DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF TURKISH SPEECH RECOGNITION ENGINE

by
Rıfat AŞLIYAN

July, 2008
İZMİR

DESIGN AND IMPLEMENTATION OF TURKISH SPEECH RECOGNITION ENGINE

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of
Philosophy in Computer Engineering

by
Rıfat AŞLIYAN

July, 2008
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN AND IMPLEMENTATION OF TURKISH SPEECH RECOGNITION ENGINE"** completed by **RIFAT AŞLIYAN** under supervision of **PROF. DR. TATYANA YAKHNO** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Tatyana YAKHNO

Supervisor

Asst. Prof. Dr. Adil ALPKOÇAK	Asst. Prof. Dr. Damla KUNTALP
Thesis Committee Member	Thesis Committee Member
Assoc. Prof. Dr. A. Fevzi BABA	Asst. Prof. Dr. Gökhan DALKILIÇ
Examining Committee Member	Examining Committee Member

Prof. Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Prof. Dr. Tatyana YAKHNO, whose expertise, understanding, and patience, added considerably to my graduate experience.

I would like to thank the other members of my committee, Asst. Prof. Dr. Adil ALPKOÇAK, and Asst. Prof. Dr. Damla KUNTALP for the assistance they provided at all levels of my study.

Special thanks go to my friend, Korhan GÜNEL, for his comments and help. I want to thank Prof. Dr. Hatice KANDAMAR for her support.

I would also like to thank my parents, Fatma and Mehmet AŞLIYAN, for the support they provided me through my entire life.

Rıfat AŞLIYAN

DESIGN AND IMPLEMENTATION OF TURKISH SPEECH RECOGNITION ENGINE

ABSTRACT

In this thesis, we have designed and implemented syllable based Turkish speech recognition systems based on Linear Time Alignment (LTA), Dynamic Time Warping (DTW), Artificial Neural Network (ANN), Hidden Markov Model (HMM) and Support Vector Machine (SVM). These speaker dependent and isolated word recognition systems consist of five main parts: Preprocessing, feature extraction, training, recognition and postprocessing. Preprocessing includes some operations such as speech signal smoothing, windowing and syllable end-point detection. In feature extraction, we have used speech features as mel frequency cepstral coefficients, linear predictive coefficients, parcor, cepstrum and rasta coefficients. In training stage for HMM, SVM and ANN, every syllable of the words in the dictionary is trained, and the syllable models are generated. In recognition stage, every syllable in the word utterence is compared with the syllable models. So, the recognized syllables are determined and ordered. Then, the recognized syllables are concatenated with each other. In postprocessing operation, we have developed the system which is based on Turkish syllable n-gram frequencies. The system decides whether the recognized word is Turkish or not. If the word is Turkish, then it is new recognized word.

The system is middle scaled speech recognition because the system dictionary has 200 different Turkish words. After the system is tested on 2000 spoken words, we have seen that the word error rate of the system is about 5.8% for DTW, 12% for ANN, 8.8% for LTA, 17.4% for HMM and 9.2% for SVM with postprocessing. System recognition rate increased approximately 14% using postprocessing.

Keywords: Turkish speech recognition, syllable based speech recognition, Hidden Markov Model, Linear Time Alignment, Dynamic Time Warping, Artificial Neural Network, Support Vector Machine, Turkish misspelled words, Turkish syllable n-gram.

TÜRKÇE KONUŞMA TANIMA SİSTEMİNİN TASARIMI VE GERÇEKLEŞTİRİLİMİ

ÖZ

Bu tezde, konuşmacıya bağımlı hece tabanlı Türkçe konuşma tanıma sistemi uygulamaları gerçekleştirilmiştir. Bu sistemlerde, konuşma tanıma yöntemlerinden Doğrusal Zaman Hizalama (DZH), Dinamik Zaman Bükmesi (DZB), yapay sinir ağlarından Çok Katmanlı Algılayıcı (ÇKA), Saklı Markov Modeli (SMM) ve Vektör Destek Makineleri (VDM) kullanılmıştır. Ayrık sözcük tanıma sistemi genel olarak önişleme, öznitelik çıkarılması, hecelerin eğitimi, tanıma ve önişleme süreçlerinden oluşmaktadır. Önişlemede, dijital sinyallerin düzleştirilmesi, pencereleme ve hece sınırların tespiti işlemleri yapılır. Hecelerin mfcc, lpc, parcor, cepstrum ve rasta öznitelikleri elde edildikten sonra ÇKA, VDM ve SMM kullanılarak eğitilir. Her yöntem için hece modelleri oluşturulur. Sözcük tanıma safhasında, tanınması istenen sözcüğün heceleri hece modelleri ile karşılaştırılır. En çok benzeyen heceler tespit edilip sıralandırılır. En çok benzeyen heceler birbirine eklenerek tanınan sözcük bulunur. Artişlemede ise bu tanınan sözcüğün Türkçe olup olmadığına bakılır. Eğer bu sözcük Türkçe ise tanıma işlemi biter. Fakat Türkçe değilse bir sonraki heceler eklenerek yeni sözcük oluşturulur. Bu işlemlere Türkçe sözcük bulunana kadar devam edilir. Bir sözcüğün Türkçe olup olmadığının tespiti için hece n-gram frekansları kullanılmıştır.

Orta dağarcıklı konuşma tanıma sisteminin sözlüğünde 200 Türkçe sözcük bulunmaktadır. Her bir sözcük 10 defa kaydedilerek 2000 sözcüklü test veritabanı oluşturuldu ve test işlemi yapıldı. Sistemin başarımını ölçmek için sözcük hata oranı (word error rate) kullanıldı. Sözcük hata oranı, DZB için %5,8, ÇKA için %12, SMM için 17,4, DZH için %8,8 ve DVM için %9,2 olarak bulunmuştur. Artişleme, sistemin başarımını yaklaşık olarak %14 oranında artırmıştır.

Anahtar sözcükler: Türkçe konuşma tanıma, hece tabanlı konuşma tanıma, Dinamik Zaman Bükmesi, Saklı Markov Modeli, Çok Katmanlı Algılayıcı, Vektör Destek Makineleri, hatalı yazılmış sözcük tespiti, Türkçe hece istatistiği, Türkçe hece n-gram.

CONTENTS

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Speech is the primary means of communication between people. For reasons ranging from technological curiosity about the mechanisms for mechanical realization of human speech capabilities, to the desire to automate simple tasks inherently requiring human-machine interactions, research in speech recognition and speech synthesis by machine has attracted a great deal of attention over the past six decades.

Speech recognition is the process by which a computer converts an acoustic speech signal to text. This process is important to virtual reality because it provides a fairly natural and intuitive way of controlling the simulation while allowing the user's hands to remain free. Speech recognition allows making it easier both to create and to use information. Text is easier to store, process and consume, both for computers and for humans, but writing text is slow and requires some intention. Speech is easier to generate, it's intuitive and fast, but listening to speech is slow, it's hard to index speech, and easy to forget.

Great advance has been achieved in last ten years in the speech recognition technology, but 100% reliable speech recognition systems are not developed yet. The most limiting factor in speech processing applications is the variability of speech signal characteristics from trial to trial, the variability of recording and transmission conditions, and the variations generated by the speaker, either deliberately or accidentally. However, the primary bottleneck is the spectral and pitch changes arising from emotional changes of the speakers.

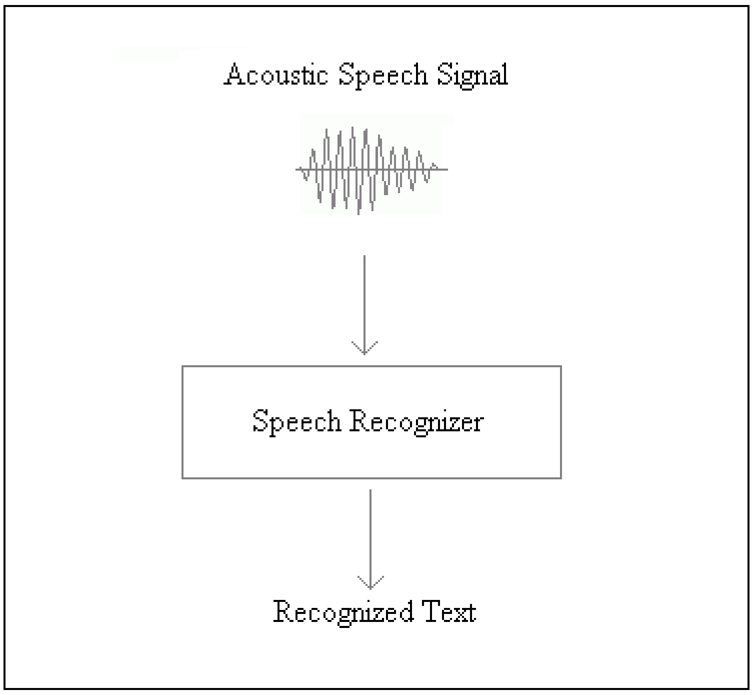


Figure 1.1 General structure of speech recognition.

In a simplified way, the general speech recognition procedure is shown in Figure 1.1. Speech recognizer includes the operations as preprocessing, feature extraction, training, recognition and postprocessing. After the speech recognizer takes the acoustic speech signal as an input, the output of the recognizer will be the recognized text.

The most common approaches to speech recognition can be divided into two classes: “template based approach” and “model based approach”. Template based approaches as LTA and DTW are the simplest techniques and have the highest accuracy when used properly, but they also suffer from the most limitations. As with any approach to speech recognition, the first step is for the user to speak a word or phrase into a microphone. The electrical signal from the microphone is digitized by an analog-to-digital converter. The system attempts to match the input with a digitized voice sample, or template. This technique is a close analogy to the traditional command inputs from a keyboard. The system contains the input template, and attempts to match this template with the actual input. Model based approaches as HMM and ANN tend to extract robust representations of the speech references in a statistical way from huge amounts of speech data. Model based approaches are currently the most popular techniques. However, when the size of the vocabulary is small and the amount of training data is limited, template based approaches are still very attractive. Even though most of the time these approaches are used separately, some of these techniques are complementary and can be combined in a very efficient way.

Another way to differentiate between speech recognition systems is by determining if they can handle only discrete words, connected words, or continuous speech. Most voice recognition systems are discrete word systems, and these systems are easiest to implement. For this type of system, the speaker must pause between words. This is fine for situations where the user is required to give only one word responses or commands. In a connected word voice recognition system, the user is allowed to speak in multiple word phrases, but he or she must still be careful to articulate each word and not slur the end of one word into the beginning of the next word. Totally natural, continuous speech includes a great deal of co-articulation, where adjacent words run together without pauses or any other apparent division between words.

Speech recognition system is speaker dependent or speaker independent. A speaker dependent system is developed to operate for a single speaker. These systems are usually more accurate. A speaker independent system is developed to operate for any speaker. These systems are the most difficult to develop, most expensive and accuracy is lower than speaker dependent systems.

The size of vocabulary is another key point in speech recognition applications. The size of vocabulary of a speech recognition system affects the complexity, processing requirements and the accuracy of the system. Some applications only require a few words such as only numbers; others require very large dictionaries such as dictation machines. According to vocabulary size, speech recognition systems can be divided into three main categories as small vocabulary recognizers (smaller than 100 words), medium vocabulary recognizers (around 100-1000 words) and large vocabulary recognizers (over 1000 words).

1.2 Speech Recognition History

First speech recognition studies started in the late 40s and early 50s, simultaneously in Europe with J. Dreyfus-Graf and in the U.S.A. with K. H. Davis and his colleagues at Bell Laboratories. Dreyfus-Graf, J. (1952) designed his first "Phonetographe" in 1952. This system transcribed speech into phonetic "atoms". Davis, K., et al, (1952) designed the first speaker dependent, isolated digit recognizer. This system used a limited number of acoustic parameters based on zero-crossing counting.

