoretical error which is calculated by using Equation 7.6. The third and fourth columns are the results of analyzing test error on development set and test set respectively and finally the last column is the result of analyzing the train error.

- *icsiboost –S Example1 –n #iteration> parameters.txt*

rnd 1: wh-err= 0.274495 th-err= 0.274495 dev= 0.032311 test= 0.041042 train= 0.034448
rnd 2: wh-err= 0.767378 th-err= 0.210641 dev= 0.031609 test= 0.030899 train= 0.024963
rnd 3: wh-err= 0.820982 th-err= 0.172933 dev= 0.031609 test= 0.030899 train= 0.024963
rnd 4: wh-err= 0.883527 th-err= 0.152791 dev= 0.041676 test= 0.033084 train= 0.024963
rnd 5: wh-err= 0.923448 th-err= 0.141094 dev= 0.030906 test= 0.029806 train= 0.020469
rnd 6: wh-err= 0.930887 th-err= 0.131343 dev= 0.030204 test= 0.029338 train= 0.020469
rnd 7: wh-err= 0.947551 th-err= 0.124454 dev= 0.030204 test= 0.029338 train= 0.020469
rnd 8: wh-err= 0.942822 th-err= 0.117338 dev= 0.029267 test= 0.029806 train= 0.020469
rnd 9: wh-err= 0.959168 th-err= 0.112547 dev= 0.033013 test= 0.029806 train= 0.018472
rnd 10: wh-err= 0.960902 th-err= 0.108146 dev= 0.033013 test= 0.029963 train= 0.017973
…

Table 7.7 Training a model by using Icsiboost.

In the lines after command shown at Table 7.7 there are five types of parameters such as weighted error, theoretical error, development error, test error and train error. At the first column, weighted error is stands for the weights of each trained weak classifier. The calculations of weights are shown at Equation 7.4 and 7.5. At the second column there is theoretical error which is calculated by using Equation 7.6. The third and fourth columns are the results of analyzing test error on development set and test set respectively and finally the last column is the result of analyzing the train error.

As it mentioned previous sections, while adding more iterations we expect that the weights will increase, and either theoretical or train error will decrease. On the other hand we have test error and development error. After a certain number of iteration, the model will become too complex i.e. over-train. While adding more iteration the development and test error decrease until an optimum number of iteration, after that number that errors will start to increase.

7.3.4 Testing Performance of the Trained Model

Once a model has trained by typing the command shown at Table 7.8 the results of each word in the either development or test set according to your command will be appear in the "results.txt" file.

There are four columns shown at Table 7.8 where each row stands for one word in either development or test set. The first two columns are written according to human labels b_i such as "s" i.e. a sentence boundary and "n" i.e. non-sentence boundary. The sequence {0 1} stands for human labeled "n" and {1 0} stands for human labeled "s". While showing human labels Icsiboost uses [0, 1] instead of using [-1, 1]. The following two columns stand for the results of the binary classifier. The sign of the parameters are the decision of the classifier and magnitude of the parameters stands for the confidence measure. In those two columns {-0,xxx 0.xxx} stands for a non-sentence boundary decision of the classifier

and {0.xxx, -0.xxx} stands for sentence boundary decision. At the following step we will compare those two pairs and do performance evaluations.

- *icsiboost –S Example1 –C < Example1.dev > results.txt*
- *icsiboost –S Example1 –C < Example1.test > results.txt*

```
0 1 -0.601237747037 0.601237747037
0 1 -0.505622123388 0.505622123388
0 1 -0.505622123388 0.505622123388
0 1 -0.283534787175 0.283534787175
0 1 -0.899572923912 0.899572923912
0 1 -0.280858442462 0.280858442462
0 1 -0.505622123388 0.505622123388
0 1 -0.516929451725 0.516929451725
0 1 -0.601237747037 0.601237747037
0 1 -0.505622123388 0.505622123388
0 1 -1.014745536400 1.014745536400
0 1 -0.781723966712 0.781723966712
0 1 -0.559636630500 0.559636630500
0 1 -0.899572923912 0.899572923912
0 1 -0.283534787175 0.283534787175
0 1 -0.674809242987 0.674809242987
1 0 0.155911534538 -0.155911534538
```

Table 7.8 Results of binary classification on either development or test set.

7.3.5 Analyzing Performance of the Trained Model

In the previous section we have explained that the "result.txt" includes a list of actual class of the word (sentence boundary or non-sentence boundary) and estimated boundaries by the classifier. According to the comparisons at each line there are four outcomes. Either estimator result or human label may agree that the word is a sentence boundary or non-sentence boundary. On the other hand, they may disagree. All of the four probable

outcomes have their own terms such as True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), which are shown at Table 7.9.

Decision Table	$y_{i=1}\ i.e.\ y_i = s$ {1, 0}	$y_{i=0}\ i.e.\ y_i = n$ {0, 1}
$H(x) = 1\ i.e.\ b_{i=S}$ {0.xxx, -0.xxx}	True Positive	False Positive
$H(x) = -1\ i.e.\ b_{i=n}$ {-0.xxx, 0.xxx}	False Negative	True Negative

- True Positive (TP): Represents correctly labeled sentence boundary.
- True Negative (TN): Represents correctly labeled non-sentence boundary.
- False Positive (FP): Unexpected sentence boundary, i.e. it is actual a non-sentence boundary but classifier recognized it as a sentence boundary.
- False Negative (FN): Missing sentence boundary, i.e. it is actual a sentence boundary but classifier recognized it as a non-sentence boundary.

Table 7.9 Four outcomes of each boundary are explained.

Before starting to do the performance evaluations we count number of TN, TP, FN and FP, where sum of them is the number of all words in the development/test set. In performance evaluation we will consider two scores which are F-measure score and Nist Error Rate. However to calculate F-measure, also Precision and Recall should calculated. All of the performance evaluation metrics with formulas are given below.

- **Precision**

Precision is one of the most commonly used and well known performance evaluation measure which implies repeatability of the system. As shown in Equation 7.16, precision measures the ratio between correctly labeled sentence boundaries and all sentence boundary decisions of the classifier. The calculation of Precision has shown at Equation 7.16. The minimum value of precision is zero when there is no correctly detected sentence boundary

and the maximum value of precision is one when there is no unexpected sentence boundary.

$$Precision = \frac{TP}{TP+FP}, \quad Precision \, \epsilon \, [0,1] \tag{7.16}$$

- **Recall**

Recall is also one of the most commonly used and well known performance measure, which is the ratio of correctly labeled sentence boundaries by the classifier over all of the actual sentence boundaries. As shown at Equation 7.17 the minimum value of Recall is 0 in the case absence of correctly labeled sentence boundaries and the maximum value of Recall is 1 when there is no missing actual sentence boundary by the classifier.

$$Recall = \frac{TP}{TP+FN}, \quad Recall \, \epsilon \, [0,1] \tag{7.17}$$

- **True Negative Rate**

True negative rate measures the performance of classifying non-sentence boundaries. The minimum value of True Negative Rate is zero in the case absence of correctly classified non-sentence boundaries and the maximum value of True Negative Rate is 1 when there are no unexpected sentence boundaries.

$$True \, Negative \, Rate = \frac{TN}{TN+FP}, \quad True \, Negative \, Rate \, \epsilon \, [0,1] \tag{7.18}$$

- **Accuracy**

Accuracy is one of the most commonly used and well known performance evaluation measure, which measures the ratio of classifiers true decisions over all of the decisions given by the classifier. The minimum value of Accuracy is 0 when there are no correct decisions given by the classifier and the maximum value of Accuracy is 1 when there are no wrong decisions given by the classifier.

$$Accuracy = \frac{TP+TN}{TN+TP+FN+FP}, \quad Accuracy \, \epsilon \, [0,1] \tag{7.19}$$

- **F-measure Score**

F-measure score is one of the most commonly used and well known performance evaluation measures. It is a measure of test's accuracy in terms of harmonic mean of precision and recall. The minimum value of F-measure approaches to 0 when the limit of sum of the Precision and Recall values approaches to zero from positive. The maximum value of F-measure is 1 when either precision or recall is 1. The advantage of using F-measure is that it is a combination of precision and recall.

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}, \quad F - measure \in [0, 1] \tag{7.20}$$

- **Nist Error Rate**

Nist (National Institute of Standards and Technology) error rate is one of the most commonly used and well known performance evaluation measures. The most significant difference between previous given performance evaluations is that, Nist Error Rate is aimed to minimize while other measures are aimed to maximize. The Nist Error Rate is the ratio of all misclassified samples i.e. all of the misclassified either sentence or non-sentence boundaries, over all of the actual sentence boundaries. The minimum value of Nist Error Rate is zero when there is no mistake done by the classifier and maximum value is the ratio sum of all mistakes done by the classifier over number of missing sentence boundaries, where in the worst case we assumed that there are no correctly labeled actual sentence boundaries.

$$Nist\ Error = \frac{FN + FP}{TP + FN}, \quad F - measure \in [0, k]\ ,\ k \in R^{+} \tag{7.21}$$

7.3.6 Maximizing the Performance

In previous sections, the use of icsiboost and analysis of outputs has mentioned. One should note that, while increasing iteration number either train error or test error varies hence it implies that the performance will also vary while iteration number is varying. Indeed, the performance will be maximized in optimum number of iterations. Below, the procedure of finding optimum number of iterations will be described.

As shown in Table 7.5, the whole data set has divided into three subsets such as training set, development set and test set. To find the optimum number of iteration, we are training the model by using the training set, testing the trained model by using the development set and we record F-measure score and Nist error, for each number of iteration. Then each iteration scores are compared. Maximum F-measure score and minimum Nist error is required for the highest performance. After doing comparisons the training set is trained by using the optimum number of iteration and performance is tested and evaluated by using the test set. Table 7.11 shows whole procedure.

For i =1,…,N

Where we are sure that model will over train with N iterations.

1. Train the model.
   icsiboost –S Example1 –n i> parameters.txt
2. Test the performance on development set for "i" iterations.
   icsiboost –S Example1 –C < Example1.dev> results.txt
3. Record current test results into a log file.

End For

Finding optimum number of iteration.

4. Find the iteration number where F-measure is maximized. (Let it happens at $M \leq N$ iteration.)
5. Find the iteration number where Nist error rate is minimized. (Let it happens at $K \leq N$ iteration.)

Performance evaluation by using optimum number of iteration.

6. *icsiboost –S Example1 –n M > parameters.txt*
7. *icsiboost –S Example1 –C < Example1.test> results_for_fmeasure_max.txt*
8. *icsiboost –S Example1 –n K > parameters.txt*
9. *icsiboost –S Example1 –C < Example1.test> results_for_nist_min.txt*

Table 7.10 The procedure of finding maximum performance for given feature set.

Chapter 8

Conclusion and Test Results

There are several tests performed on sentence boundary detection ability of extracted prosodic features. The F-measure score and Nist error rate evaluation metrics, where they have been defined at Chapter 7, has been used in order to measure the performance of trained model at icsiboost. Chapter 7 also includes the procedure of model training and testing. In this chapter, at first the overview of the method and second the performance evaluations of single-speaker based tests and multi-speaker based tests will be shown.

8.1 Data Sets and Overview of the Used Method

In this work, I have used Voice of America (VOA) Turkish broadcast news [43]records and segment time marks (STM) which includes the transcription of spoken words in defined time segments, information about the speaker such as gender, native non-native and speaker-id, which are recorded and prepared by BUSIM speech group [44] in Boğaziçi University. In addition the STM files also contain all punctuation signs. Those STM files are used as reference files. Each broadcast new program is 30 minutes long, and 16 kHz, 16 bit PCM sampled in audio (wav) format. Those records are also contains the speech of several speakers.

In whole procedure we have worked on UNIX operation system. Because of the Mary Harper's Prosodic Feature Extraction Tool operates in speaker based, in the first step I have re-organized the audio files and corresponding time-segment marks. The main procedure was to detect the selected speaker in each segment-time mark files and concatenate them without impair the time labels, and also performing same procedure for the corresponding audio files. In re-organizing segment-time mark files, several Perl scripts are used and in re-organizing audio files, Sox tool has been used.

In the second step, the HCopy tool of HTK used in order to extract Mel-Frequency Cepstral Coefficients and Energy of short-time segments. Then by using MFCC vectors, speaker-based re-organized segment-time marks and the Turkish Language Model developed by the BUSIM speech group, the word and phoneme based CTM files are obtained by using HVite tool of HTK.

The size of whole collected data is shown at Table 8.1.

Speaker ID	Gender	Number of Words
VOA Speaker 1 (Main Speaker)	Male	13381
VOA Speaker 2 (Main Speaker)	Female	17831
VOA Speaker 3	Male	3547
VOA Speaker 4	Female	2296
VOA Speaker 5	Male	1211
VOA Speaker 6	Male	1840

Table 8.1 The whole data set, the size of the whole data set is 40106 words.

Afterwards, the prosodic feature extraction performed by using Mary Harper's Prosodic Feature Extraction Tool, and finally the sentence boundaries are labeled in order to obtain the references of sentence boundaries.

In the last step, the data is divided into three sets such as training set, development set and test set, according to the task for each test a model is trained by using *icsiboost* and training set (see Chapter 7 for whole procedure) and performance evaluation is performed in both development and test sets in corresponding to either speaker based and non-speaker based performance evaluation tasks. In single speaker performance evaluations, the data collected from main speakers are used and in multi-speaker tasks whole data set has been used. For the each performance evaluation tables below, the used subset of the whole data will be explained.