A research group at Bell Laboratories adopted a phonetic decoding approach to design a word recognizer based on segmentation in phonetic units (Dudley & Balashek, 1958). At the same period, a system was designed on the basis of the distinctive features proposed in Jakobson, R., et al. (1952), for the speaker independent recognition of vowels (Wiren & Stubbs, 1956). Another phonetic approach was used at RCA laboratories in the first "phonetic typewriter" capable of recognizing syllables dictated in isolation by a single speaker (Olson & Belar, 1956). A rudimentary phoneme recognizer was developed at University College, London (Denes, 1959). This system was the first to incorporate linguistic knowledge under the form of statistical information about allowable sequences of two phonemes in English.

All the above mentioned systems were electronic devices. The first experiments on computer based speech recognition were carried out in the late 50s and early 60s, especially Lincoln Laboratory for the speaker independent of ten vowels (Forgie & Forgie, 1959). At the same period, the first Japanese systems were developed, still as special purpose hardware for vowel (Suzuki & Nakata, 1961) or phoneme identification (Sakai & Doshita, 1962), and for digit recognition (Nagate et al., 1963). But the systems actually correspond to the generalization of the use of digital processing and computers.

This decade was also marked by two major milestones in the history of speech recognition methodology. The first is preliminary development of techniques normalization in speech pattern matching. Acoustic feature abstraction was proposed in Martin et al., 1964, and the basic concepts of dynamic time warping using dynamic programming were proposed by Russian researchers (Slutsker, 1968; Vitsyuk, 1968). The second is the recognition of continuous speech by dynamic tracking of phonemes from Stanford University (Reddy, 1966). It led to the speaker dependent recognition of sentences with five vocabularies of 561 words (Vicens, 1969).

The 1970s were very active period for speech recognition with two distinct types of activities. First is the understanding of large vocabularies, continuous speech, based on the use of high level knowledge such as lexical and syntactic to compensate for the errors in phonetic decoding. The main contributions of these artificial intelligence projects were more in software architecture of knowledge based systems (Lesser et al., 1975). Such systems were primarily developed in the framework of the ARPA. The goal of speech understanding research project from 1971 to 1976 was understood of continuous speech sentences from a vocabulary of about 1000 words produced by one speaker. Several systems were developed which more or less fulfilled the initial goal: HARPY (Lowerre, 1976) and HEARSAY II (Lesser et al., 1975) at Carnegie Mellon University, and HWIM (Wolf & Woods, 1977). Similar systems were proposed in France: MYRTILLE I (Haton & Pierrel, 1976), KEAL (Marcier, 1977). The second is the recognition of isolated words based on pattern recognition template based methods (Velichko & Zagoruyko, 1970). Several basic techniques still in use today were introduced during this decade. The first is elastic matching of speech patterns by dynamic time warping algorithms. These algorithms were first developed in USSR (Slutsker, 1968; Vistsyuk, 1968) and in Japan (Sakoe & Chiba, 1971). Sub-optimal, but less time consuming versions were also proposed (Haton, 1974). The second technique is clustering algorithms adapted from data analysis methods in order to design speaker independent systems (Rabiner et al., 1979). The third is speech analysis based on linear predictive coding (lpc) instead of the classical fast fourier transform (fft) or filter bank methods (Itakura, 1975).

In the late 1970s, important progress was made with the implementation of speech recognition systems on microprocessor boards. This technological advance made possible the commercialization of the first low cost speech recognizers.

The 1980s were marked by series of important milestones. The first one is the extension of dynamic programming to connected word recognition such as (Sakoe, 1979) and onepass methods (Bridle & Brown, 1979; Lee & Rabiner, 1989). The second one is the shift in methodology from template based methods to statistical modeling based on HMMs (Ferguson, 1980; Rabiner, 1989). These methods were developed in the 1970s (Baker, 1975a, Jelinek, 1976) for continuous speech recognition. The third one is the reintroduction of neural networks techniques (Lippmann, 1987). The first neural network models as the perceptron were proposed in the 1950s, and then reappeared in the late 1980s. The fourth one is the acoustic-phonetic decoding of continuous speech using knowledge based approaches. Expert system technology has been advocated to design phonetic decoders based on the expertise of phoneticians in spectrogram reading (Cole et al., 1980). The fifth one is the recording of large databases such as TIMIT (Fisher et al., 1986) which directly contributed to the advances made in speech recognition. During this same decade, an ARPA program contributed to substantially improve the accuracy of continuous speech recognition for medium size vocabulary with resource management task.

The 1990s and 2000s have experienced a continuous and an extension of the ARPA program towards two main directions. These are the introduction of natural language and user system dialog in an air travel information application, and the extension of speech recognition systems to large vocabularies for dictation purposes (Makhoul & Schwatz, 1994). Another major trend of these years is an important increase in the use of speech recognition technology within public telephone networks (Wilpon, 1994). As a result, an increasing interest of speech processing under noisy or adverse conditions, as well as for spontaneous speech recognition emerged.

Some general conclusions can be drawn from this past experience of six decades in speech recognition research and development: First, present systems are based upon models and techniques that appeared quite early in the history of speech recognition. Second, transforming a laboratory prototype with excellent accuracy into a reliable commercial system is a long, and yet not totally mastered process. Third, the performance of today's best systems is more than an order of magnitude in error rate from human performance. Finally, the general solution to the problem will not be found suddenly by an ingenious researcher. Rather, it will necessitate a long and tedious multi-disciplinary work.

1.3 A Survey of Turkish Speech Recognition

Today, there are several speech recognition studies on Turkish. But, Turkish speech recognition studies have increased in the past decade. We have mentioned some of them as the followings.

Arturner (1994) firstly constructed Turkish codebook for each Turkish phoneme. He then designed and implemented a Turkish speech phoneme clustering system using self organizing feature map.

Meral (1996) developed speech recognition system based on pattern comparison techniques. He used lpc speech feature and dynamic time warping method. The WER of the system is about 0% on the vocabulary (26 Turkish words).

Özkan (1997) implemented a speech recognition system for Turkish connected numerals. The system is speaker dependent isolated word recognition using dynamic time warping method. The WER of the system is about 0%. He used lpc speech recognition feature.

Mengüşoğlu (1999) designed and implemented a rule based speech recognition system for Turkish. It is used rasta and mel-cepstrum features for the phoneme-based and speaker dependent system. This isolated word system was tested on 248 words. For mel-cepstrum and rasta, the WER is 11.4% and 8.8% respectively.

Yılmaz (1999) proposed a large scaled Turkish speech recognition system which is speaker dependent. Each word is modeled with triphones using hidden markov model. The WER of the system which is tested with 1000 words is about 10%.

Karaca (1999) has developed a Turkish isolated word recognition system under noisy environments. This system is word-based and speaker independent. According to lpc and rasta features, each word is modeled using hidden markov model. The system is tested on the vocabulary which has 130 Turkish words. The WER is 26.5%.

Koç (2002) studied on acoustic feature analysis for robust speech recognition. The system is based on hidden markov model and uses mfcc and rasta-plp features.

Arısoy & Dutağacı (2006) have developed a unified language model for large vocabulary continuous speech recognition of Turkish using hidden markov model. The developed systems are speaker dependent and speaker independent. Letter error rates (LER) are approximately 28% for a speaker independent system and 20% for a speaker dependent system.

Avcı (2007) presented an automatic system for word recognition using real Turkish word signals. A Discrete Wavelet Neural Network (DWNN) model is used, which consists of two layers: discrete wavelet layer and multi-layer perceptron. The discrete wavelet layer is used for adaptive feature extraction in the time-frequency domain and is composed of Discrete Wavelet Transform (DWT) and wavelet entropy. The performance of the used system is evaluated by using noisy Turkish word signals. The WER is about 8% for small vocabulary (15 words).

Salor & Pellom (2007) developed Turkish speech corpora and recognition tools developed by porting SONIC: Towards multilingual speech recognition. The system is speaker independent based on HMM triphone model. The speech recognition feature is mfcc. The phone recognition error rate is about 29.2%.

1.4 The Thesis Perspective

In this thesis, we introduced a new approach for Turkish speech recognition. We have designed and implemented some syllable based speech recognition systems based on LTA, DTW, HMM, ANN and SVM methods and evaluated the efficiency of the systems with the new approach.

Turkish language, that is one of the least studied language in the speech recognition field, has different characteristics than European languages which require different language modeling technique (Hakkani, Oflazer & Tür, 2000; Oflazer, 1994). Since Turkish is an agglutinative language, the degree of inflection is very high. So, many words are generated from a Turkish word's root by adding suffixes. That's why, word based speech recognition systems are not adequate for large scaled Turkish speech recognition and Turkish is syllabified language. We have developed syllable based isolated word speech recognition systems. First, acoustic signal of the word utterance as an input is applied by preprocessing. The utterance is divided into syllable utterances by the endpoint detection algorithm using signal's energy and zero-crossing point. Each syllable of the word is separately trained and modeled by speech recognition methods and recognized. The recognized syllables are sorted and the most similar syllables are concatenated in order. So the recognized word is found in that way. After that, we have applied postprocessing operation which decides whether or not the recognized word is Turkish. This new approach used in this thesis increased the accuracy rate about 14%. For this purpose, we have developed TASA (Turkish Automatic Syllabifying Algorithm) which spells the Turkish words into syllables (Aşlıyan & Günel, 2005). TASA syllabifies the words by approximately 100% success rate. The system decides whether a word is Turkish or not using syllable *n*-gram language model (Aşlıyan, Günel & Yakhno, 2007).

1.5 Structure of the Thesis

In Chapter 2, we have mentioned about speech recognition in detail. The general procedure of speech recognition system which consists of speech acquisition, preprocessing, feature extraction, acoustic and language model is introduced.

In Chapter 3, we give the definitions of speech recognition methods as linear time alignment, dynamic time warping, artificial neural network, hidden markov model and support vector machine. The mathematical formulation of the speech features as linear predictive coding coefficients, mel frequency cepstral coefficients, rasta, cepstrum and parcor coefficients are explained in detail.

In Chapter 4, we have explained Turkish syllable *n*-gram analysis. Turkish Automatic Syllabifying Algorithm (TASA) and Turkish syllable statistics have been presented.

In Chapter 5, we have mentioned how to be decided whether a word is Turkish or not using Turkish syllable *n*-gram frequencies.

In Chapter 6, the speech recognition experiments are described using the most efficient methods and features. In addition, the experimental results are given and compared.

We have presented the conclusions and future directions of the thesis in Chapter 7.

CHAPTER TWO
SPEECH RECOGNITION

In this chapter we describe the main steps of speech recognition as speech acquisition, preprocessing, feature extraction, recognition operation, acoustic and language model.

2.1 Definition of Speech Recognition

Speech recognition is the process that allows humans communicate with computers by speech. The purpose is to transmit the idea to the computer.

There are a lots of other communication methods between humans and computers which require some input devices. Keyboards, mouses, touch screens are the most classical examples of input devices with high accuracies. Those input devices are not efficient enough in some conditions, especially when the use of hands is not possible. They need also a certain level of expertise for being used.

There are some other recent researches on human-computer interaction with brain waves but this research field is still in its beginning phase (Anderson & Kirby, 2003). Since speech is the most natural way of communication between humans, it is important to make possible the use of speech to communicate with computers. By enabling speech recognition, communication between humans is faster than the other alternatives like keyboards or touch screens.

There are many application areas for speech recognition. The main areas can be listed as home use, office use, education portable and wearable technologies, control of vehicles, avionics, telephone services, communications, hostile environments, forensics and crime prevention, entertainment, information retrieval, biometrics surveillance, etc.

Speech recognition is closely related to other speech related technologies such as automatic speech recognition, speech synthesis, speech coding, spoken language understanding, spoken dialogue processing, spoken language generation, auditory modeling, paralinguistic speech processing (speaker verification/recognition/identification, language recognition, gender recognition, topic spotting), speech verification, time-stamping/automatic subtitling, speech to speech translation, etc.

Speech recognition is a multi-discipline spanning to acoustics, phonetics, linguistics, psychology, mathematics and statistics, computer science, electronic engineering and human sciences.

Speech recognition has been a research field since the 1950s. The advances are not satisfactory enough despite more than 50 years of research. This is mainly due to openness of speech communication to environmental effects and existence of various variabilities that are difficult to model in the speech. The speech is acquired by computers using microphones which record it as energy levels at certain frequencies. Since speech is passed through air before having recorded digitally, the recording contains environmental effects also. Speech recognition process is based only on the speech content of the recorded signal. The quality of the signal must be improved before speech recognition. Hermansky (1998) claims that indiscriminate use of accidental knowledge about human hearing in speech recognition may not be what is needed. What is needed is to find the relevant knowledge and extract it before doing any further processing towards speech recognition.

Figure 1.1 shows the speech recognition process in a simplified way. Speech recognizer contains the necessary information to recognize the speech at the point. At the input there should be a microphone and at the output there is a display that shows the recognized speech.

The remaining part of this chapter defines the speech recognition cycle by decomposing it to its basic parts. In a more general context, speech recognition can be seen as a signal modeling and classification problem. The main is to create models of speech of and use these models to classify it. The speech includes two parts which can be modeled: Acoustic signal and language.

As a modeling problem, speech recognition includes two models: Acoustic model and language model. These two models will be explained later in detail. Acoustic model is the modeling of acoustic signal and it starts with acquisition of speech by computers. Language model is the modeling of speaker's language and it will be used at the end of classification process to restrict the speech recognition to extract only acceptable results from speech signal.

The speech recognizer as shown in Figure 1.1 can be extended as on Figure 2.1 that shows, the main procedures in speech recognition are speech acquisition, preprocessing, feature extraction, recognition, recognition is sometimes called decoding. The most important parts which affect the performance of the system are acoustic model and language model. These models are obtained after a training procedure. Speech acquisition, preprocessing and feature extraction are also important for representing speech signal in recognition phase.

2.2 Speech Acquisition

Speech acquisition includes converting the acoustic signal to some computer readable digital signal codes. This process can also be called as "digital recording".

Speech signal is an analog signal which has a level (loudness), shape, and frequency. The first thing to do with speech signal is to convert it from analog domain which is continuous to digital domain which is discrete. To convert a signal from continuous time to discrete time, a process called sampling is used. The value of the signal is measured at certain intervals in time. Each measurement is referred to as a sample.

When the continuous analog signal is sampled at a frequency F, the resulting discrete signal has more frequency components than did the analog signal. To be precise, the frequency components of the analog signal are repeated at the sample rate. That is, in the discrete frequency response they are seen at their original position, and are also seen centered around +/- F, and around +/-2F, etc.

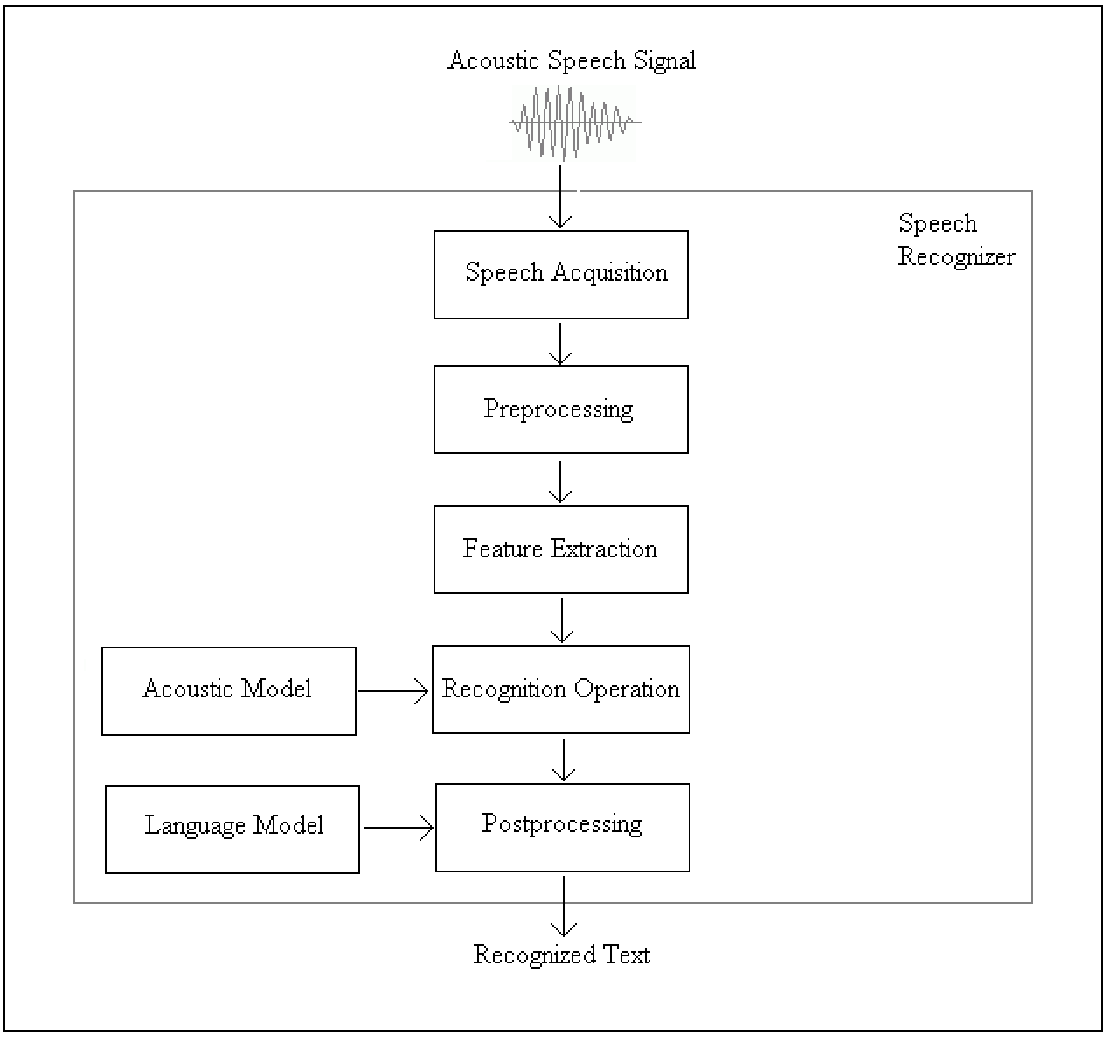


Figure 2.1 Speech recognition procedure.

If the signal contains high frequency components, we will need to sample at a higher rate to avoid losing information that is in the signal. In general, to preserve the full information in the signal, it is necessary to sample at twice the maximum frequency of the signal. This is known as the Nyquist rate.

Telephone speech is sampled at 8 kHz, which means the highest frequency represented is 4000 Hz which is greater than the maximum frequency standard for telephone in Europe (3400 Hz). A sampling frequency of 16 kHz is regarded as sufficient for speech recognition. Generally, speech signal sampling frequency is chosen 600 Hz and 16000 Hz. The frequency range that human ear hear is between 80 Hz and 8000 Hz. The extreme limits are 20 Hz and 20 kHz (Boite & Kunt, 1987).

The level of sampled speech signal is the sampling resolution. Using of more bits gives better resolution. For telephone speech, compressed 8 bits sampling resolution is used. For speech recognition, in general, 12 bits are sufficient. For higher accuracies, we need to use more bits per sample.

The speech signal can contain some redundant frequency components which are considered as noise. Some of those frequencies can be filtered. Generally, filters are used to modify the magnitude of signals as a function of frequency. Desirable signals in one range of frequencies (usually called a band) are passed essentially unchanged, while unwanted signals (noise) in another band are attenuated.

Figure 2.2 shows the structure of a speech acquisition block which can be integrated into speech recognizer in which the analog filtering part is generally integrated into a microphone. A device is used to record the speech digitally according to sampling theory. Digitalized speech can than be filtered digitally to improve the quality of speech.

The digital speech signal can have various formants. Digital representation of speech is generally called “coding”. There are three groups of coding as waveform coding, source coding and hybrid coding.

The waveform coding attempts to produce a reconstructed signal whose waveform is as close as possible to the original. The resulting representation is independent of the type signal. The most commonly used waveform coding is called “Pulse Code Modulation” (PCM). It is made up of quantizing and sampling the input waveform.

There are two variants of this coding method. Those are Differential PCM (DPCM) which quantizes the difference between two samples, and Adaptive DPCM (ADPCM) which tries to predict the signal and use a suitable quantization for different portion of that signal.

The source coding is model based. A model of the source signal is used to code the signal. This technique needs a priori knowledge about production of signal. The model parameters are estimated from the signal. Linear Predictive Coding (LPC) uses source coding method. The value of the signal at each sample time is predicted to be linear function of the past values of the quantized signal.

The hybrid coding is a combination of two other coding methods. An example of this type of coding is “Analysis by Synthesis”. The waveform is first, coded by source coding technique. Then the original waveform is reconstructed and the difference between original and coded signal is tried to be minimized.

2.3 Preprocessing and Feature Extraction

The original analogue signal which to be used by the system in both training and recognition is converted from analogue to discrete speech signal, $x(n)$. n is represented as the sample index.

The sample rate, Fs was 11025 Hz. An example of a signal in waveform sampled is given in Figure 2.2.

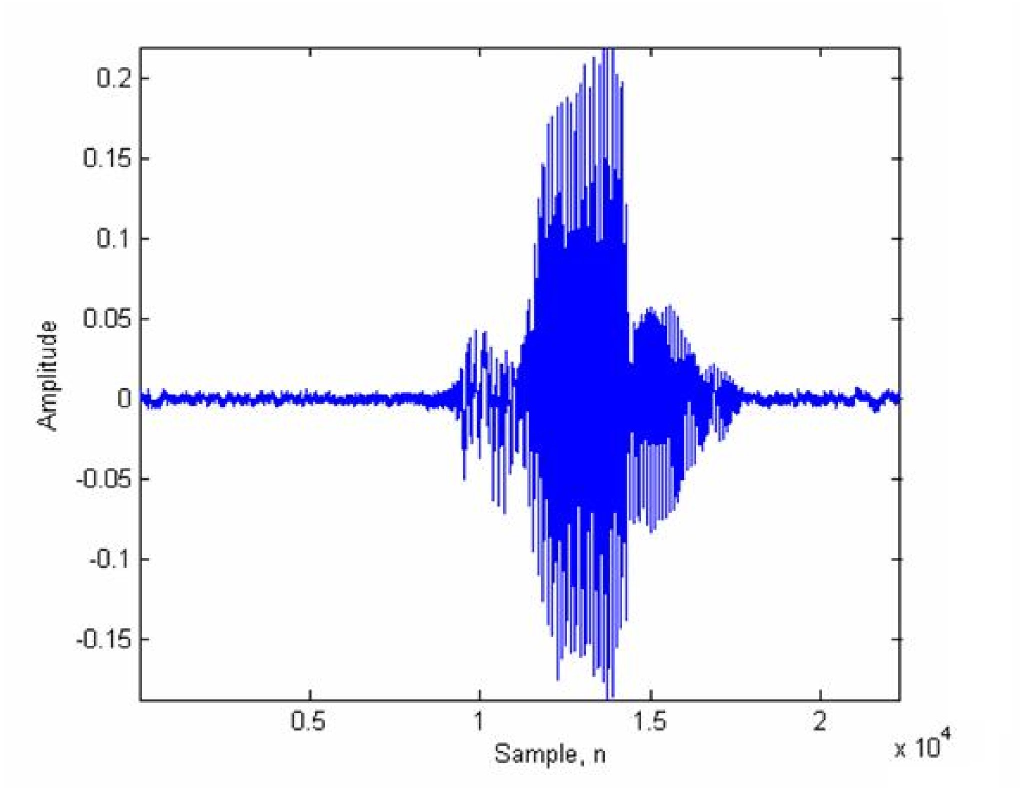


Figure 2.2 Sampled utterence signal of "fen" in waveform.

There is a need for spectrally flatten the signal. The preemphasizer, often represented by a first order high pass FIR filter is used to emphasize the higher frequency components. The transfer function of this filter in time domain is described in Eq.2.1.

$$H(z) = 1 - 0.95z^{-1} \qquad \text{(Eq.2.1)}$$

The result of the filtering is given in Figure 2.3.