8.2 Single-Speaker Based Tests

The single-speaker based tests are performed the data corresponding to main speakers who are the first two speakers in Table 8.1. In those tests I have used several classes of features such as all continuous features, formant frequency and formant frequency derived features, and energy and energy derived features and also a feature set which will be denotes as *Model 1*.

In the first case, we have trained models by using all of the features extracted by using *Praat* corresponding the VOA Speaker 1 and VOA Speaker 2 respectively. Table 8.2 shows the test results of models trained by using approximately 250 to 2K words with increment of 250 words, in training set, 4271 words on development set and 6408 words on test set.

8.2.1 All of the continuous variable features on VOA speaker 1.

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
259Words	1	0.708180708180	0.804713804713		
	7	0.777931034482	0.542087542087	0.8004158004	0.460431654
501Words	1	0.671280276816	0.959595959595		
	44	0.915422885579	0.171717171717	0.892485549132	0.22302158
751Words	1	0.671280276816	0.95959595959		
	52	0.913738019169	0.181818181818	0.896	0.218225419
999Words	1	0.671280276816	0.95959595959		
	17	0.892744479495	0.228956228956	0.907621247113	0.191846522
1254Words	1	0.604938271604	1,292929292929		
	773	0.912396694214	0.178451178451	0.880630630630	0.254196642
1492Words	1	0.604938271604938	1.29292929292929		
	136	0.90625	0.181818181818182	0.89281507656	0.218225419
1749Words	1	0.604938271604938	1.29292929292929		
	232	0.928925619834711	0.144781144781145	0.916568742655	0.170263788
2003Words	1	0.580690627202255	1,4268585131894		
	195	0.937293729372937	0.127946127946128	0.900692840646651	0.206235011

Table 8.2 Performance measurements on all continuous features set. The speaker is VOA Speaker 1.

This feature set includes 242 features which are duration, energy, F0 (fundamental frequency) basilar and derived features. In addition it includes rhyme features. When we look at the results in general, we see acceptable F-measure scores and Nist error rates. The best maximum performance is achieved by using all continuous features both for VOA Speaker 1 and VOA Speaker 2 (See Table 8.9). However, using 242 features in both training and testing stages increases the complexity of the system, where recall that all of the features are extracted on each word boundary and lots of weak learners are trained in order to find the best strong learner.

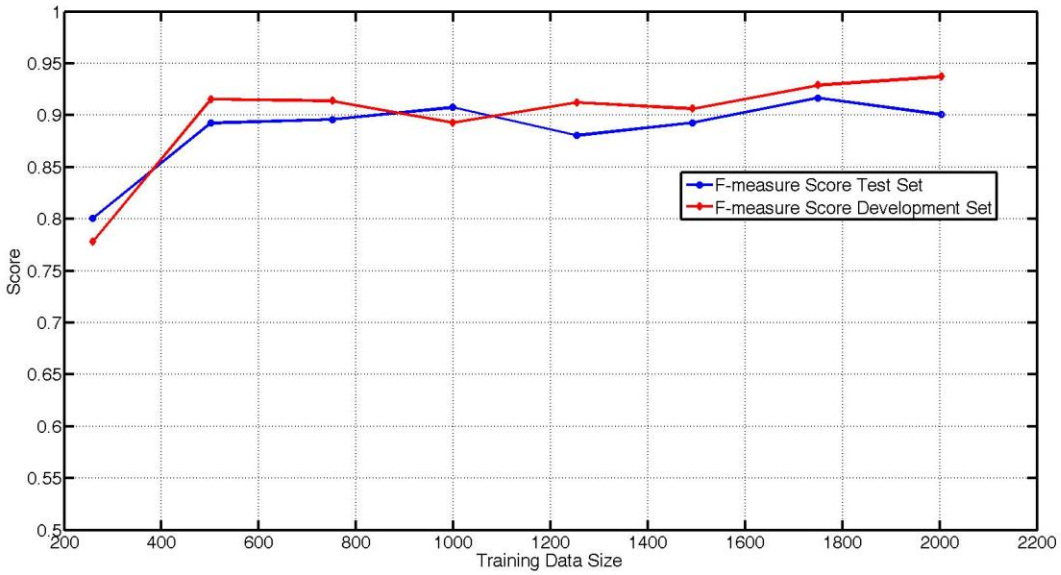


Figure 8.1 F-measure scores of all continuous feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

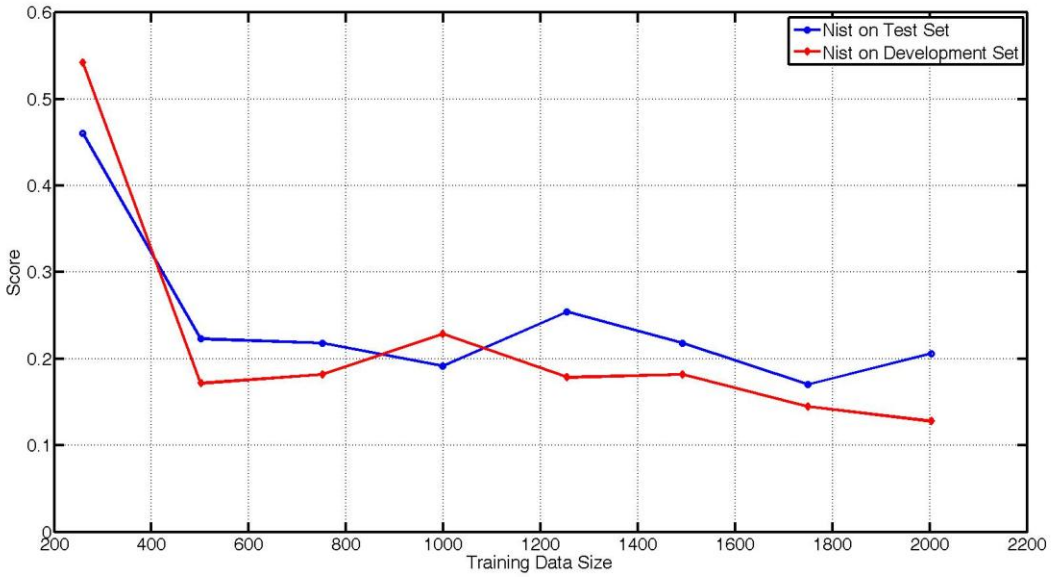


Figure 8.2 Nist error rates of all continuous feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.2 The feature set used in *Model 1*

PAUSE_DUR: continuous.	ENERGY_WORD_DIFF_LOLO_N: continuous.
PATTERN_BOUNDARY: X, ff, fr, rf, rr.	F0K_WORD_DIFF_LOHI_N: continuous.
ENERGY_PATTERN_BOUNDARY: X, ff, fr, rf, rr.	ENERGY_WORD_DIFF_LOHI_N: continuous.
SLOPE_DIFF: continuous.	F0K_WIN_DIFF_HIHI_N: continuous.
ENERGY_SLOPE_DIFF: continuous.	ENERGY_WIN_DIFF_HIHI_N: continuous.
LAST_SLOPE: continuous.	F0K_WIN_DIFF_HILO_N: continuous.
ENERGY_LAST_SLOPE: continuous.	ENERGY_WIN_DIFF_HILO_N: continuous.
LAST_SLOPE_N: continuous.	F0K_WIN_DIFF_LOLO_N: continuous.
ENERGY_LAST_SLOPE_N: continuous.	ENERGY_WIN_DIFF_LOLO_N: continuous.
F0K_WORD_DIFF_HIHI_N: continuous.	F0K_WIN_DIFF_LOHI_N: continuous.
ENERGY_WORD_DIFF_HIHI_N: continuous.	ENERGY_WIN_DIFF_LOHI_N: continuous.
F0K_WORD_DIFF_HILO_N: continuous.	F0K_WORD_DIFF_MNMN_N: continuous.
ENERGY_WORD_DIFF_HILO_N: continuous.	ENERGY_WORD_DIFF_MNMN_N: continuous.
F0K_WORD_DIFF_LOLO_N: continuous.	F0K_WORD_DIFF_BEGBEG: continuous.
F0K_WORD_DIFF_ENDBEG: continuous.	ENERGY_WORD_DIFF_BEGBEG: continuous.
ENERGY_INWORD_DIFF: continuous.	ENERGY_WORD_DIFF_ENDBEG: continuous.
	F0K_INWORD_DIFF: continuous.

Table 8.3 List of the used features in Model 1.

Model 1 [38] features includes several information such as pause duration between the word preceding a boundary and after that boundary, pattern slope and pattern slope energy values, log differences of minimum, maximum, and mean values of stylized F0 and energy features between adjacent words, frames and word extremes [38].

Instead of using all of 242 continuous valued features in that set only 33 features are used. Those feature set includes the most specific information to detect the sentence boundaries. When the results with all continuous feature set are compared, there is a little decrease in the performance of the maximized model, however in general the performance is increased. Using too much features may add redundant information and may cause reduced performance in different training sets.

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
259Words	1	0.6925880923	0.8518518518518		
	11	0.9154228857	0.1919191919191	0.8473372781065	0.3093525179856
501Words	1	0.6478632478	0.9880095923261		
	7	0.8756218905	0.2525252525252	0.8342989571263	0.3429256594724
751Words	1	0.686346231	0.900764354345		
	94	0.9047619047	0.1885521885521	0.856097560975	0.2829736211031
999Words	1	0.7170294494	0.7441077441077		
	184	0.9319727891	0.1346801346801	0.8805790108564	0.2374100719424
	1149	0.9326599326	0.1346801346801	0.8841099163679	0.2326139088729
1254Words	1	0.6322580645	1,151515151515		
	106	0.9169435215	0.1683501683501	0.8723897911832	0.2637889688249
1492Words	1	0.6322580645	1,151515151515		
	1024	0.9259896729	0.144781144781	0.867924528301	0.2685851318944
1749Words	1	0.6322580645	1,151515151515		
	235	0.9305555555	0.1346801346801	0.8857142857142	0.2302158273381
	467	0.9310344827	0.1346801346801	0.8757396449704	0.2517985611510
2003Words	1	0.2517985611	1.1515151515151		
	1708	0.9331046312	0.1313131313131	0.8752997601918	0.2494004796163

Table 8.4 Performance measurements on the Model 1 features set. The speaker is VOA Speaker 1.

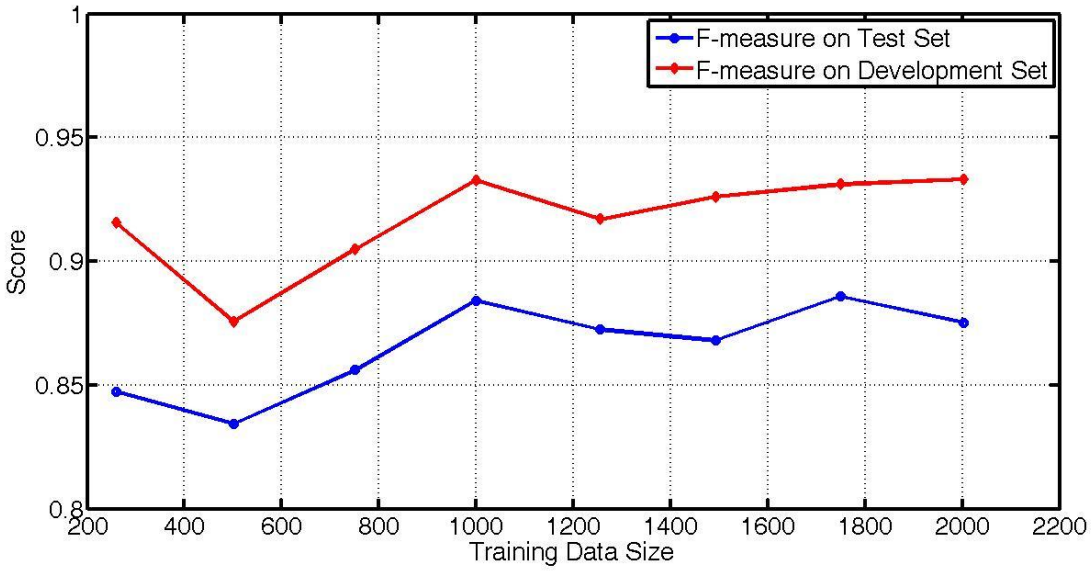


Figure 8.3 F-measure scores of *model 1* feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

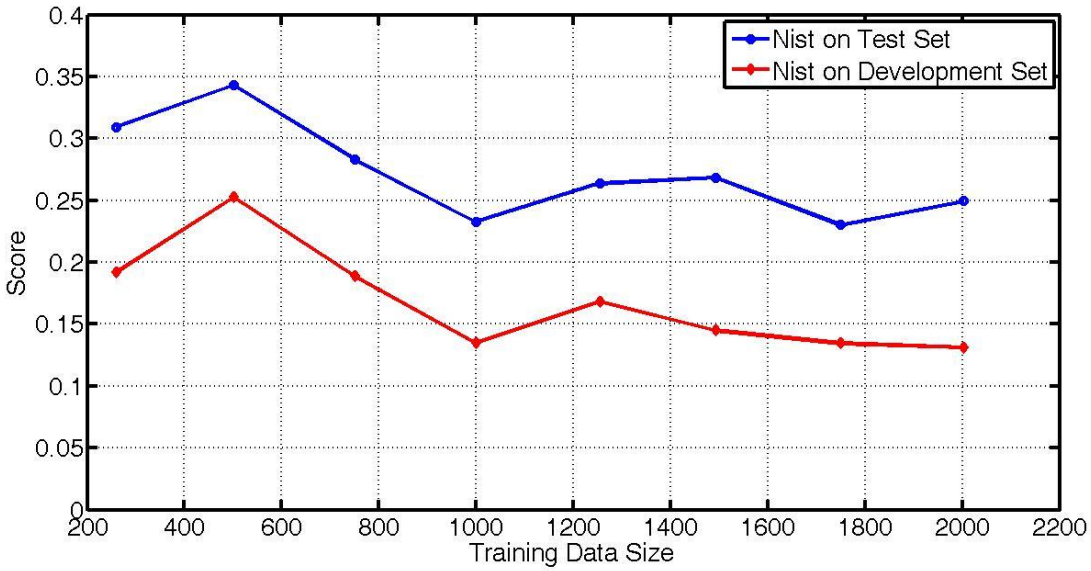


Figure 8.4 Nist error rate of *model 1* feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.3 The tests performed by using F0 derived features on VOA Speaker 1.