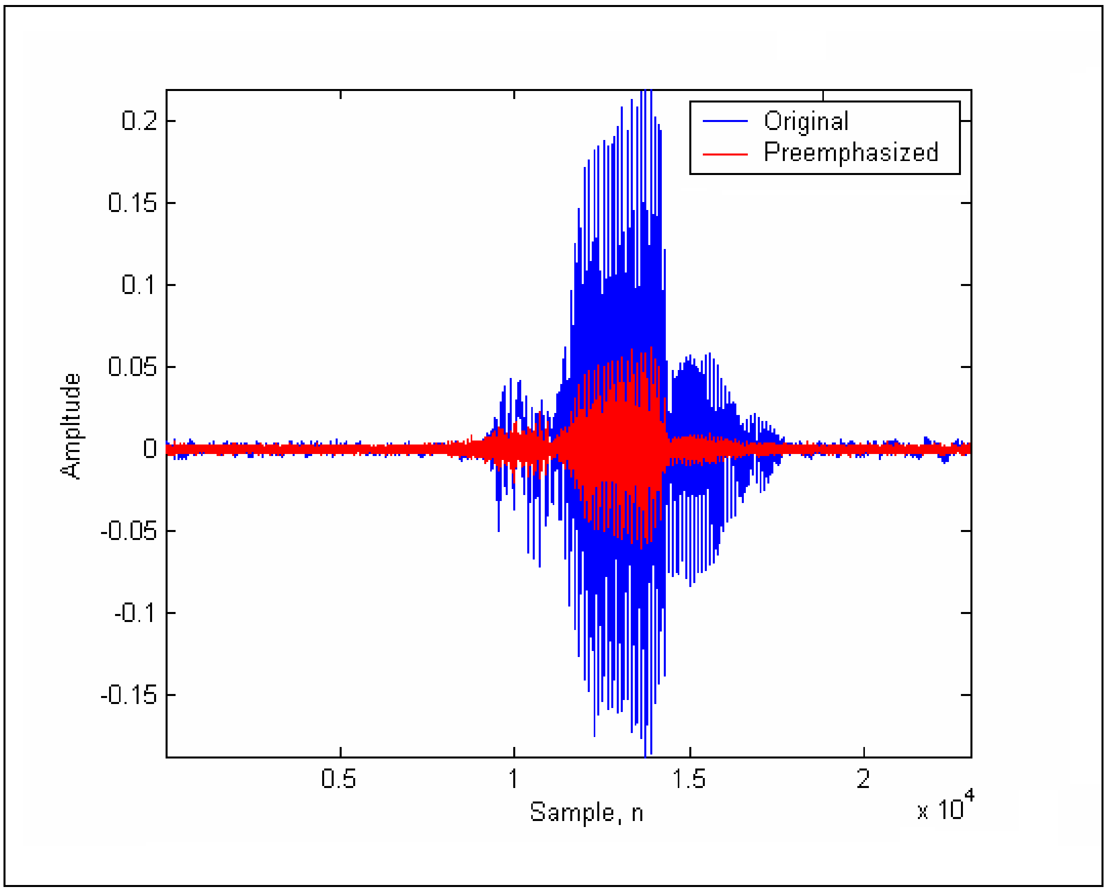


Figure 2.3 Original signal (blue color) and preemphasized signal (red color).

After detecting the end-point of the syllables from the preemphasized speech signal, frameblocking is applied to each syllable signal. Syllable end-point detection is explained in Chapter 6 in detail. The objective of frameblocking is to divide the signal into a matrix form with an appropriate time length for each frame. Due to the assumption that a signal within a frame of 20 ms is stationary and a sampling rate at 16000 Hz will give the result of a frame of 320 samples.

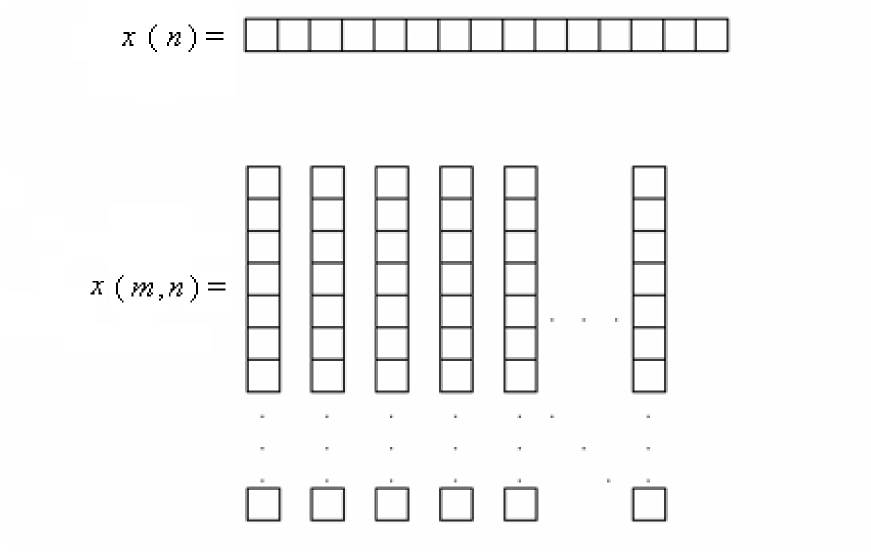


Figure 2.4 Frameblocking.

As shown in Figure 2.4 the speech signal, $x(n)$ is divided into the matrix form, $x(m, n)$. There are m frames and, each frame consists of n samples.

After the frameblocking is done, a Hamming window, which is graphically demonstrated in Figure 2.5, is applied to each frame. This window is to reduce the signal discontinuity at the ends of each block.

The equation which defines a Hamming window is shown in Eq.2.2.

$$w(k) = 0.54 - 0.46\cos(\frac{2\pi k}{K-1}) \qquad \text{(Eq.2.2)}$$

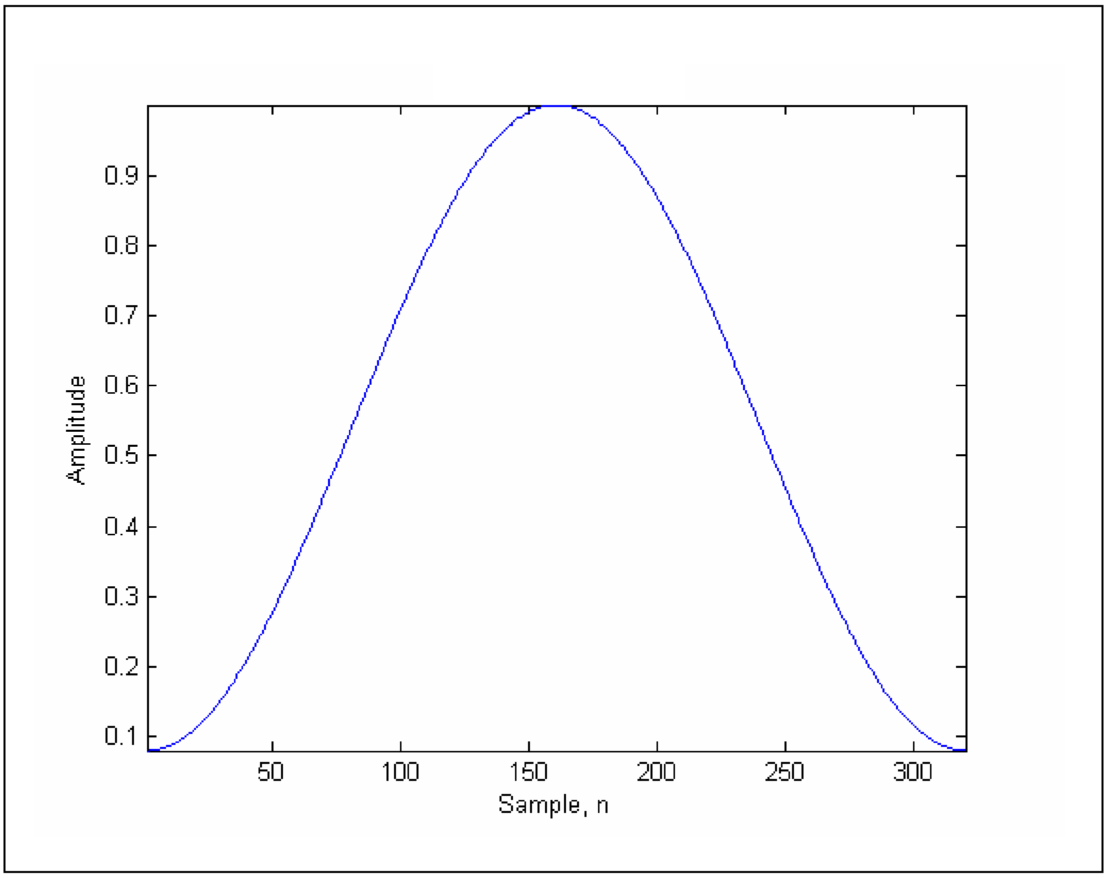


Figure 2.5 Hamming window.

Figure 2.6 shows only one frame's signals which are results of frame blocking. In Figure 2.7, the frame windowed by Hamming window is displayed. The result gives a reduction of the discontinuity at the ends of the frame.

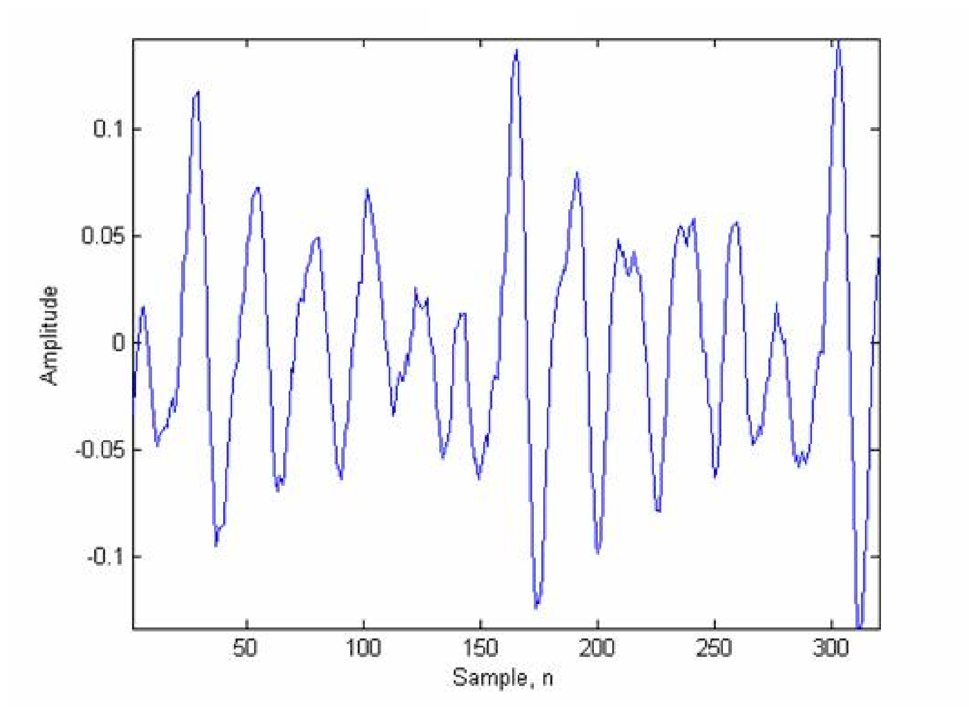


Figure 2.6 Signals on a frame before windowing.