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
259Words	1	0.4270749395	2		
	15	0.7761194029	0.40404040404040	0.727509778357	0.5011990407673
501Words	1	0.7538200339	0.48821548821548		
	1549	0.7540983606	0.45454545454545	0.7462686567164	0.4892086330935
751Words	1	0.7774193548	0.46464646464646		
	12	0.796019900	0.41414141414141	0.760693641618	0.496402877697
999Words	1	0.777419354	0.46464646464646		
	596	0.7789473684	0.42424242424242	0.7404580152671	0.4579124579124
1254Words	1	0.7774193548	0.46464646464646	0.7119386637458	0.6306954436450
	4	0.7731397459	0.42087542087542	0.7717528373266	0.4340527577937
1492Words	1	0.7774193548	0.46464646464646	0.7119386637458	0.6306954436450
	2	0.7593582887	0.45454545454545	0.7717528373266	0.4340527577937
1749Words	1	0.7774193548	0.46464646464646	0.7119386637458	0.630695443645
	2	0.7593582887	0.45454545454545	0.7543424317617	0.4748201438848
2003Words	1	0.7774193544	0.46464646464646		
	8	0.7810858143	0.42087542087542	0.7701564380264	0.4580335731414

Table 8.5 Performance measurements on the F0 derived features set for VOA Speaker 1.

This feature set includes only the derived fundamental frequency features in order to observe the role of using F0 derived features. We got reduced performance but still acceptable results. In the next stage we have also added the basilar F0 features. We have observed a little increase when we have added also basilar F0 features. So those results gave us the idea that, while making a combined feature set, several F0 features should be included.

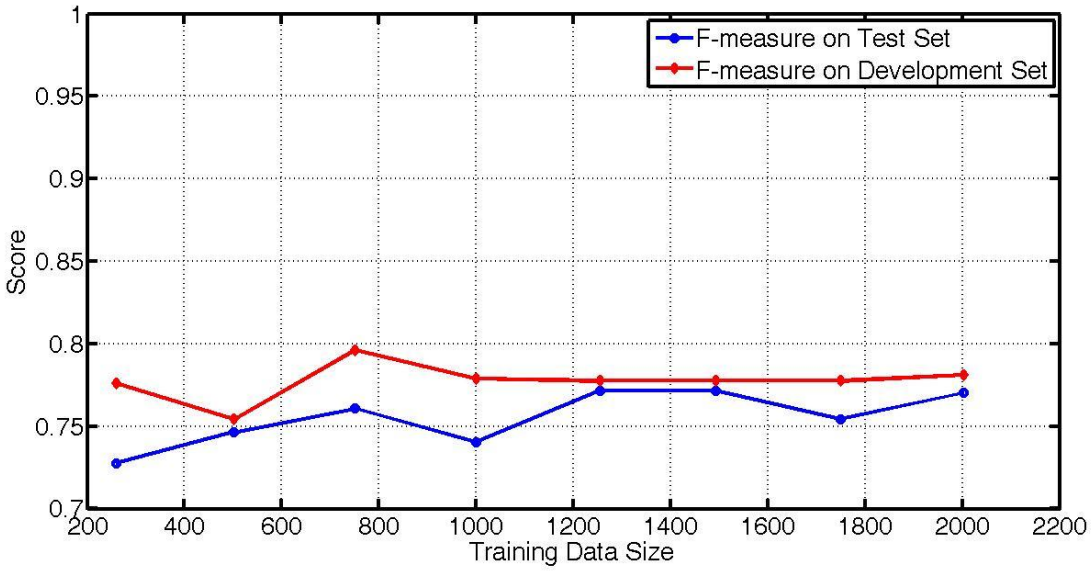


Figure 8.5 F-measure score of F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

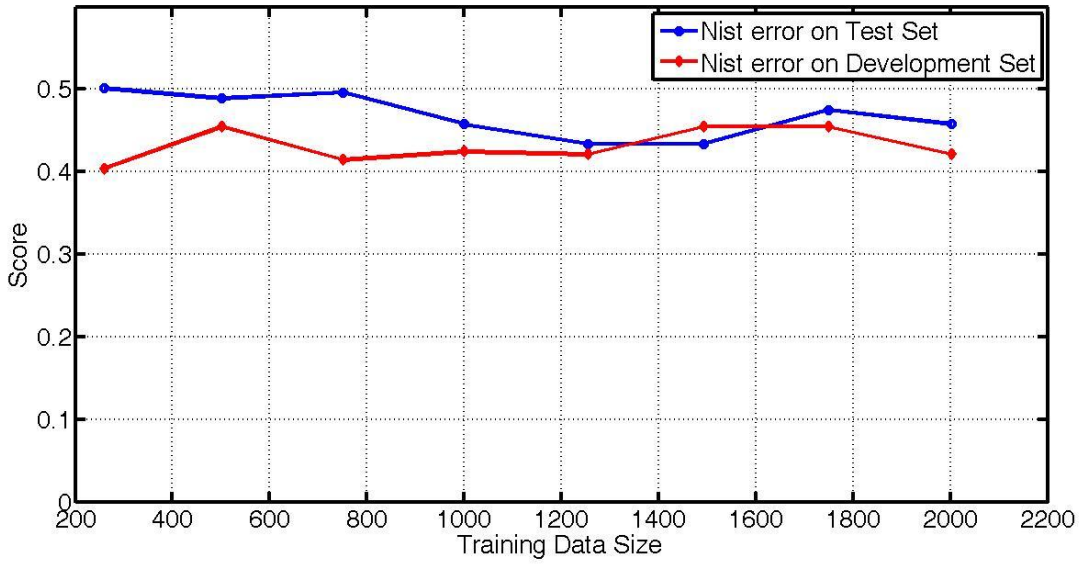


Figure 8.6 Nist error rate of F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.4 Using both formant frequency and formant frequency derived features on VOA_speaker 1.

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
259 Words	1	0.4270749395648	0.23939393939393		
	14	0.7705479452054	0.45117845117845	0.7182587666263	0.71825876662
	17	0.7673179396092	0.44107744107744	0.6889714993804	0.60191846522
501 Words	1	0.7538200339558	0.48821548821548		
	2	0.7686832740213	0.43771043771043	0.7528809218950	0.46282973621
751Words	1	0.7774193544838	0.46464646464646		
	2	0.7810858143607	0.42087542087542	0.77015643802647	0.45803357314
999Words	1	0.7774193548387	0.46464646464646		
	2	0.7925801011804	0.41414141414141	0.75630252100840	0.48681055155
1254Words	1	0.7774193548387	0.46464646464646		
	4	0.8027444253859	0.38720538720538	0.78303030303030	0.42925659472
1492Words	1	0.7774193548387	0.46464646464646		
	2	0.7918781725888	0.41414141414141	0.75030156815440	0.49640287769
1749Words	1	0.7774193548387	0.46464646464646		
	2	0.7918781725888	0.41414141414141	0.75030156815440	0.49640287769
2003Words	1	0.7774193548387	0.46464646464646		
	6	0.7959183673469	0.404040404040404	0.760765550239234	0.479616306954

Table 8.6 Performance measurements on f0 and f0 derived features. The speaker is VOA Speaker 1.

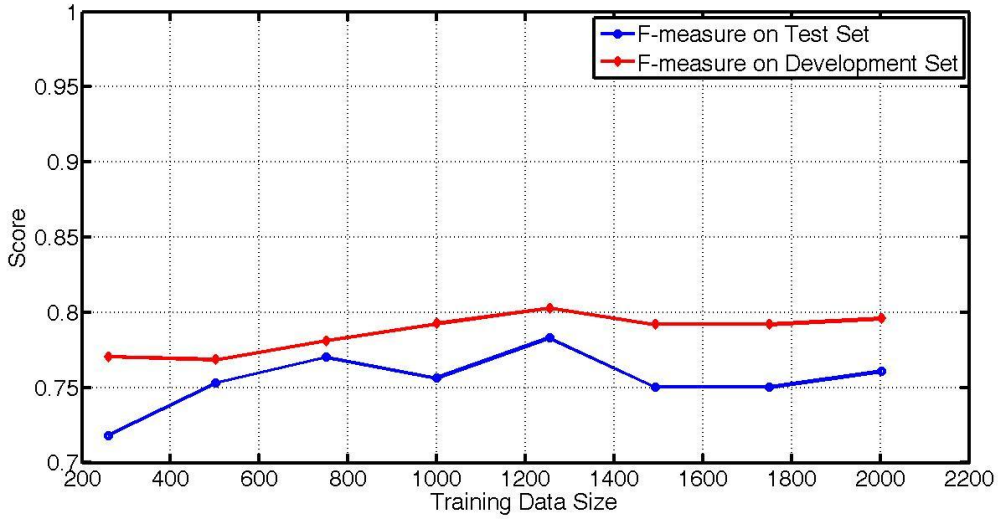


Figure 8.7 F-measure score of F0 and F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

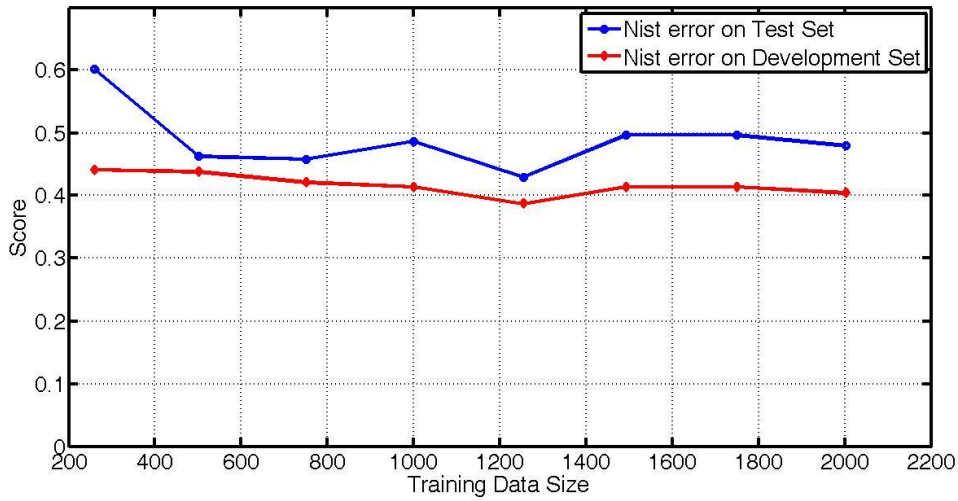


Figure 8.8 Nist error rate of F0 and F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.5 Using energy and energy derived features on VOA speaker 1.

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
259 Words	1	0	1		
	6	0.4598698481	0.83838383838383	0.54696132596685	0.79036144578313
	673	0.5618915159	1.06060606060606	0.57468354430379	0.80575539568345
501 Words	1	0	1		
	58	0.5245202558	0.75084175084175	0.59595959595959	0.76738609112709
	2900	0.6187290969	0.76738609112709	0.58751529987760	0.80815347721822
751Words	1	0	1		
	2191	0.6159420289	0.71380471380471	0.62913096695226	0.72661870503597
	3114	0.6136783733	0.70370370370370	0.63093788063337	0.72661870503597
999Words	1	0	1		
	477	0.6747404844	0.63299663299663	0.63940520446096	0.69784172661870
	491	0.6654676258	0.62626262626262	0.64267990074441	0.69064748201438
1254Words	1	0	1		
	153	0.6309751434	0.64983164983165	0.63080684596577	0.72422062350119
	881	0.6395759717	0.68686868686868	0.60838323353293	0.78417266187050
1492Words	1	0	1		
	128	0.6264591439	0.64646464646464	0.64233576642335	0.70503597122302
1749Words	1	0	1		
	6381	0.5350877192	0.71380471380471	0.61887694145758	0.76498800959232
2003Words	1	0	1		
	2371	0.5267857142	0.713804713804714	0.636150234741784	0.743405275779377

Table 8.7 Performance measurements on energy and energy derived features. The speaker is VOA Speaker 1.

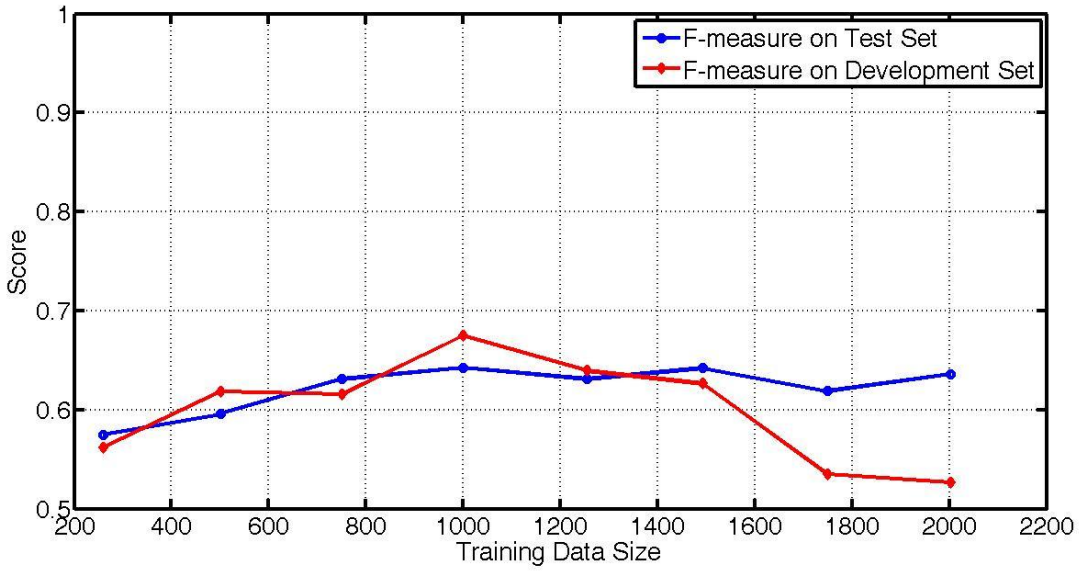


Figure 8.9 F-measure score of energy and energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

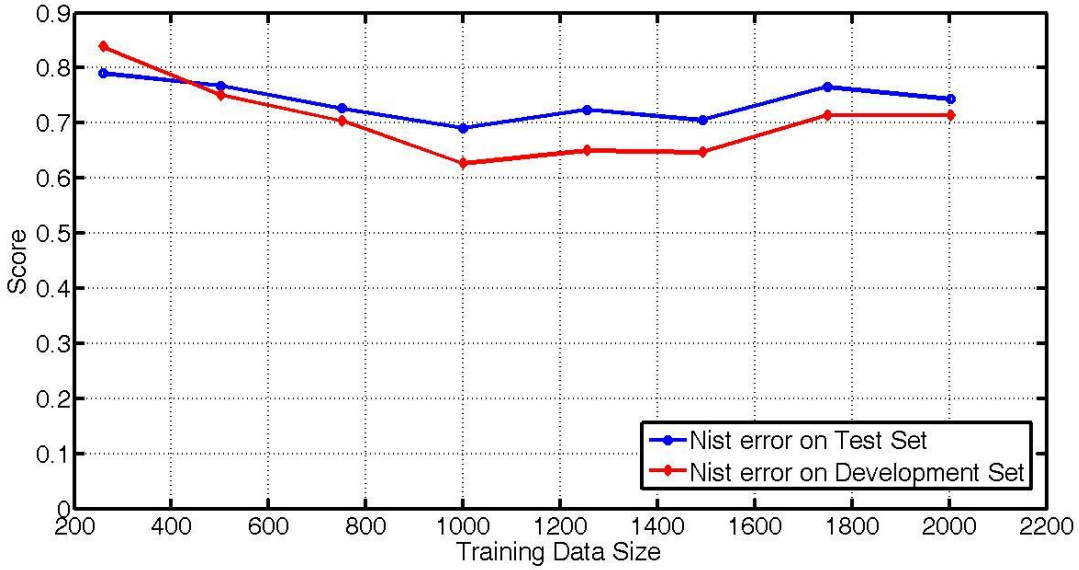


Figure 8.10 Nist error rate of energy and energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.6 Using energy derived features on VOA speaker 1.

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
259 Words	1	0	1		
	2	0.472727272	0.78378378378378	0.51923076923076	0.72289156626506
501 Words	1	0	1		
	81	0.490566037	0.81818181818181	0.58259773013871	0.793764988009592
751Words	1	0	1		
	233	0.590998043	0.70370370370370	0.62195121951219	0.743405275779377
999Words	1	0	1		
	56	0.652908067	0.62289562289562	0.62295081967213	0.717026378896882
1254Words	1	0	1		
	923	0.617424242	0.68013468013468	0.61975308641975	0.738609112709832
1492Words	1	0	1		
	220	0.564102564	0.68686868686868	0.64259927797833	0.712230215827338
1749Words	1	0	1		
	46	0.5517241379	0.7003367003367	0.61090909090909	0.76978417266187
2003Words	1	0	1		
	8	0.3864229765	0.79124579124579	0.57142857142857	0.949640287769784

Table 8.8 Performance measurements on energy derived features. The speaker is VOA Speaker 1.

When we use only energy features to train a model, we get unacceptable results. However we should not conclude that the energy features are useless. The combination of energy and F0 features provides high performance as we have observed at Model 1 feature set.

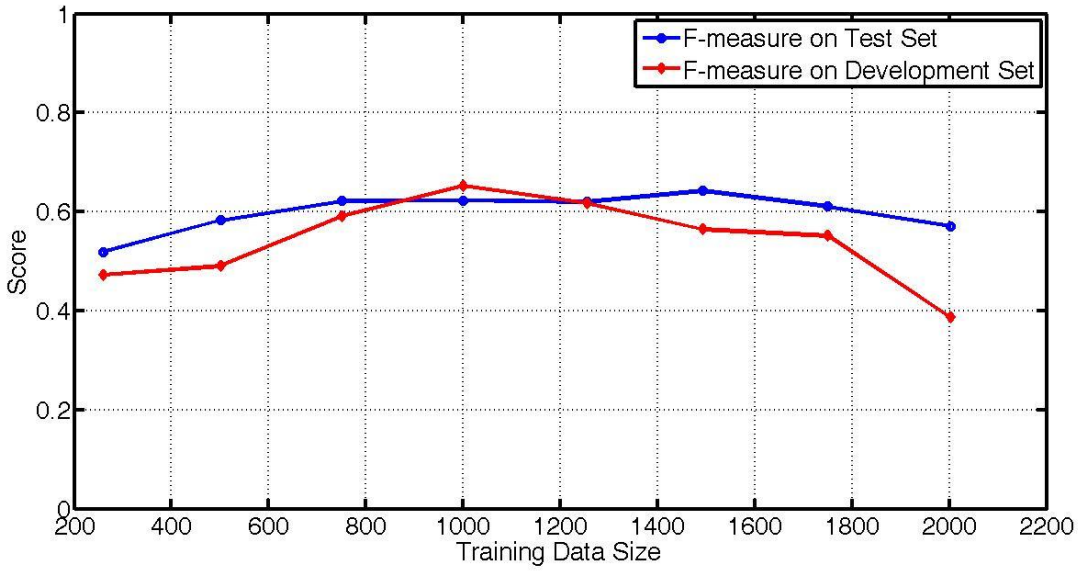


Figure 8.11 F-measure score of energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

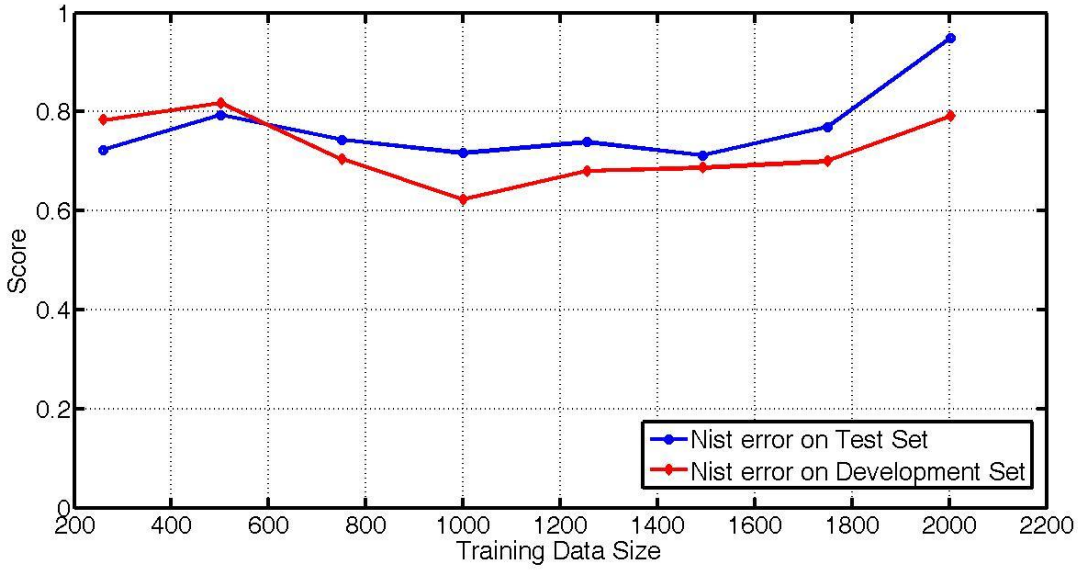


Figure 8.12 Nist error rate of energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 1.

8.2.7 All of the continuous variable features on VOA speaker 2

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
249 Words	1 5	0 0.860759493670	1 0.268839103869	0.853994490358	0.28042328042328
503 Words	1 256	0 0.922246220302	1 0.146639511201	0.904135338345	0.17989417989418
749Words	1 40	0 0.930526315789	1 0.134419551934	0.906787330316	0.18165784832451
1009Words	1 561	0 0.92713833157	1 0.140529531568	0.911737943585	0.171075837742
1246Words	1 1609	0 0.93736951983	1 0.122199592668	0.91109074243813	0.1710758377425
1508Words	1 2505	0 0.93782929399	1 0.1201629327902	0.9085872576177	0.1746031746031
1748Words	1 208	0 0.93993677555	1 0.1160896130346	0.9288928892889	0.1393298059964
1997Words	1 994	0 0.94363256784	1 0.1099796334012	0.929856115107	0.137566137566
2253Words	1 139	0 0.9468196037	1 0.103869653767	0.934163701067	0.13051146384475
2520Words	1 704	0 0.9446185997	1 0.10794297352	0.92735426008	0.14285714285567
2707Words	1 212	0 0.94375	1 0.109979633	0.932263814	0.13403880075646

Table 8.9 Performance measurements on all continuous features. The speaker is VOA Speaker 2.

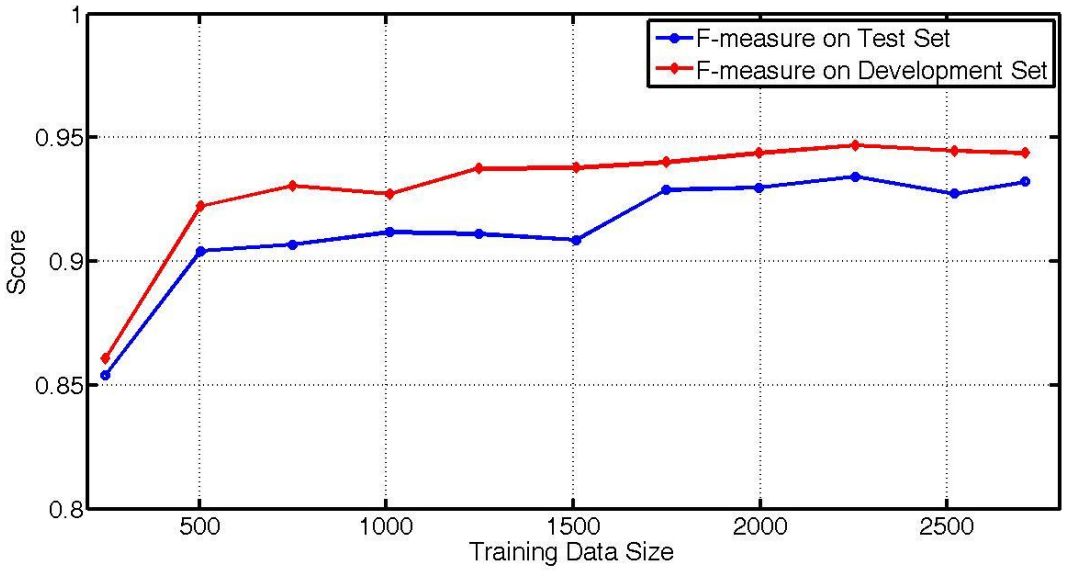


Figure 8.13 F-measure score of all continuous feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

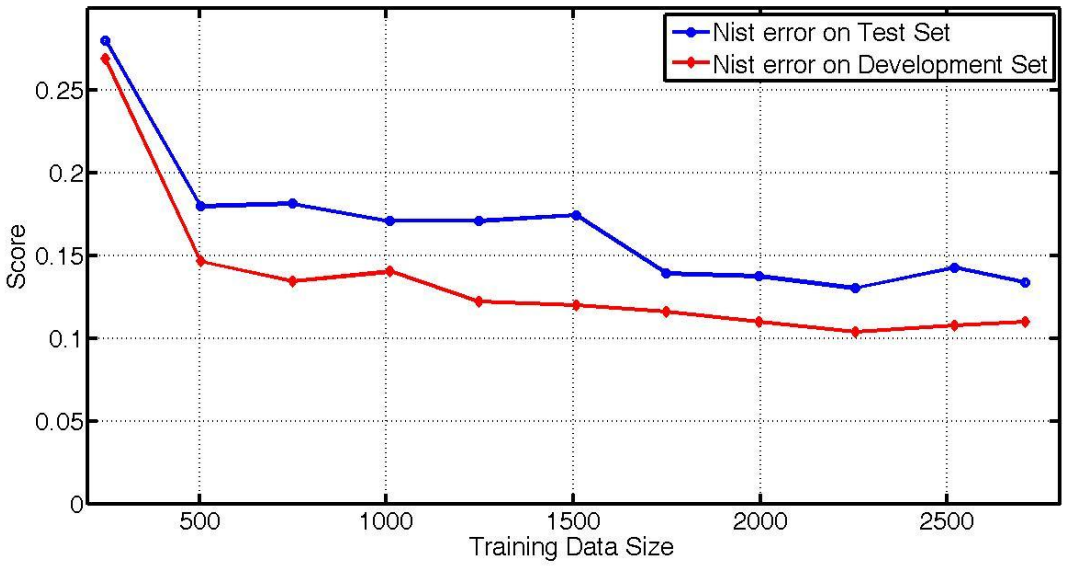


Figure 8.14 Nist error rate of all continuous feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

8.2.8 Model 1 features on VOA speaker 2

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
249 Words	1	0	1		
	30	0.875776397515	0.244399185336	0.882196634189	0.234567901234
503 Words	1	0	1		
	1394	0.907407407407	0.183299389002	0.905357142857	0.18694885361552
749Words	1	0	1		
	33	0.906414300736	0.18126272912	0.905357142857	0.18694885361552
1009Words	1	0	1		
	58	0.91806722689	0.158859470468	0.91992882562	0.158730158730
1246Words	1	0	1		
	123	0.92387904066	0.14867617107943	0.91829484902	0.16225749559
1508Words	1	0	1		
	37	0.91869060190	0.15682281059	0.91823899371	0.160493827160
1748Words	1	0	1		
	147	0.9196617336	0.154786150712	0.916890080428	0.164021164021
1997Words	1	0	1		
	1767	0.9186906019	0.156822810590	0.907161803713	0.1851851851851
2253Words	1	0	1		
	74	0.92323869610	0.14867617107943	0.915467625899	0.165784832451
2520Words	1	0	1		
	37	0.9175475687	0.15885947046	0.918918918918	0.15873015873
2707Words	1	0	1		
	51	0.92096944151	0.15274949083	0.9111900532859	0.17636684303351

Table 8.10 Performance measurements on Model 1 features. The speaker is VOA Speaker 2.