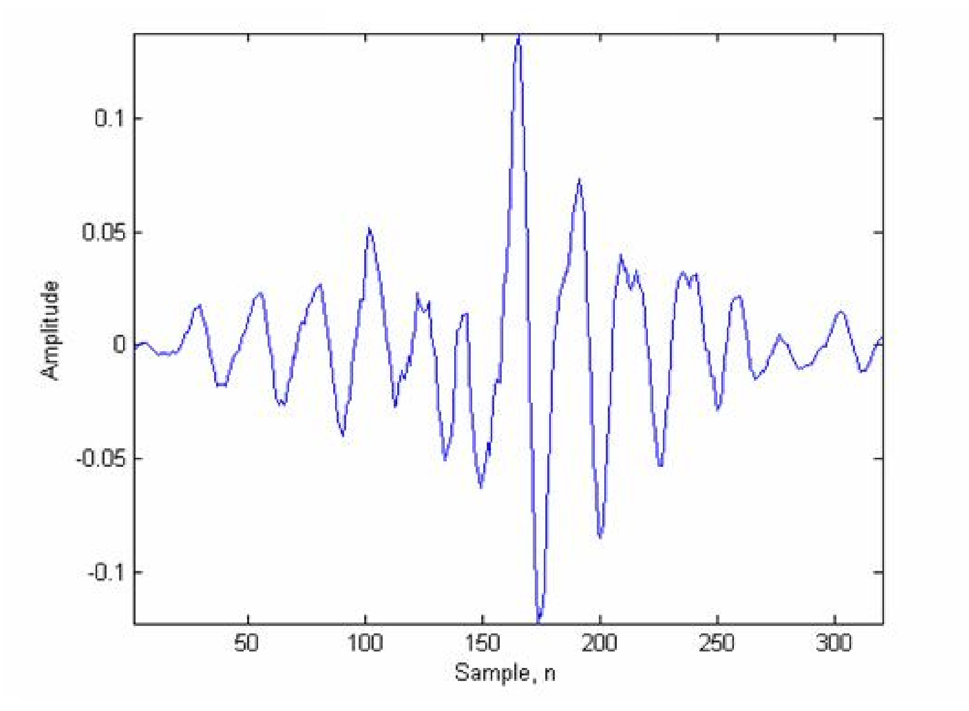


Figure 2.7 Signals on a frame after windowing.

2.4 Recognition Operation

Speech recognition includes two pattern classification steps. The first one is acoustic processing which results a sequence of syllable speech units. The output of first step is then used for language processing which guaranties a valid speech output within the rules of current language. As a pattern classification problem, speech recognition must be mathematically formulated and decomposed into simpler subproblems.

Let $Y = Y_1, Y_2, ..., Y_k$ be a sequence of feature vectors obtained from the speech signal. The feature vectors Y_i are generated sequentially by increasing values of *i* and *k* is the number of feature vector in the sequence.

Let $S = s_1, s_2, ..., s_n$ be the syllable content of the speech signal. *n* is the number of syllables in the speech signal.

$P(S \mid Y)$ is the probability that the syllable *S* was spoken, given the feature vector sequence, which is called "observation". After defining these elements, the speech recognition can be defined as a decision making process searching for the most probable syllable sequence $\hat{S}$ as Eq.2.3 which consists of searching for the most likely syllable sequence *S* conditioned on observation sequence *Y*. The probability $P(S \mid Y)$ can not be observed directly because of randomness of feature vector space. We need to rewrite this probability.

$$\hat{S} = \arg\max_{S} P(S \mid Y) \qquad \text{(Eq.2.3)}$$

The right-hand side probability of Eq.2.3 can be rewritten according to Bayes' formula of probability theory as shown in Eq.2.4.

$$P(S \mid Y) = \frac{P(S)P(Y \mid S)}{P(Y)} \qquad \text{(Eq.2.4)}$$

$P(S)$ is the probability that the syllable string S will be spoken by the speaker, $P(Y \mid S)$ is the likelihood, and $P(Y)$ is the average probability that Y will be observed. $P(Y)$ in Eq.2.4 is known also as evidence, and it is generally omitted in speech recognition since this probability is same for all acoustic signal observations. The new version of maximization Eq.2.3 can be rewritten as Eq.2.5 after omitting the evidence of observing acoustic observation Y in Bayes' formula.

$$\hat{S} = \arg\max_{S} P(S)\, P(Y \mid S) \qquad \text{(Eq.2.5)}$$

Eq.2.5 is the base for classification in speech recognition. By writing the equation in this form we have the apportunity of computing the probabilities $P(S)$ and $P(Y \mid S)$ by training some models. $P(S)$ can be obtained by training a model for the language and is independent of acoustic information. Language modeling is based on assigning a probability to each syllable occurrence within a context and the model can be trained on a large text containing virtually all occurrences of syllable sequences in the language.

The second probability in Eq.2.5 can be obtained by training a model for the acoustic realizations of syllables. This modeling is called acoustic modeling and can be obtained by training a model from a large acoustic database which contains virtually all realizations of the syllables in the language.

2.5 Acoustic Modeling

Acoustic modeling is the process of generating models for each class in speech recognition. The class can be a word, a syllable, a semi-syllable, or a phoneme. There are many kinds of acoustic models and modeling techniques. The simplest acoustic model can be the acoustic realization of each words in the vocabulary of speech recognizer.

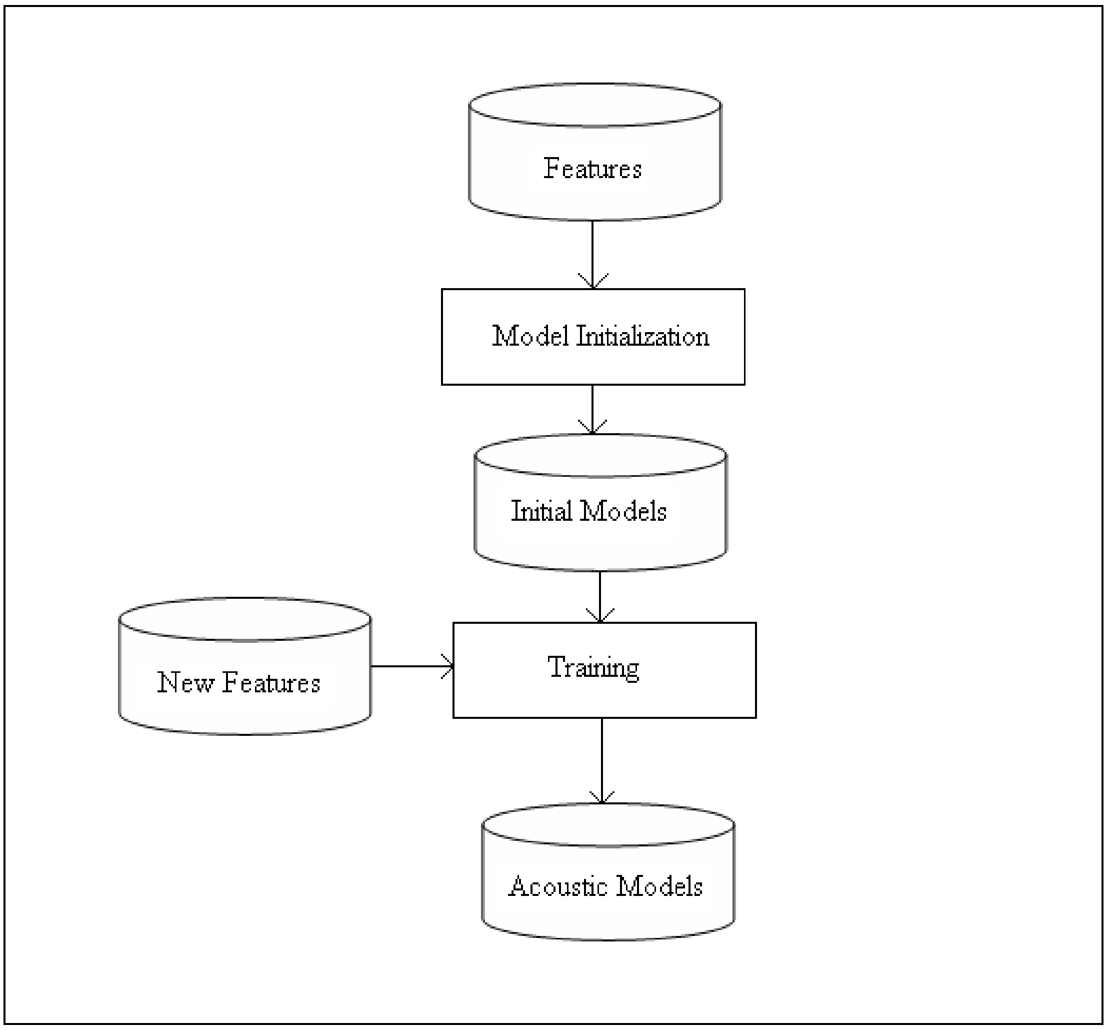


Figure 2.8 Constructing acoustic models.

Figure 2.8 gives the acoustic modeling process. Acoustic modeling process is not a part of speech recognition. It provides the acoustic models which are used in speech recognition for classification.

The flowchart in Figure 2.8 is not standard for all acoustic modeling techniques but it includes the common steps in acoustic modeling.

The first step is “initialization” of models. At this step pre-segmented feature vectors are assigned to classes and a model for each class is created. In “training” step initial models are used for classification of new feature vectors which are not segmented. After segmentation new class boundaries for models are determined in an iterative approach. Some generalization algorithms are applied to have a better modeling of unseen data. The output of this process is acoustic models for each class. The acoustic models are used by the recognizer to determine the probability $P(Y \mid S)$ of Eq.2.5.

An important aspect of classification process is distance measure which is common in training of acoustic models. All acoustic modeling techniques are based on some distance measure to find the closeness of a new feature vector to a model. Distance measure is used for comparing feature vectors to some stored templates for classification purposes. The stored templates can be updated with new data in an iterative approach.

Let *C* be the available classes by a template feature vector as shown in Eq.2.6.

$$C = c_1, c_2, \ldots, c_N \qquad \text{(Eq.2.6)}$$

In Eq.2.6, *N* is the number of classes. The simplest way to classify a feature vector y_i is to compare it to class templates and find the closest template.

$$T = \arg \min_{t=1}^{N} d(y_i, c_t) \qquad \text{(Eq.2.7)}$$

In Eq.2.7, *T* is the class for feature vector y_i and $d(y_i, c_t)$ is the distance function between the feature vector y_i and the class c_t.

The most commonly used distance function is Euclidean distance function which is defined as Eq.2.8.

$$d(m,n) = \sqrt{\sum_i (m_i - n_i)^2} \qquad \text{(Eq2.8)}$$

In Eq.2.8, *m* and *n* are feature vectors.

The main acoustic modeling techniques are DTW, ANN, HMM and SVM which are explained in Chapter 3 in detail.

2.6 Language Modeling

Language modeling is the process of extracting important properties of a natural language by analyzing statistically a corpus of language. The goal is to assign probabilities to strings of words in the language. These properties are then used to rank the word sequences candidates from recognition results of acoustic model. Probability that the word sequence *S* were spoken given the feature vector *X*, $P(S \mid X)$, can be rewritten from Bayes' formula as shown in Eq.2.9.

$$P(S \mid X) = \frac{P(S)P(X \mid S)}{P(X)} \qquad \text{(Eq.2.9)}$$

$P(S)$ is the probability that the syllable string *S* will be spoken by the speaker. $P(X \mid S)$ is the probability that when the speaker says *S* the speech signal represented by *X* will be observed, and $P(X)$ is the probability of observing *X*.

In Eq.2.9, the probability $P(X \mid S)$ is the acoustic model probability. $P(X)$ is omitted because of assumption about randomness of speech. The last unknown probability is the probability $P(S)$, the language model probability.

By using a language model for speech recognition, the number of acceptable syllable sequences is limited. This limitation leads to an increase in the accuracy of the speech recognizer since some erroneous syllable sequences will be replaced by nearest approximations which are mostly the correct syllable sequences. Language models are useful for large vocabulary continuous speech recognition tasks. For small vocabulary isolated.

Language models are used to assign probabilities to syllable sequences. Models are trained with a large text corpus from the language to be modeled. Language modeling is based on estimation of probability that word sequence *S* can be exist in the language. For a syllable sequence S=s_1, s_2, s_3, ..., s_N, probability $P(S)$ is defined as Eq.2.10.

$$P(s) = \prod_{i=1}^{N} P(s_i \mid s_1, s_2, ..., s_{i-1}) \qquad \text{(Eq.2.10)}$$

n is the number of syllables in the sequence. $P(s_i \mid s_1, s_2, ..., s_{i-1})$ is the probability that s_i is observed after syllable sequence $\{s_1, s_2, ..., s_{i-1}\}$ which is called history.