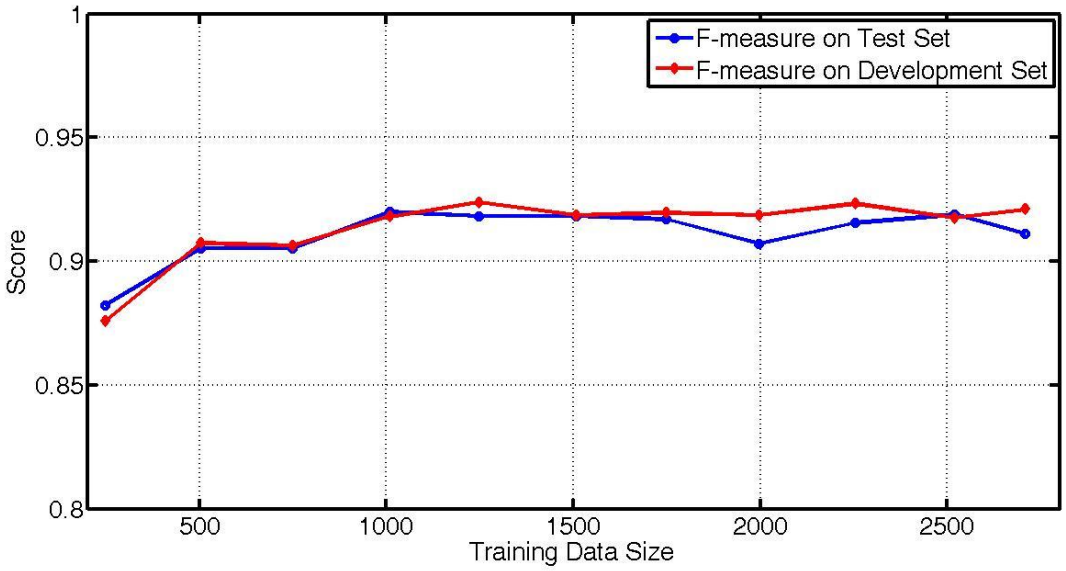


Figure 8.15 F-measure score of Model 1 feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

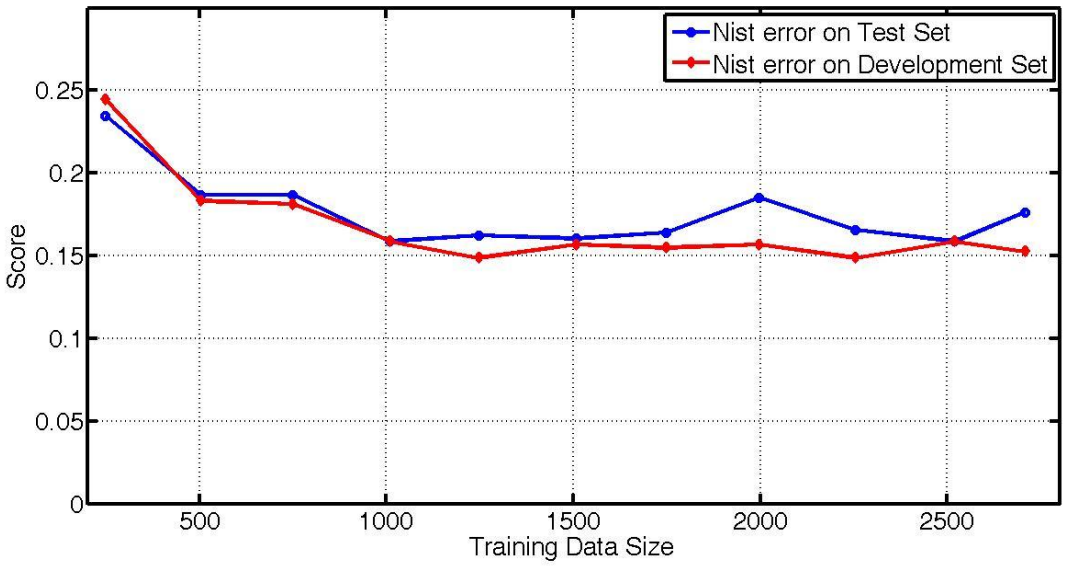


Figure 8.16 Nist error of Model 1 feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

8.2.9 F0 derived features on VOA speaker 2

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
249 Words	1 308	0 0.7570754716	1 0.419551934826	0.723529411764	0.497354497354
503 Words	1 619	0 0.789053591	1 0.3767820773	0.74157303370	0.486772486772
749Words	1 712	0 0.786697247	1 0.378818737270	0.74102079395	0.48324514991
1009Words	1 58	0 0.7871116225	1 0.37678207739	0.75023386342376	0.470899470899
1246Words	1 55	0 0.793721973	1 0.374745417515	0.767730496453	0.462081128747
1508Words	1 43	0 0.814238042	1 0.340122199592	0.7603305785123	0.46031746031746
1748Words	1 34	0 0.806167400	1 0.358452184928	0.7594254937163	0.472663139329
1997Words	1 14	0 0.8247863247	1 0.334012219959	0.75996457041	0.477954144620
2253Words	1 39	0 0.8048511576	1 0.360488798370	0.74270072992	0.47878365832
2520Words	1 88	0 0.821585903	1 0.32993890020	0.769230769230	0.460317460317
2707Words	1 87	0 0.826985854	1 0.32382892057	0.78141135972	0.447971781305

Table 8.11 Performance measurements on F0 derived features. The speaker is VOA Speaker 2.

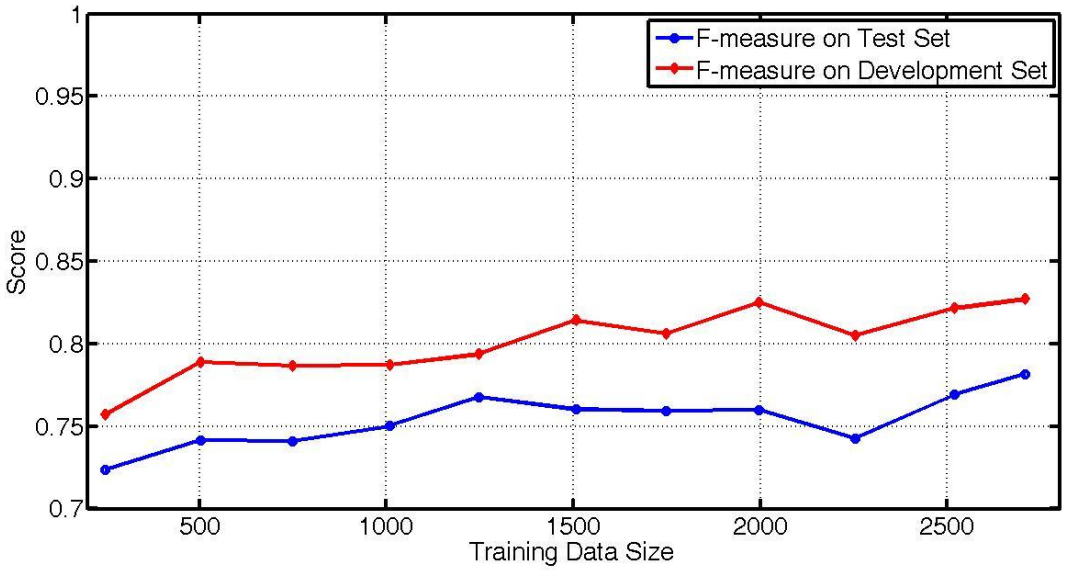


Figure 8.17 F-measure score of F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

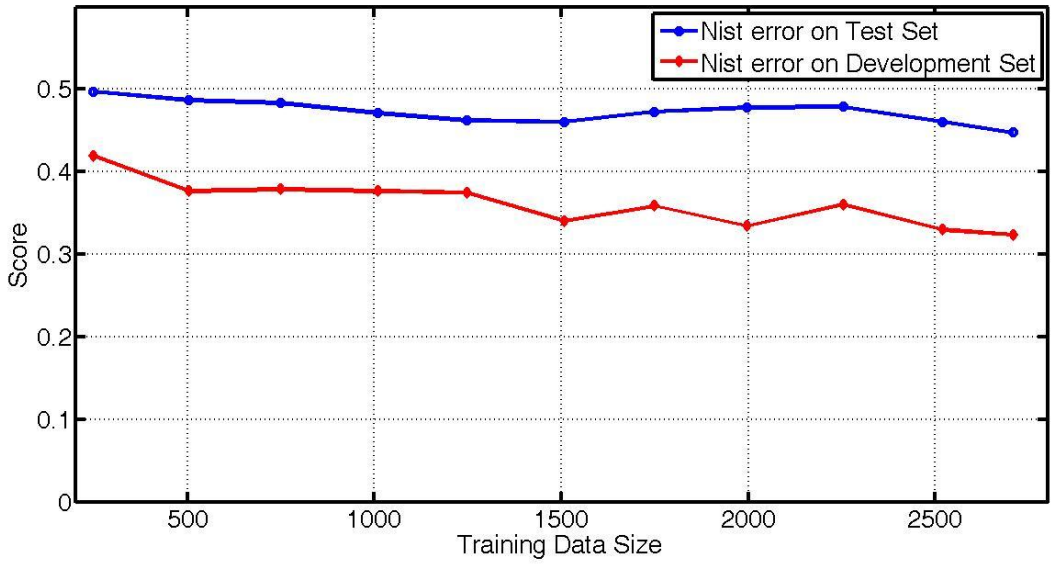


Figure 8.18 Nist error rate of F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

8.2.10 F0 and F0 derived features on VOA speaker 2

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
249 Words	1 8	0 0.7323290845	1 0.470468431771	0.686807653575	0.548500881834
503 Words	1 144	0 0.782022471	1 0.395112016293	0.743127962085	0.477954144620
749Words	1 2068	0 0.768166089	1 0.409368635437	0.722772277227	0.493827160493
1009Words	1 239	0 0.7968217934	1 0.364562118126	0.755725190839	0.451499118165
1246Words	1 669	0 0.8120133481	1 0.344195519348	0.7762039660056	0.4179894179894
1508Words	1 43	0 0.7982261640	1 0.3706720977596	0.752380952380	0.4585537918871
1748Words	1 156	0 0.803591470	1 0.356415478615	0.771668219944	0.432098765432
1997Words	1 89	0 0.808189655	1 0.362525458248	0.780487804878	0.412698412698
2253Words	1 232	0 0.8030803080	1 0.364562118126	0.772258669165	0.42857142857
2520Words	1 94	0 0.8217391304	1 0.334012219959	0.767657992565	0.440917107583
2707Words	1 161	0 0.8190682556	1 0.34012219959	0.790235081374	0.409171075837

Table 8.12 Performance measurements on F0 and F0 derived features for VOA Speaker 2.

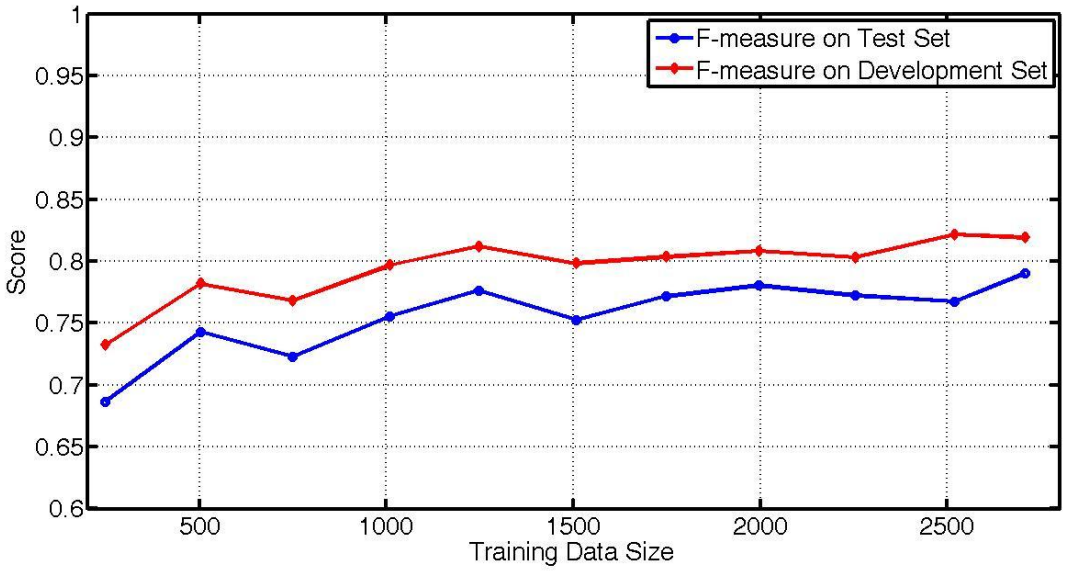


Figure 8.19 F-measure score of F0 and F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

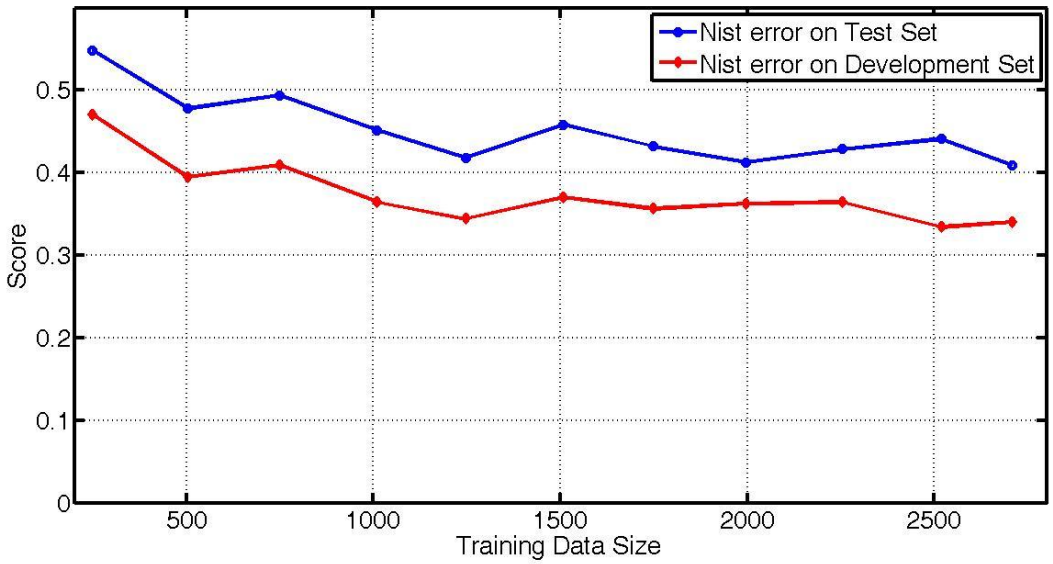


Figure 8.20 Nist error rate of F0 and F0 derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

8.2.11 Energy and energy derived features on VOA speaker 2

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
249 Words	1 1	0 0	1 1	0	1
503 Words	1 13	0 0.24749163879	1 0.916496945010	0.2586206896551	0.91005291005291
749Words	1 43	0 0.39595375722	1 0.85132382892057	0.3915211970074	0.8606701940035
1009Words	1 91	0 0.4846938775	1 0.822810590631	0.471508379888	0.8342151675485
1246Words	1 80	0 0.53531598513	1 0.7637474541751	0.50755939524838	0.8042328042328
1508Words	1 197	0 0.52131546894	1 0.80040733197556	0.486064659777	0.81305114638448
1748Words	1 950	0 0.54373522458	1 0.786150712830	0.514631685166	0.8483245149911
1997Words	1 113	0 0.52088452088	1 0.7942973523421	0.5174973488865	0.8024691358024
2253Words	1 338	0 0.52450980392	1 0.790224032586	0.544761904761	0.8430335097001
2520Words	1 127	0 0.54312354312	1 0.79837067209776	0.5072164948453	0.8430335097001
2707Words	1 162	0 0.55359246171	1 0.771894093686	0.5342601787487	0.8271604938271

Table 8.13 Performance measurements on energy and energy derived features. The speaker is VOA Speaker 2.

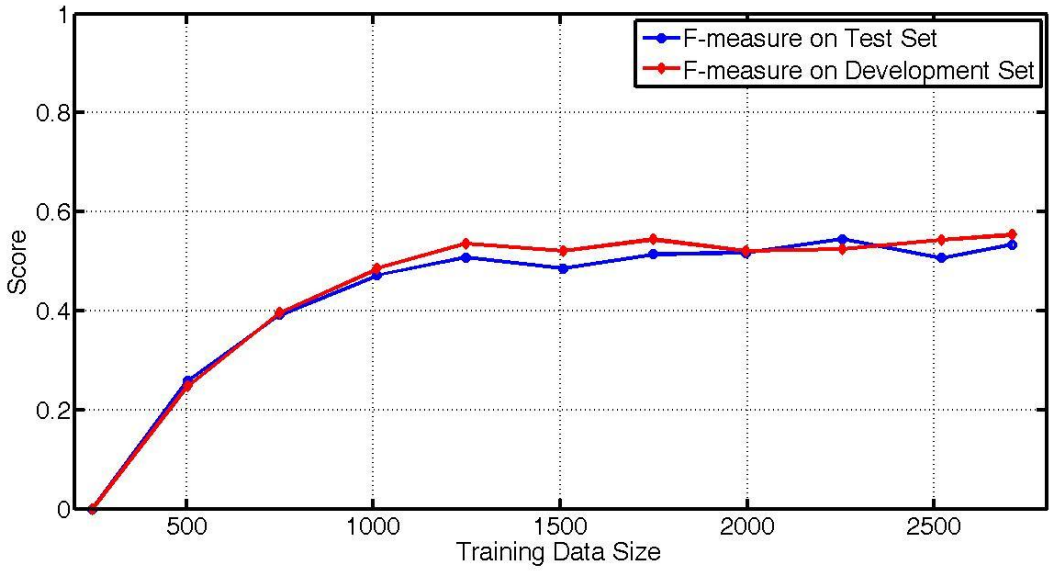


Figure 8.21 F-measure score of energy and energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

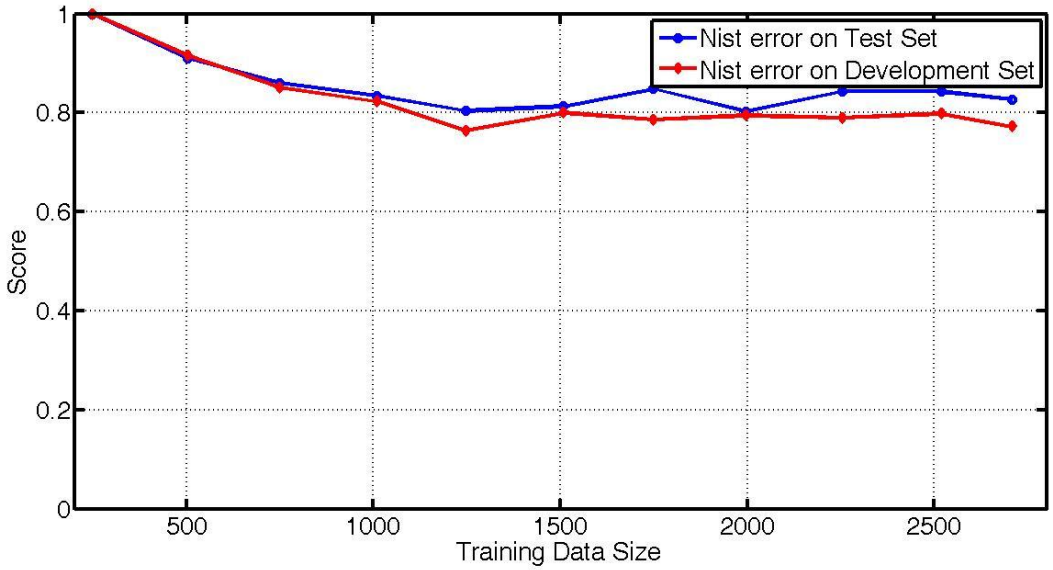


Figure 8.22 Nist error rate of energy and energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

8.2.12 Energy derived features on VOA speaker 2

Training Set Size	Iteration Number	F-measure on Development Set	Nist on Development Set	F-measure on Test Set	Nist on Test Set
249 Words	1	0	1		
	10	0.1839464882	0.9938900203665	0.2113821138211	1.02645502645503
503 Words	1	0	1		
	1	0	1	0	1
749Words	1	0	1		
	5	0.3042836041	0.9592668024439	0.246771879483	0.9259259259259
1009Words	1	0	1		
	94	0.3656207366	0.947046843177	0.455958549222	0.925925925925
1246Words	1	0	1		
	19	0.41551246537	0.8594704684317	0.4425531914893	0.9241622574955
1508Words	2	0	1		
	236	0.4189189189	0.875763747454	0.431289640591	0.9488536155202
1748Words	1	0	1		
	179	0.3977746870	0.881873727087	0.4667349027635	0.9188712522045
1997Words	1	0	1		
	239	0.397183098	0.871690427698	0.438614900314	0.943562610229
2253Words	1	0	1		
	648	0.4137022397	0.906313645621	0.4386129334582	1.05643738977072
2520Words	1	0	1		
	20	0.4183168316	0.95723014256	0.359813084112	0.966490299823
2707Words	1	0	1		
	22	0.3834482758	0.910386965376	0.371929824561	0.947089947089

Table 8.14 Performance measurements on energy derived features. The speaker is VOA Speaker 2.

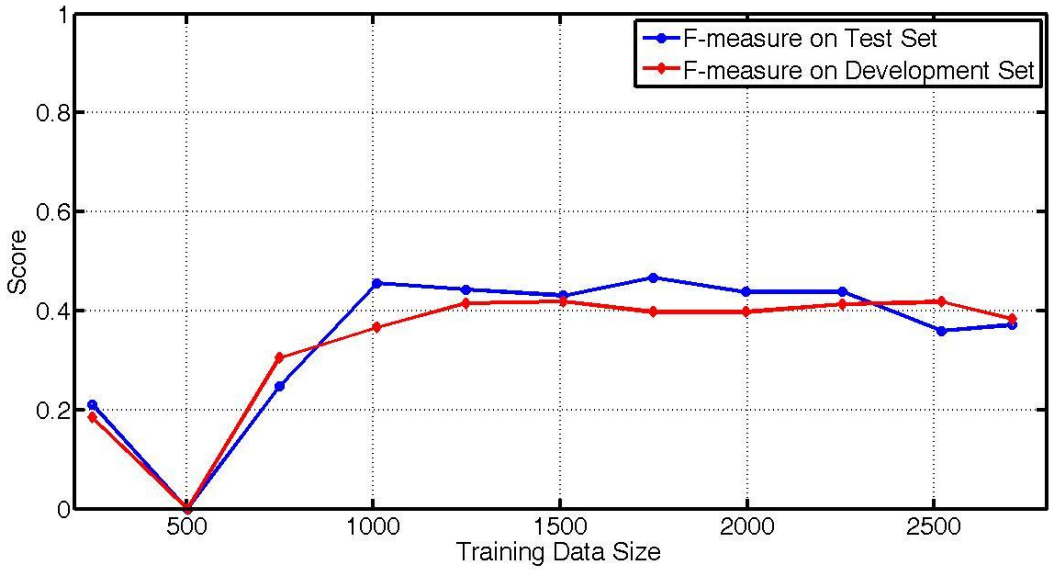


Figure 8.23 F-measure score of energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

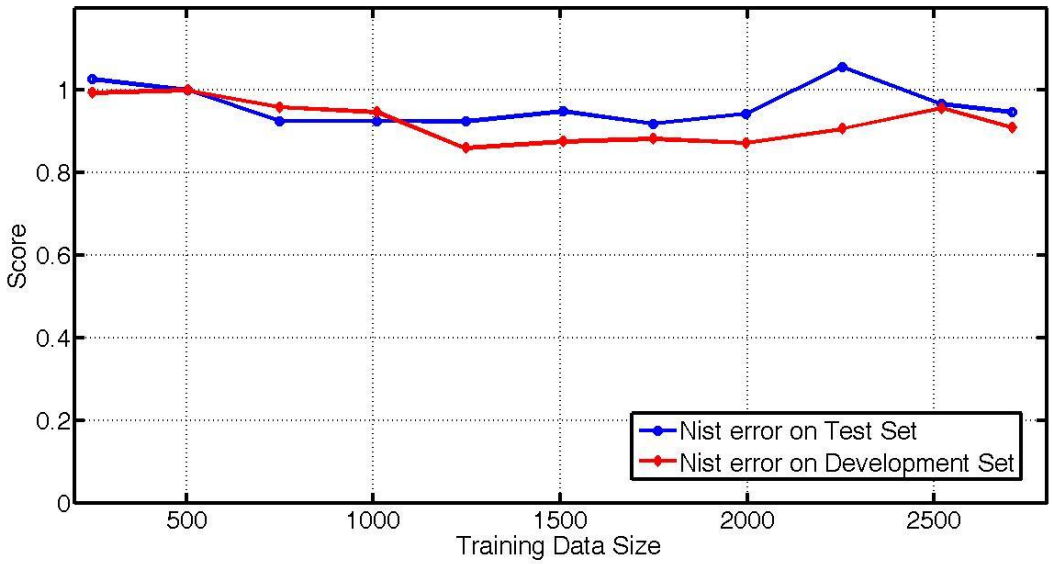


Figure 8.24 Nist error rate of energy derived feature set on development set (red color) and test set (blue color) of VOA Speaker 2.