Statistical language modeling is based on the formulation as Eq.2.10. The main task in language modeling is to provide good estimation of $P(s_i \mid s_1, s_2, ..., s_{i-1})$, the probability of i^{th} syllable given the history $\{s_1, s_2, ..., s_{i-1}\}$ (Jelinek, 1998). There are two methods frequently used for language modeling: *n*-gram language modeling and part-of-speech (POS) based language models. The details of these two types of modeling techniques will be explained in the following subsections. Both of them are based on statistics obtained from a training corpus. *n*-gram language models are based directly on the occurrences of syllables in the history list where POS models use linguistic information instead of syllables.

Language modeling is based on counting the occurrences of syllable sequences. When long histories are used some syllable sequences may not be appear in the training text. This results in poor modeling of acceptable syllable sequences and is called as data sparseness problem.

Sparseness problem in language modeling is solved by applying smoothing techniques to language models. Smoothing techniques are used for better estimating probabilities when there is insufficient examples of some syllable sequences to estimate accurate syllable sequence probabilities directly from data. Since smoothing techniques are applied to *n*-gram language modeling, some of smoothing techniques will be presented in the following subsections.

When the number of possible syllable sequences that can be accepted by the speech recognizer is known and limited, then it is possible to create some finite state grammars which limit the output of the recognizer. The finite state grammars used in this case are also called language models. This type of language models are task oriented and can be created in a deterministic way.

2.6.1 n-gram Language Models

n-gram language models are the most widely used language modeling methods. The *n* is generally selected as 1 (monogram), 2 (bigram) or 3 (trigram) in most *n*-gram language models.

P(*S*) is the probability of observing syllable sequence *S* and can be decomposed as Eq.2.11.

$$\begin{aligned} P(S) &= P(s_1, s_2, ..., s_n) \\ &= P(s_1)P(s_2 \mid s_1)P(s_3 \mid s_1, s_2)...P(s_n \mid s_1, s_2, s_3, ..., s_{n-1}) \\ &= \prod_{i=1}^{N} P(s_i \mid s_1, s_2, s_3, ..., s_{i-1}) \end{aligned} \qquad \text{(Eq.2.11)}$$

$P(s_i \mid s_1, s_2, ..., s_{i-1})$ is the probability that s_i will be observed after history s_1, s_2, …, s_{i-1}. This formulation is the general form for *n*-gram language models. For monogram language model the probabilities $P(s_i)$, for bigram $P(s_i \mid s_{i-1})$ and for trigram $P(s_i \mid s_{i-1}, s_{i-2})$ are computed.

The size of history depends on the selection of *n* for an *n*-gram language. There is no history when for monogram language models, the history has only one syllable for bigram and two syllables for trigram language models.

The probability *P*(*S*) is computed by counting the frequencies of syllable sequence *S* and the history. For example, trigram probabilities are computed as Eq.2.12.

$$P(s_i \mid s_{i-2}, s_{i-1}) = \frac{C(s_{i-2}, s_{i-1}, s_i)}{C(s_{i-2}, s_{i-1})} \qquad \text{(Eq.2.12)}$$

$C(s_{i-2}, s_{i-1}, s_i)$ is the number of occurrences of syllable sequence s_{i-2}, s_{i-1}, s_i and $C(s_{i-2}, s_{i-1})$ is the number of occurrences of history s_{i-2}, s_{i-1}.

In order to have a good estimate of language model probabilities we need a large text corpus including virtually all occurrences of all syllable sequences. For trigrams a corpus of several millions of syllables can be sufficient but for higher values of *n* the number of syllables should be very high.

2.6.2 Perplexity

The efficiency of *n*-gram language model can be simply evaluated by using it in a speech recognition task. Alternatively it is possible to measure the efficiency of a language model by its perplexity. Perplexity is a statistically weighted syllable branching measure on a test set. If the language model perplexity is higher, the speech recognizer needs to consider more branches which mean there will be a decrease on its performance.

Computation of perplexity does not involve speech recognition. It is defined as the derivative of cross-entropy (Huang, Acero & Hon, 2001). The perplexity based on cross-entropy is defined as Eq.2.13.

$$PP(S) = 2^{H(S)} \qquad \text{(Eq.2.13)}$$

$H(S)$ is the cross-entropy of the syllable sequence S and is defined as Eq.2.14.

$$H(S) = -\frac{1}{N}\log_2 P(S) \qquad \text{(Eq.2.14)}$$

N is the length of syllable sequence and $P(S)$ is the probability of the syllable sequence from language model. It must be noted that S is a sufficiently long syllable sequence which helps to find a good estimate of perplexity.

Perplexity can be measured for the training set and the test set (Huang, Acero & Hon, 2001). When it is measured for training set it provides a measure of how the language model fits the training data, for the test set it gives a measure of the generalization capacity of language model. Perplexity is seen as a measure of performance since it correlates with better recognition results. Higher perplexity means there will be more branches to consider statistically for a recognition task which leads to lower recognition accuracies.

2.6.3 Smoothing

Another important issue in *n*-gram language modeling is smoothing. Smoothing is defined as adjusting the maximum likelihood probabilities, obtained by counting to model syllable sequences, to produce more accurate probability distributions. This is necessary since data sparseness problem in training data due to high number of available syllable sequence may result in assigning low probabilities or zeroes to certain syllable sequences that will probably seen in test data. The purpose of smoothing is to make the probability distributions more uniform which means assigning higher probabilities to syllable sequences with low probabilities obtained by counting, and assigning low probabilities to syllable sequences with too high probabilities. This gives better generalization capability to the language model.

A good smoothing example is to consider each bigram is occurred one more time than it occurred in the training set. It can be done as Eq.2.15 by modifying Eq.2.12. By doing such a simple smoothing we avoided zero probabilities which could be harmful to the speech recognizer since it can reject a correct syllable sequence that could not in training set of language model but had a higher probability from acoustic model.

$$P(s_i \mid s_{i-1}) = \frac{1 + C(s_{i-1}, s_i)}{\sum_{s_i}(1 + C(s_{i-1}, s_i))} \qquad \text{(Eq.2.15)}$$

There are several smoothing techniques that can be used for language models. For different smoothing techniques, Huang et al. (2001) is a good reference. We will consider only the back-off smoothing (Katz back-off model) technique which is commonly used.

Katz back-off smoothing is based on Good-Turing estimates which partition *n*-gram into groups depending on their frequency of appearance in the training set. In the approach the frequency, r, of an *n*-gram, *n* is replaced by r_* which is defined as Eq.2.16.

$$r_* = (r+1)\frac{n_{r+1}}{n_r} \qquad \text{(Eq.2.16)}$$

n_r is the number of *n*-grams that occurs exactly *r* times and n_{r+1} is the number of *n*-grams that occurs exactly *n*+1 times. The probability of an *n*-gram, *a*, is then defined as Eq.2.17.

$$P(a) = \frac{r_*}{N} \qquad \text{(Eq.2.17)}$$

N is the number of all counts in the distribution. In Katz smoothing, the n-grams are partitioned into three class according to their frequencies in the training set. For partitioning a constant count number, k, is used. This is partitioned number generally selected between 5 and 8. If r is the count of an n-gram:

- Large counts are considered as reliable and there is no smoothing; $r > k$.
- The counts between zero and k are smoothed with Good-Turing estimates; $0 < r \leq k$. This smoothing is a discounting process which use a ration based on Good-Turing estimate to reduce the lower counts.
- The zero counts are smoothed according to some function, α, which tries to equalize the discounting of nonzero counts with increasing zero counts by a certain amount.

For bigram language model, the Katz smoothing can be summarized as Eq.2.18, Eq.2.19 and Eq.2.20 (Huang, Acero & Hon, 2001; Katz, 1987).

$$P_{Katz}(s_i \mid s_{i-1}) = \begin{cases} C(s_{i-1}, s_i)/C(s_{i-1}) & \text{if } r > k \\ d_r\, C(s_{i-1}, s_i)/C(s_{i-1}) & \text{if } k \geq r > 0 \\ \alpha(s_i)P(s_i) & \text{if r} = 0 \end{cases} \qquad \text{(Eq.2.18)}$$

where

$$d_r = \frac{\dfrac{r^*}{r} - \dfrac{(k+1)n_{k+1}}{n_1}}{1 - \dfrac{(k+1)n_{k+1}}{n_1}} \qquad \text{(Eq.2.19)}$$

and

$$\alpha(s_{i-1}) = \frac{1 - \sum_{s_i;\, r>0} P_{Katz}(s_i \mid s_{i-1})}{1 - \sum_{s_i;\, r>0} P_{Katz}(s_i)} \qquad \text{(Eq.2.20)}$$

It can be seen from Eq.2.20 that the probability of zero count bigrams is increased by weighting monogram probabilities with α.

There are several disadvantage of *n*-gram language models:

- They are unable to incorporate long-distance syllable order constraints since the length of history is generally small and the exact order is considered.
- It is not possible to integrate new syllables or alternative domains into language models.
- The meaning can not be modeled by *n*-gram language models.

Despite these disadvantages, *n*-gram language models gives good results when used in speech recognition tasks because they are based on a large corpus with helps to model the approximate syllable orders that exist in the language. Many languages have a strong tendency toward standard syllable order.

Some of the disadvantage of *n*-gram language models can be avoided by using clustering techniques. Clustering can be made manually or automatically on training set. Clustering can improve the efficiency of language model by creating more flexible models. The next subsection gives details of a clustering technique, part-of-speech (POS) tagging.

CHAPTER THREE
SPEECH RECOGNITION FEATURES AND METHODS

3.1 Speech Feature Extraction

The main objective of feature extraction is to detect specific characteristics from the speech signal that are unique to each Turkish syllable which will be used to differentiate Turkish words. We have mentioned the speech features as linear predictive coding, parcor, cepstrum, rasta and mel frequency cepstral coefficients in the following subsections.

3.1.1 Linear Predictive Coding Coefficients

It is desirable to compress a speech signal for efficient transmission or storage in variety applications. For example, to accommodate many speech signals in a given bandwidth of a cellular phone system, each digitized speech signal is compressed before transmission. In the case of a digital answering machine, to save a memory space, a message is digitized and compressed. For medium or low bit-rate speech coders, linear predictive coding (lpc) is most widely used (Ayuso & Soler, 1993; Becchetti & Ricotti, 1999; Mengüşoğlu, 1999; Meral, 1996). Redundancy in a speech signal is removed by passing the signal through a speech analysis filter. The output of the filter, which is termed the residual error signal, has less redundancy than original speech signal and can be quantized by smaller number of bits than the original speech. The residual error signal along with the filter coefficients are transmitted to the receiver. At the receiver, the speech is reconstructed by passing the residual error signal through the synthesis filter. To model a human speech production system, all-pole model (also known as the linear prediction model) is used.

An all-pole system (or the linear prediction system) is used to model a vocal tract as shown in Figure 3.1.

An efficient algorithm known as the Levinson-Durbin algorithm is used to estimate the linear prediction coefficients from a given speech waveform. Assume that the present sample of the speech is predicted by the past M samples of the speech as shown in Eq.3.1.

$$\tilde{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \ldots + a_M x(n-M) = \sum_{i=1}^{M} a_i x(n-i) \qquad \text{(Eq.3.1)}$$

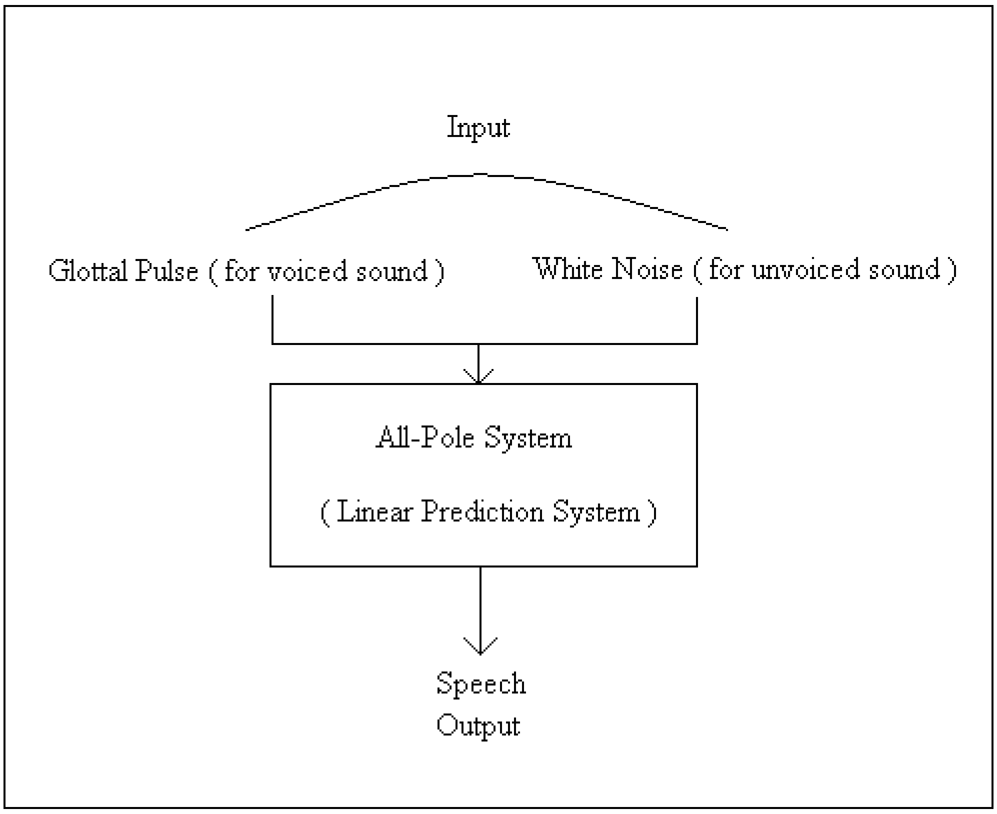

Input
Glottal Pulse (for voiced sound)
White Noise (for unvoiced sound)
All-Pole System
(Linear Prediction System)
Speech
Output

Figure 3.1 Simplified model of the speech production.

$\tilde{x}(n)$ is the prediction of $x(n)$, $x(n-i)$ is the i-th step previous sample, and $\{a_i\}$ are called the linear prediction coefficients. The error between the actual sample and the predicted one can be expressed as Eq.3.2.

$$\varepsilon(n) = x(n) - \tilde{x}(n) = x(n) - \sum_{i=1}^{M} a_i x(n-i) \qquad \text{(Eq.3.2)}$$

The sum of the squared error to be minimized is expressed as Eq.3.3.

$$E = \sum_n \varepsilon^2(n) = \sum_n \left(x(n) - \sum_{i=1}^{M} a_i x(n-i) \right)^2 \qquad \text{(Eq.3.3)}$$

We would like to minimize the sum of the squared error. By setting to zero the derivative of E with respect to a_i (using the chain rule), one obtains Eq.3.4.

$$2\sum_n x(n-k)\left(x(n) - \sum_{i=1}^{M} a_i x(n-i) \right) = 0 \quad \text{for } k = 1, 2, 3, ..., M \qquad \text{(Eq.3.4)}$$

Eq.3.4 results in M unknowns in M equations such that

$$\begin{aligned} &a_1 \sum_n x(n-k)x(n-1) + a_2 \sum_n x(n-k)x(n-2) + ... \\ &+ a_M \sum_n x(n-k)x(n-M) = \sum_n x(n-k)x(n) \quad \text{for } k = 1, 2, 3, ..., M \end{aligned} \qquad \text{(Eq.3.5)}$$

Let us assume that a speech signal is divided into many segments (or frames) with N samples. If the length of each segment (or frame) is short enough, the speech signal in the segment may be stationary. In other words, the vocal tract model is fixed over the time period of one segment. The length of each segment is usually chosen as 20-30 ms. If a speech signal is sampled at the rate of 8000 samples/second and the length of each segment is 20 ms, then the number of samples in each segment will be 160. If the length is 30 ms, then the number of samples is going to be 240.

If there are N samples in the sequence indexed from 0 to N−1 such that $\{x(n)\} = \{x(0), x(1), x(2), ..., x(N-2), x(N-1)\}$, Eq.3.5 can be approximately expressed in terms of matrix equation.

$$\begin{bmatrix} r(0) & r(1) & . . . & r(M-2) & r(M-1) \\ r(1) & r(0) & . . . & r(M-3) & r(M-2) \\ . & . & . . . & . & . \\ . & . & . . . & . & . \\ . & . & . . . & . & . \\ r(M-2) & r(M-3) & . . . & r(0) & r(1) \\ r(M-1) & r(M-2) & . . . & r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_{M-1} \\ a_M \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ . \\ . \\ . \\ r(M-1) \\ r(M) \end{bmatrix} \qquad \text{(Eq.3.6)}$$

where $r(k) = \sum_{n=0}^{N-1-k} x(n)x(n+k)$ (Eq.3.7)

This is called the autocorrelation method. To solve the matrix equation as Eq.3.6, Gauss elimination, iteration method, or QR decomposition can be used. In any case, an order of M^3 multiplications is required to solve the equation. However, because of the special characteristics of the matrix, the number of multiplications can be reduced to the order of M^2 with the Levinson-Durbin algorithm that will be introduced in the next section.

Once the linear prediction coefficients $\{a_i\}$ are computed, Eq.3.2 can be used to compute the error sequence $\varepsilon(n)$. The implementation of Eq.3.2, where $x(n)$ is the input and $\varepsilon(n)$ is the output, is called the analysis filter and shown in Figure 3.2.

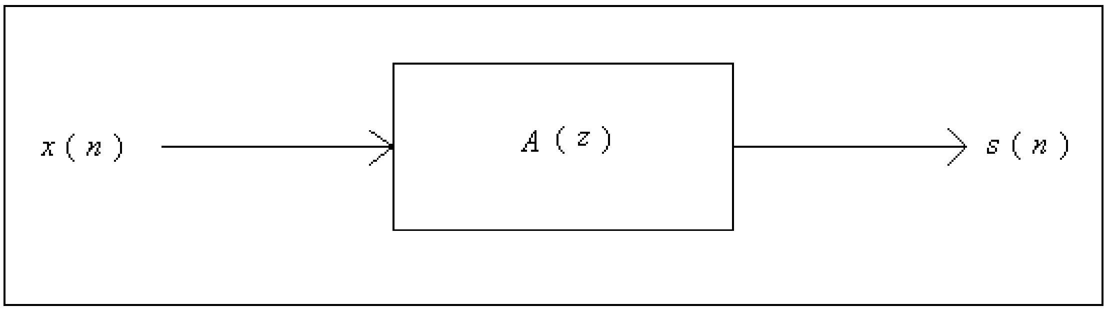


Figure 3.2 Speech analysis filter.

The transfer function is given by Eq.3.8.

$$A(z) = 1 - \sum_{i=1}^{M} a_i z^{-i} \qquad \text{(Eq.3.8)}$$

Because residual error, $\varepsilon(n)$, has less standard deviation than speech itself, smaller number of bits is needed to quantize the residual error sequence.

Eq.3.2 can be rewritten as the difference equation of a digital filter whose input is $\varepsilon(n)$ and output is $x(n)$ as Eq.3.9.

$$x(n) = \sum_{i=1}^{M} a_i x(n-i) + \varepsilon(n) \qquad \text{(Eq.3.9)}$$

The implementation of Eq.3.9 is called the synthesis filter and is shown in Figure 3.3.

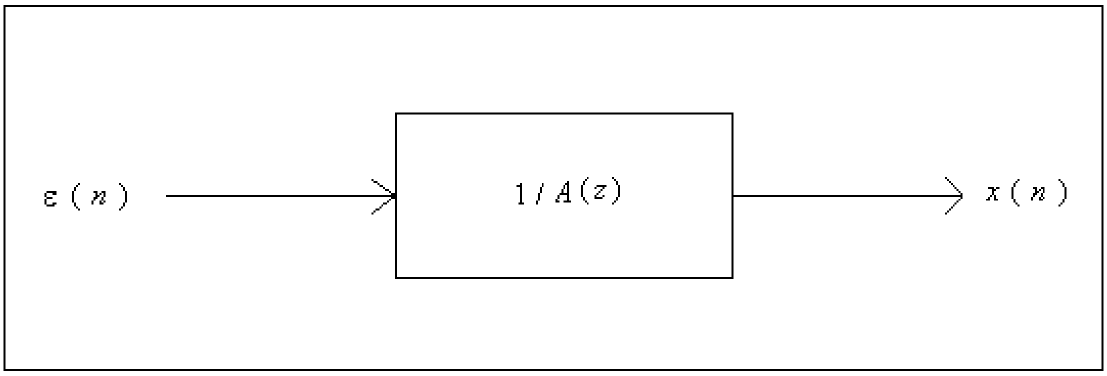


Figure 3.3 Speech synthesis filter.

If both the linear prediction coefficients and the residual error sequence are available, the speech signal can be reconstructed using the synthesis filter. In practical speech coders, linear prediction coefficients and residual error samples need to be compressed before transmission. Instead of quantizing the residual error, sample by sample, several important parameters such as pitch period, code for a particular excitation, etc are transmitted. At the receiver, the residual error is reconstructed from the parameters.

3.1.1.1 Levinson-Durbin Recursive Method

In this section, the Levinson-Durbin method is introduced to solve Eq.3.6 recursively. The Levinson-Durbin method is efficient, as it needs only the order of M^2 multiplications to compute the linear prediction coefficients.

The sum of squared errors of the *M*-th order prediction (or simply the *M*-th order prediction error) in Eq.3.3 can be rewritten as Eq.3.10.

$$E_M = \sum_n x(n)\varepsilon(n) - \sum_n \left(\sum_{i=1}^{M} a_i x(n-i) \right) \varepsilon(n) \qquad \text{(Eq.3.10)}$$

Subscript *M* of E_M denotes the order of prediction. Eq.3.4 can be rewritten as Eq.3.11.

$$\sum_n x(n-i)\varepsilon(n) = 0 \quad \text{for } i = 1, 2, 3, ..., M \qquad \text{(Eq.3.11)}$$

Because of Eq.3.11, the second summation of Eq.3.10 is zero. Thus, the final expression of the prediction error becomes as Eq.3.12.

$$\begin{aligned} E_M &= \sum_n x(n) \left(x(n) - \sum_{i=1}^{M} a_i x(n-i \right) \\ &= r(0) - a_1 r(1) - ... - a_{M-1} r(M-1) - a_M r(M) = r(0) - \sum_{i=1}^{M} a_i r(i) \end{aligned} \qquad \text{(Eq.3.12)}$$

We now want to develop a recursive method to solve Eq.3.6. Let us start from the order *m*=0 and increase it until the desired order reaches.

m=0: When *m*=0 (i.e., when no prediction is made), the error is expressed as Eq.3.13 from Eq.3.12.

$$E_0 = r(0) \qquad \text{(Eq.3.13)}$$

m=1: When *m*=1, the error is expressed as Eq.3.14.

$$E_1 = r(0) - a_{11} r(1) \qquad \text{(Eq.3.14)}$$

The second subscript 1 of a_{11} indicates that the prediction order *m* in this case is 1. The solution to Eq.3.6 is as Eq.3.14.

$$a_{11} = r(1)/r(0) = \kappa_1 \qquad \text{(Eq.3.15)}$$

κ_1 is termed the reflection coefficient. Note that magnitude of κ_1 is less than 1. ($|\kappa_1|<1$) as $|r(1)|$ is less than $r(0)$. Now the prediction error for *m*=1 becomes as Eq.3.16.

$$E_1 = r(0) - \kappa_1 r(1) = r(0)\left[1 - \kappa_1{}^2\right] = E_0\left[1 - \kappa_1{}^2\right] \qquad \text{(Eq.3.16)}$$

One can easily show that the prediction error E_1 is smaller than E_0.

m=2: When *m*=2, Eq.3.12 and Eq.3.6 can be combined in a single matrix equation.