In the tests above, several feature sets are tested by using both speeches of VOA speaker 1 and VOA speaker 2, such as a feature set contains all continuous prosodic features, model 1 features, which is mostly combination of formant frequency and corresponding (similar) energy features, basilar F0 and F0 derived features together, formant frequencies alone, energy and energy features together and finally energy derived features alone.

Above we have observed that, most succeed results are occurred in the models which are trained by using either all continuous features or the model 1 features. However, in all continuous feature set there are approximately 200 features and in model 1 feature set there are approximately 35 features. As it has mentioned before, N features are extracted and trained for each word. While designing a system, the computational complexity should be also considered. In this sight of view, using model 1 feature set is much more efficient when compared with using all continuous features.

8.3 Multi-Speaker Based Tests

In this step, speaker-independent tests will be performed. Hence, all of the data sets which are train sets, test sets and development sets will include features of more than one person. The tests are divided into two steps. At the first step, a model is trained by using features of VOA speaker 1 and VOA speaker 2. Then the performance of that models are evaluated in development sets and test sets respectively where the development sets and test sets includes only those speakers. In the second step, the trained models in step 1 are used in order to perform sentence segmentation only on features belongs to VOA Speakers 3-4-5-6. Hence the test set has changed. In the new test set, there is no any features belongs to either VOA speaker 1 or 2.

In this test, two models are used. The first one is model 1 which has explained before, and the second one is model 2, where in that model similar with [33], the duration features, F0 and corresponding energy features are used, in addition the gender information is also used.

8.3.1 Tests performed with model 1 feature set

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
508Words	1	0.60632411067	1.26395939086294	0.8873239436619	0.2276422764227
	324	0.90542244640	0.1903553299492	**0.8263959390862**	**0.33011583011583**
1004Words	1	0.60632411067	1.26395939086294	0.8964813870474	0.20630081300813
	1987	0.91923076923	0.15989847715736	**0.82962962962963**	**0.3108108108108**
1500Words	1	0	1	0.8849467815509	0.2306910569105
	1987	0.9070063694	0.185279187817	**0.8121085594989**	**0.3474903474903**
2008Words	1	0	1	0.8967280163599	0.2052845528455
	1987	0.9102730819	0.175126903553	**0.7683741648106**	**0.4015444015444**
2500Words	1	0	1	0.89963973237262	0.19817073170731
	256	0.9216710182	0.152284263959	**0.8139281828073**	**0.33011583011583**
3000Words	1	0	1	0.8881922675026	0.2174796747967
	256	0.9184210526	0.1573604060913	**0.8216216216216**	**0.3185328185328**
3493Words	1	0	1	0.89723526343244	0.20020325203252
	256	0.9098090849	0.173857868020	**0.8083242059145**	**0.3378378378378**
4000Words	1	0	1	0.9034589571502	0.1900406504065
	256	0.9125326370	0.17005076142132	**0.8079034028540**	**0.3378378378378**

Table 8.15 Performance measurements on model 1 set of features. In the test column, bold results are correspond to tests VOA Speakers 3-6.

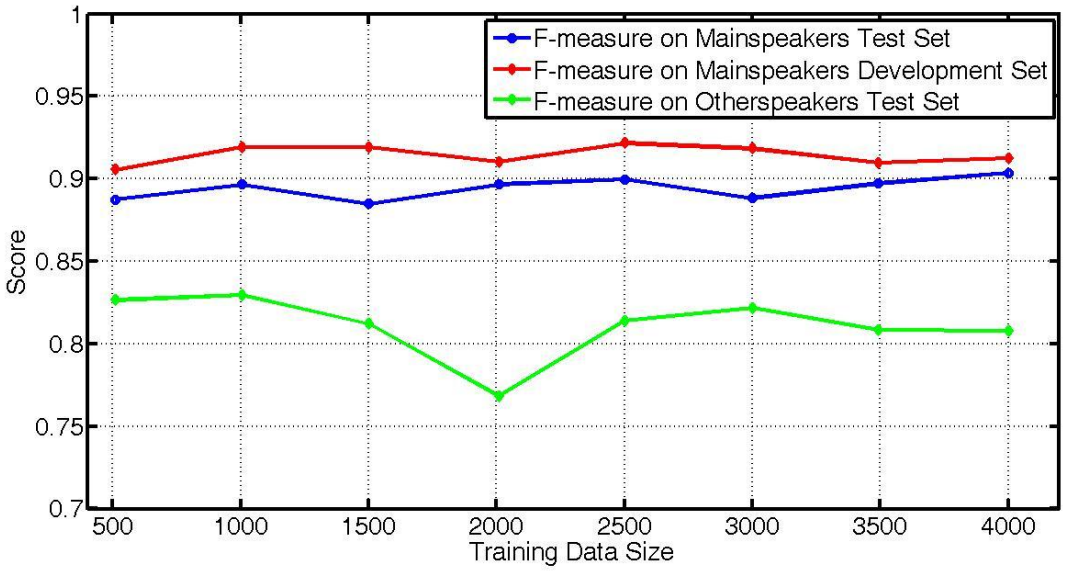


Figure 8.25 F-measure score on model 1 feature set on development set belongs to main speakers (red color), test set belongs to main speakers (blue color), and test set belongs to other speakers (green color).

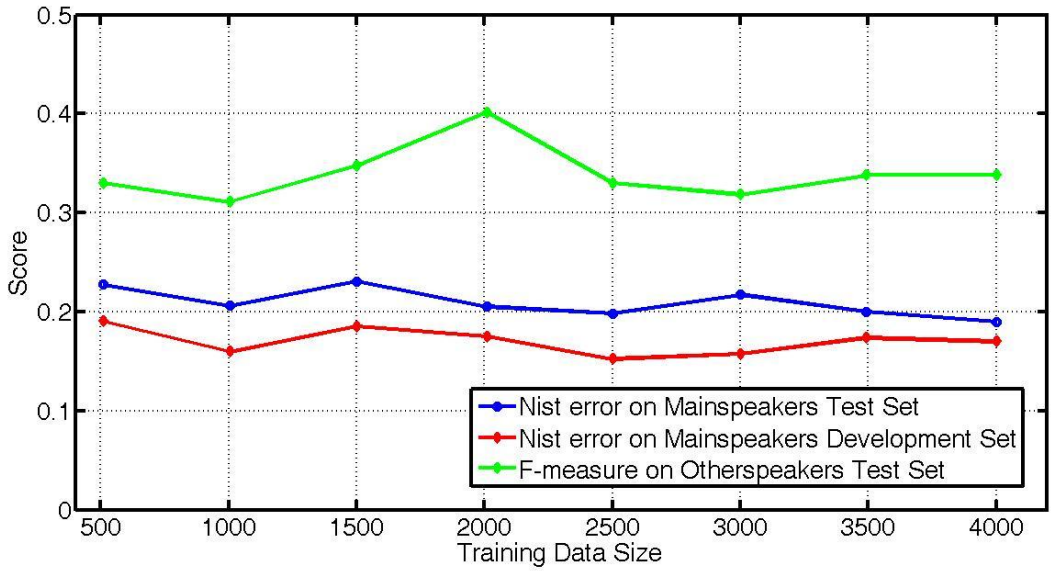


Figure 8.26 Nist error rate on model 1 feature set on development set belongs to main speakers (red color), test set belongs to main speakers (blue color), and test set belongs to other speakers (green color).

8.3.2 Tests performed with model 2 feature set

Training Set Size	**Iteration Number**	**F-measure on Development Set**	**Nist on Development Set**	**F-measure on Test Set**	**Nist on Test Set**
508Words	1	0.6063241106	1.26395939086294	0.8403730115194	0.2957317073170
	15	0.8438356164	0.2893401015228	**0.6418835192069**	**0.5579150579150**
1004Words	1	0.6063241106	1.26395939086294	0.8699360341151	0.2479674796747
	172	0.86320109439	0.2538071065989	**0.7235023041474**	**0.46332043320463**
1500Words	1	0	1	0.8639744952178	0.2601626016260
	1148	0.85733695652	0.2664974619289	**0.710495963091**	**0.4845559845559**
2008Words	1	0	1	0.881663113006	0.2256097560975
	172	0.8869448183	0.213197969543	**0.735260115606**	**0.4420849420849**
2500Words	1	0	1	0.8815165876777	0.2286585365853
	219	0.8797297297	0.22588832487	**0.7697368421052**	**0.4054054054054**
3000Words	1	0	1	0.879957127545	0.2276422764227
	165	0.888432580	0.20685279187	**0.7556561085972**	**0.4169884169884**
3493Words	1	0	1	0.881720430107	0.2235772357723
	154	0.8852005532	0.2106598984771	**0.71864406779661**	**0.4806949806949**
4000Words	1	0	1	0.894764674775	0.2022357723577
	2518	0.8943089430	0.19796954314	**0.7360178970917**	**0.4555984555984**

Table 8.16 Performance measurements on model 2 set of features. In the test column, bold results are corresponds to tests VOA Speakers 3-6.

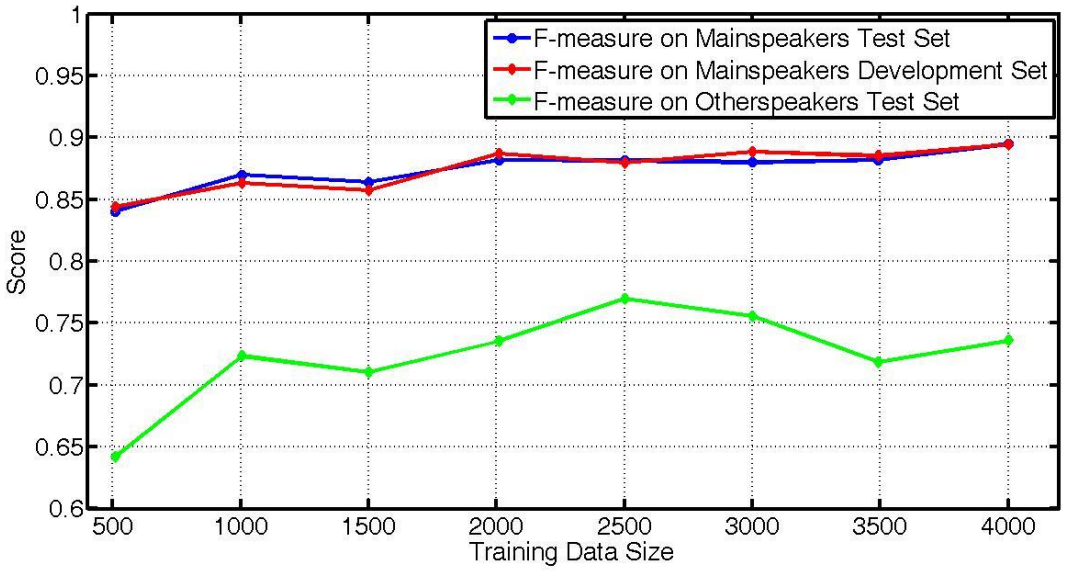


Figure 8.27 F-measure score on model 2 feature set on development set belongs to main speakers (red color), test set belongs to main speakers (blue color), and test set belongs to other speakers (green color).

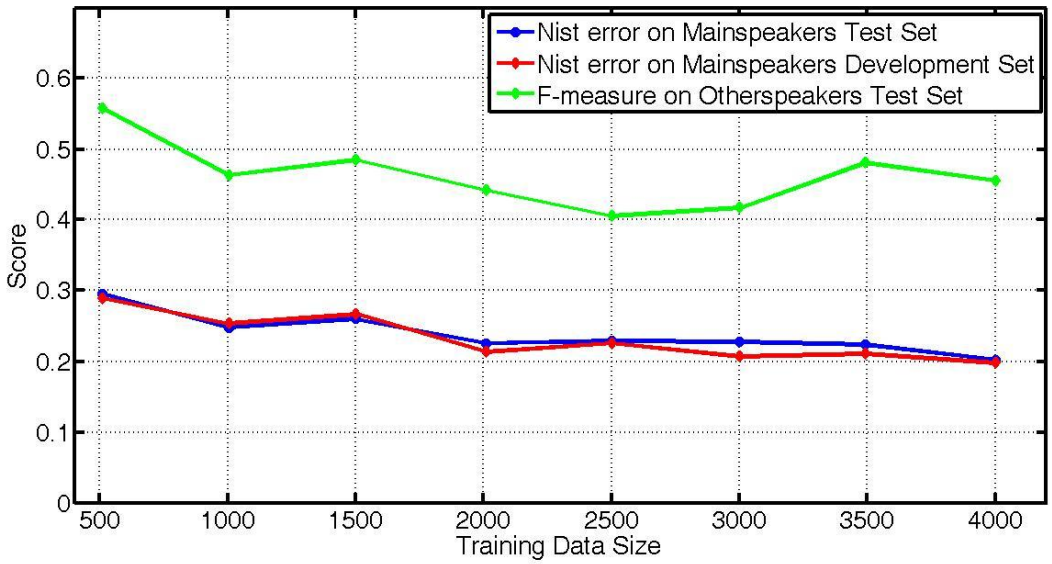


Figure 8.28 Nist error rate on model 2 feature set on development set belongs to main speakers (red color), test set belongs to main speakers (blue color), and test set belongs to other speakers (green color).

GEN$: male, female.
PAUSE_DUR: continuous.
FLAG$: 0, SUSP.
LAST_VOWEL_DUR: continuous.
NORM_LAST_RHYME_DUR: continuous.
MEAN_F0: continuous.
MEAN_STYLFIT_F0: continuous.
MEAN_STYLFIT_F0_WIN: continuous.
MEAN_F0_NEXT: continuous.
MEAN_STYLFIT_F0_NEXT: continuous.
MEAN_F0_NEXT_WIN: continuous.
MEAN_STYLFIT_F0_NEXT_WIN: continuous.
PATTERN_BOUNDARY$: X, ff, fr, rf, rr.
WORD_DUR: continuous.
WORD_AV_DUR: continuous.
NORM_WORD_DUR: continuous.
LAST_VOWEL_DUR_Z: continuous.
LAST_VOWEL_DUR_N: continuous.
LAST_VOWEL_DUR_ZSP: continuous.
LAST_VOWEL_DUR_NSP: continuous.
LAST_RHYME_DUR_PH_ND: continuous.
LAST_RHYME_DUR_PH_NR: continuous.
LAST_RHYME_NORM_DUR_PH_ND: continuous.
LAST_RHYME_NORM_DUR_PH_NR: continuous.
F0K_WORD_DIFF_HIHI_NG: continuous.
F0K_WORD_DIFF_HILO_NG: continuous.
F0K_WORD_DIFF_LOLO_NG: continuous.
F0K_WORD_DIFF_LOHI_NG: continuous.
F0K_WORD_DIFF_MNMN_NG: continuous.
F0K_WIN_DIFF_HIHI_NG: continuous.
F0K_WIN_DIFF_HILO_NG: continuous.
F0K_WIN_DIFF_LOLO_NG: continuous.
F0K_WIN_DIFF_LOHI_NG: continuous.
F0K_WIN_DIFF_MNMN_NG: continuous.
ENERGY_WORD_DIFF_HIHI_NG: continuous.
ENERGY_WORD_DIFF_HILO_NG: continuous.
ENERGY_WORD_DIFF_LOLO_NG: continuous.
ENERGY_WORD_DIFF_LOHI_NG: continuous.
ENERGY_WORD_DIFF_MNMN_NG: continuous.
ENERGY_WIN_DIFF_HIHI_NG: continuous.
ENERGY_WIN_DIFF_HILO_NG: continuous.
ENERGY_WIN_DIFF_LOLO_NG: continuous.
ENERGY_WIN_DIFF_LOHI_NG: continuous.
ENERGY_WIN_DIFF_MNMN_NG: continuous.
AVG_PHONE_DUR_Z: continuous.
MAX_PHONE_DUR_Z: continuous.
AVG_PHONE_DUR_N: continuous.
MAX_PHONE_DUR_N: continuous.

Table 8.17 Model 2 feature set.

The experiment results in section 8.3 shows that, model 1 features succeed better than model 2 features. Especially, in the sentence segmentation for unknown speaker task, model 1 performs much better.

To conclude, in this work we have added sentence segmentation property to the output of the ASR system. We have used strong learners which are trained by using several set of prosodic features. All of the used computer programs in this work are open source and similar systems can be developed for not only Turkish language but also the other languages. We have observed that, using F0, duration and energy features together, performs maximum performance. This work can be considered as the first step of further ASR applications such as topic segmentation and summarization. This work can also be used at online applications such as online subtitles with punctuation signs in a television show.

References

[1] A Chronology of Communication Related Events
http://people.seas.harvard.edu/~jones/history/comm_chron1.html

[2] Lawrence R. Rabiner, Ronald W. Schafer, *"Theory and Applications of Digital Speech Processing"*, First Edition, Pearson International Edition.

[3] TürkDilKurumu, SesHarfveAlfabe
http://www.tdk.gov.tr/index.php?option=com_content&view=article&id=174:Ses-Harf-ve-Alfabe&catid=50:yazm-kurallar&Itemid=132

[4] In Search of Music's Biological Roots
http://www.dukemagazine.duke.edu/issues/050608/music2.html

[5] J.L. Flaganan, *Speech Analysis, Synthesis, and Perception*, 2nd edition, Springer-Verlag, New York, 1972.

[6] Lawrence Rabiner, Biing-Hwang Juang, *Fundamentals of Speech Recognition,*Prenctice-Hall International Edition IBSN 0-13-285826-6

[7] J.L. Flaganan, C.H. Coker, L.R. Rabiner, R.W. Schafer and N. Umeda:Syntetic Voices for Computers, *IEEE Spectrum, Vol. 7, No. 10, pp. 22-45*, October 1970.

[8] Prof. Dr. M. AyhanZeren, *MüzikFiziği (Pyhsics of Music)*, Pan Yayıncılık: 34, ISBN 975-7652-46-6, October 1995.

[9] Professor em. Peter W. Alberti, *The Anatomy and Physiology of the Ear and Hearing*, University of Toronto, Canada.

[10] Hallowell, Davis and S. Richard Silverman (Ed.), *Hearing and Deafness*, 3rd Edition, Holt, Rinehart and Winston, 1970.

[11] Basilar of Membrane: Analysis of sounds.

http://www.britannica.com/EBchecked/media/537/The-analysis-of-sound-frequencies-by-the-basilar-membrane

[12] G. von Bekesy, *Experiments in Hearing*, McGraw-Hill, New York, 1960.

[13] N.Y.S. Kiang and E.C. Moxon:Tails of Tunning Curves of Auditory Nerve Fibers, *J. of Acoustical Society of America*, Vol.55, pp. 620-630, 1974.

[14] W.S.Rhode, C.D.Geisler, and D.T. Kennedy: Auditory Nerve Fiber Responses to Wide-Band Noise and Tone Combinations, *J. Neurophysiology, Vol. 41*, pp. 692-704, 1978.

[15] M.B. Sachs and E.D. Young: Encoding of Steady State Vowels in the Auditory Nerve: Represenation in Terms of Discharge Rates, *J. of Acoustical society of America, Vol. 66*, pp. 470-479, 1979.

[16] E. Zwickler and H. Fastl, *Psychoacoustics, 2nd ed*., Springer, Berlin, Germany, 1999.

[17] O. Ghitza, *Auditory Nerve Representation as a Basis for Speech Processing, Advances in Speech and Signal Processing,*S.Fuuri and M.M. Sondhi (eds.), Marcel-Dekker, NY, pp.453-485, 1991.

[18] H. Hermansky: Auditory Modeling in Automatic Recognition of Speech, *Proc. First European Conf. on Signal Analysis and Prediction, pp.17-21*, Prague, Czech Republic, 1997.

[19] R.F. Lyon: A Computational Model of Filtering, Detection, and Compression in the Cochlea, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 1282-1285, 1982.

[20] S. Seneff, A Joint Synchorony: Mean-Rate Model of Auditory Speech Processing, *J. of Phonetics*, Vol. 16, pp. 55-67, 1988.

[21] W. Koenig, H. K. Dunn, and L. Y. Lacy: The Sound Spectrograph, *J. of Acoustical Society of America, Vol. 18*, pp. 19-49, July 1946.

[22] R. K. Potter, G. A. Kopp, and H. C. Green Kopp, *Visible Speech*, D. Van Nostrand Co., New York, 1947. (Republished by Dover Publications, Inc., 1966.)

[23] TürkAlfabesi'nin IPA ilesöylenişi. http://tr.wikipedia.org/wiki/T%C3%BCrk_alfabesi

[24] SesinDoğasıveOluşumu, The Nature of Sound.

www.jandarma.tsk.tr/kriminal/turkish_internet/.../bilarinde5.pdf

[25] MihricanAynacı, *TürkiyeTürkçesindeSesEtkileşimleri,* KocaeliÜniversitesi, SosyalBilimlerEnstitüsü, TürkDiliveEdebiyatı Ana Bilim Dalı, YüksekLisansTezi.

[27] John R. Deller, John H. L. Hansen, John G. Proakis, *Discrete-Time Processing of Speech Signals,* IEEE Press 445 Hoes Lane, P. O. Box 1331, Piscataway, NJ 08855-1331 IBSN 0.7803-5386-2

[28] R. G. Leonard:A Database for Speaker-Independent Digit Recognition, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* pp.42.11.1-42.11.4, 1984.

[29] W. Ward: Evaluation of the CMU ATIS system, *Proc. DARPA Speech and Natural Language Workshop,* pp.101-105, February 1991.

[30] J. J. Godfrey, E. C. Holliman, and J. McDaniel: SWITCHBOARD: Telephone Speech Corpus for Research and Development, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 517-520, 1992.

[31] Zhongqiang Huang, Lei Chen, Mary P. Harper, *Purdue Prosodic Feature Extraction Tool on Praat, Spoken Language Processing* Lab. School of Electrical and Computer Engineering Purdue University, West Lafayette, June 23, 2006.

[32] Yang Liu, Nitesh V. Chawla, Mary P. Harper, Elizabeth Shriberg, Andreas Stolcke, *A Study in Machine Learning from Imbalanced Data for Sentence Boundary Detection in Speech*, 14 March 2005.

[33] Elizabeth Shriberg, Andreas Stolcke, DilekHakkani-Tür, GökhanTür:*Prosody-based automatic segmentation of speech into sentences and topics, Speech Communication* 32 (2000) 127-154

[34] Zhongqiang Huang, Lei Chen, Mary Harper, *An Open Source Prosodic Feature Extraction Tool, School of Electrical and Computer Engineering*, Purdue University, West Lafayette, IN 47907.

[35] Luciana Ferrer, *Prosodic Features for the Switchboard Database*, Speech Technology and Research Lab. SRI International, Menlo Park, CA 94025, July 17, 2002.

[36] Talkin, D. (1995):A Robust Algorithm for Pitch Tracking (RAPT). In Kleijn, W.B. and Paliwal, K. K. (Eds.), *Speech Coding and Syntesis*. New York: Elsevier.

[37] M. Kemal Sonmez, Larry Heck, Mitchel Weintraub, Elizabeth Shriberg:*A lognormal Tied Mixture Model of Pitch for Prosody-Based Speaker Recognition*.

[38] Ümit Güz, Member, IEEE, SebastienCuendet, DilekHakkani-Tür, Senior Member, IEEE, and GökhanTür, Senior Member, IEEE:Multi-Viwe Semi-Supervised Learning for Dialog Act Segmentation of Speech, *IEEE Transactions on Audio, Speech and Language Processing Vol 18 No 2* pp. 320-329, February 2010.

[39] Artur Ferreira, Instituto de Telecomunicaçoes, *Survey on Boosting Algorithms for Supervised and Semi-supervised Learning*, 12 October 2007.

[40] Robert E. Schapire, *The Boosting Approach to Machine Learning An Overwiew*: AT&T Labs - Research, Shannon Laboratory, December, 19, 2001.

[41] Open-source implementation of Boostexter (Adaboost based classifier) http://code.google.com/p/icsiboost/

[42] EthemAlpaydin,*Introduction to Machine Learning*, The MIT Press, Cambridge, Massachusetts London, England, IBSN: 0-262-01211-1

[43] Amerika'nınSesi. http://www.voanews.com/turkish/

[44] BUSIM, Signal and Image Processing Lab. http://www.busim.ee.boun.edu.tr/

[45] Elliot Sound Products, Frequency, Amplitude and dB, Rod Ellitot (ESP) http://sound.westhost.com/articles/fadb.htm

[46] What is decibel, School of Physics, Music Acoustics. http://www.phys.unsw.edu.au/jw/dBNoFlash.html

[47] Bellman, R., and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton, N.J.: Princeton University Press, 1962.

[48] C. S. Myers, and L. R. Rabiner, "A level building dynamic time warping algorithm for connected word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Washington, D. C., pp. 224-227, Apr. 1979.

[49] J. S. Bridle and M. D. Brown, "Connected word recognition using whole word templates", *Proceedings of the Institute for Acoustics, Autumn Conference*, pp. 25-28, Nov. 1979.

[50] J. S. Bridle, R. M. Chamberlain and M. D. Brown, "An algorithm for connected word recognition", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing,* Paris vol. 2, pp. 899-902, 1982.

[51] I. B. Pawate, M. L. McMahan, R. H. Wiggins et al. "Connected word recognizer on a multiprocessor system", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* Dallas, Tex., vol. 2, pp. 1151-1154, 1987.

[52] S. M. Miller, D.P. Morgan, H.F. Silverman et al. "Real-time evaluation system for a real-time connected speech recognizer", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* Dallas, Tex., vol. 2, pp. 801-804, 1987.

[53] H. Stark, J. W. Woods, *Probability and Random Process with Applications to Signal Processing,* Third Edition, Prentice Hall, Upper Saddle River, New Jersey 07458, ISBN 0-13-020071-9.

[54] S.Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valcthev, P. Woodland, *The HTK book* Version 3.4 http://htk.eng.cam.ac.uk/

[55] F. Jelinek and R.L. Mercer,"Interpolated Estimation of Markov Source Parameters From Sparse Data", *Pattern Recognition in Practice,* E.S. Gelsema and L.N. Kanal, Eds., North-Holland Pub. Co., Amsterdam, pp.381-397, 1980.

[56] Formant Frekanslarve F2-F1 farkınınanlamı (kalın/ince, dar/geniş) http://egurbuz.wordpress.com/2011/01/28/141/

[57] D. O. Shaughnessy, "Linear Predictive Coding, One popular technique of analyzing certain physical signals." *IEEE Potentials* February 1988.

[58] Praat: Doing phonetics by computer http://www.fon.hum.uva.nl/praat/

Curriculum Vitae

Doğan Dalva was born on 28th May 1986 in Istanbul, Turkey. He received his BS degree in Electronic Engineering in 2009. Since 2011 he has been working as a research assistant at Işık University.