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0 & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -a_{12} \\ -a_{22} \end{bmatrix} = \begin{bmatrix} E_2 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(Eq.3.17)}$$

Assume that the solution can be found recursively as shown below.

$$\begin{bmatrix} 1 \\ -a_{12} \\ -a_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ -a_{11} \\ 0 \end{bmatrix} - \kappa_2 \begin{bmatrix} 0 \\ -a_{11} \\ 1 \end{bmatrix} \qquad \text{(Eq.3.18)}$$

κ_1 is the reflection coefficient. The subscript 2 of a_{12} and a_{22} indicates that these are the second order linear prediction coefficients. When the prediction order m=1, Eq.3.19 can be easily shown.

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -a_{11} \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \end{bmatrix} \qquad \text{(Eq.3.19)}$$

Now Eq.3.17 becomes as Eq.3.20.

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \left\{ \begin{bmatrix} 1 \\ -a_{11} \\ 0 \end{bmatrix} - \kappa_2 \begin{bmatrix} 0 \\ -a_{11} \\ 1 \end{bmatrix} \right\} = \begin{bmatrix} E_1 \\ 0 \\ q_2 \end{bmatrix} - \kappa_2 \begin{bmatrix} q_2 \\ 0 \\ E_1 \end{bmatrix} = \begin{bmatrix} E_2 \\ 0 \\ 0 \end{bmatrix} \qquad \text{(Eq.3.20)}$$

$$q_2 = r(2) - a_{11} r(1) \qquad \text{(Eq.3.21)}$$

Because $q_2 - \kappa_2 E_1 = 0$ from Eq.3.20, the reflection coefficient becomes as Eq.3.22.

$$\kappa_2 = q_2 / E_1 \qquad \text{(Eq.3.22)}$$

The new prediction error for M=2 becomes as Eq.3.23.

$$E_2 = E_1 - \kappa_2 q_2 = E\left[1 - {\kappa_2}^2\right] \qquad \text{(Eq.3.23)}$$

The linear prediction coefficients as Eq.3.24 can be obtained using Eq.3.18.

$$\begin{aligned} a_{11} &= a_{11} - \kappa_2 a_{11} \\ a_{22} &= \kappa_2 \end{aligned} \qquad \text{(Eq.3.24)}$$

m=3: When m=3, one can show Eq.3.25.

$$q_3 = r(3) - a_{12}r(2) - a_{22}r(1)$$
$$\kappa_3 = q_3 / E_2 \qquad \text{(Eq.3.25)}$$
$$E_3 = E_2\left[1 - \kappa_3{}^2\right]$$

with the assumption as Eq.3.26.

$$\begin{bmatrix} 1 \\ -a_{13} \\ -a_{23} \\ -a_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ -a_{12} \\ -a_{22} \\ 0 \end{bmatrix} - \kappa_3 \begin{bmatrix} 0 \\ -a_{22} \\ -a_{12} \\ 1 \end{bmatrix} \qquad \text{(Eq.3.26)}$$

Now the linear coefficients as Eq.3.27 can be obtained from Eq.3.26.

$$a_{13} = a_{i2} - \kappa_3 a_{(3-i)2} \quad \text{for } i = 1, 2.$$
$$\mathrm{a}_{33} = \kappa_3 \qquad \text{(Eq.3.27)}$$

3.1.1.1.1 Recursive Algorithm. Now the recursive solution method for any prediction order M is described below.

Initial values:

$$E_0 = r(0)$$

$$a_{11} = \kappa_1 = r(1) / E_0$$

$$E_1 = E_0(1 - \kappa_1{}^2)$$

with $m \geq 2$, the following recursion is performed.

(i) $q_m = r_m - \sum_{i=1}^{m-1} a_{i(m-1)} r(m-i)$

(ii) $\kappa_m = \dfrac{q_m}{E_{(m-1)}}$

(iii) $a_{mm} = \kappa_m$

(iv) $a_{im} = a_{i(m-1)} - \kappa_m a_{(m-i)(m-1)}$ for $i = 1, ..., m - 1$

(v) $E_m = E_{m-1}\left[1 - \kappa_m{}^2\right]$

(vi) If $m<M$, then increase m to $m+1$ and go to (i). If $m = M$, then stop.

In the recursion, there are $2m+1$ multiplications are involved for each m. Thus, the total number of multiplications to estimate prediction coefficients for the prediction order, M, becomes as Eq.3.28.

$$\text{\# multiplication} = \sum_{m=1}^{M}(2m+1) = M(M+2) \qquad \text{(Eq.3.28)}$$

3.1.1.2 Lattice Implementation of LPC Filters

Linear prediction coefficients are computed recursively using the Levinson-Durbin algorithm. The first order prediction coefficient a_{11} is the same as the reflection coefficient κ_1. The m^{th} order linear prediction coefficients are obtained from the $(m-1)^{th}$ order prediction coefficients and the reflection coefficient κ_m. Thus, M linear prediction coefficients are equivalent to M reflection coefficients. If reflection coefficients are given, the corresponding linear prediction coefficients can be obtained or vice versa. Quantization of reflection coefficients is easier because of the well-defined range of values that they take on. Note that the absolute value of reflection coefficients is never greater than one. This is why reflection coefficients instead of linear prediction coefficients are often used to represent a vocal tract filter. In this section, linear predictive coding (lpc) filters are implemented in a lattice form using reflection coefficients.

The prediction error for the m^{th} order prediction is rewritten as Eq.3.28.

$$\varepsilon_m(n) = x(n) - \sum_{i=1}^{m} a_{im} x(n-i) \qquad \text{(Eq.3.28)}$$

where $\varepsilon_m(n)$ indicates that this error is the forward prediction error. Let us assume that the backward linear prediction of $x(n-m)$ is made based on $x(n)$, $x(n-1)$, ..., and $x(n-m+1)$. The backward prediction error $\beta_m(n)$ is defined as Eq.3.29.

$$\beta_m(n) = x(n-m) - \sum_{i=1}^{m} a_{im} x(n-m+i) \qquad \text{(Eq.3.29)}$$

Now the $(m-1)^{\text{th}}$ forward prediction error is given by Eq.3.30.

$$\varepsilon_{m-1}(n) = x(n) - \sum_{i=1}^{m-1} a_{i(m-1)} x(n-i) \qquad \text{(Eq.3.30)}$$

The $(m-1)^{\text{th}}$ backward prediction error is as Eq.3.31.

$$\beta_{m-1}(n) = x(n-m+1) - \sum_{i=1}^{m-1} a_{i(m-1)} x(n-m+1+i) \qquad \text{(Eq.3.31)}$$

Because the recursive formula for linear prediction coefficients is given by Eq.3.32.

$$a_{im} = a_{i(m-1)} - \kappa_m a_{(m-i)m-1)} \quad \text{for } i = 1, 2, ..., m - 1$$
$$\text{a}_{\text{mm}} = \kappa_m \qquad \text{(Eq.3.32)}$$

Eq.3.23 can be shown.

$$\varepsilon_m(n) = \varepsilon_{m-1}(n) - \kappa_m \beta_{m-1}(n-1)$$
$$\beta_m(n) = \beta_{m-1}(n-1) - \kappa_m \varepsilon_{m-1}(n) \qquad \text{(Eq.3.33)}$$
$$\text{for } m = 1, 2, ..., M$$

The initial values are given by Eq.3.34.

$$\varepsilon_0(n) = \beta_0(n) = x(n) \qquad \text{(Eq.3.34)}$$

The final value is given by Eq.3.35.

$$\varepsilon(n) = \varepsilon_M(n) \qquad \text{(Eq.3.35)}$$

Thus, the analysis filter can be implemented as shown in Figure 3.4 where the input is the speech sequence and the output is the forward prediction error.

From each frame of speech samples, M reflection coefficients are computed. Because important information about the vocal tract model is extracted in the form of reflection coefficients, the output of the lpc analysis filter using reflection coefficients will have less redundancy than the original speech. Thus, less number of bits is required to quantize this so-called residual error. This quantized residual error along with the quantized reflection coefficients are transmitted or stored. To play back, a lattice implementation of the lpc synthesis filter is required. In this case, the input is the residual error and the output is the reconstructed speech. By reversing all the arrows in the top part of the analysis filter, one can implement the synthesis filter as shown in Figure 3.5.

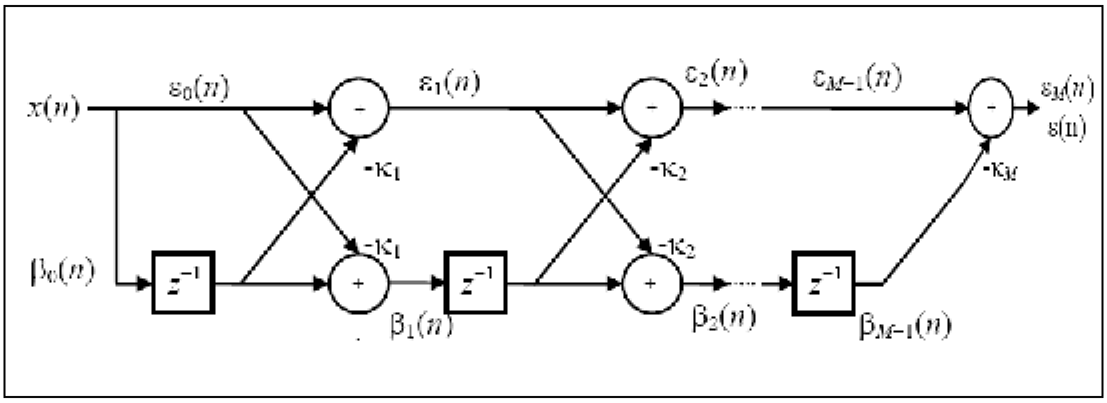


Figure 3.4 Lattice implementation of the lpc analysis filter using reflection coefficients.

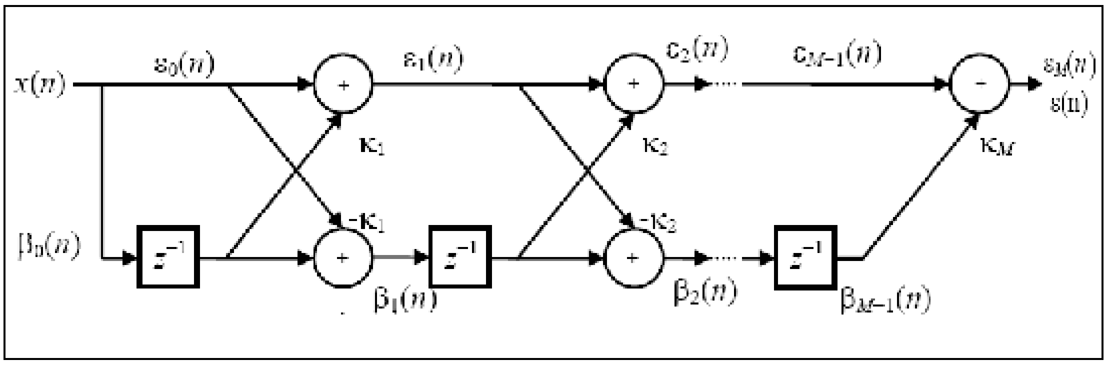


Figure 3.5 Lattice implementation of the lpc synthesis filter using reflection coefficients.

In the synthesis filter, the initial value is as Eq.3.36.

$$\varepsilon_M(n) = \varepsilon(n) \qquad \text{(Eq.3.36)}$$

The final values are as Eq.3.37 and Eq.3.38.

$$\varepsilon_0(n) = \beta_0(n) = x(n) \qquad \text{(Eq.3.37)}$$

$$\begin{aligned} &\varepsilon_{m-1}(n) = \varepsilon_m(n) + \kappa_m \beta_{m-1}(n-1) \\ &\beta_m(n) = \beta_{m-1}(n-1) - \kappa_m \varepsilon_{m-1}(n) \\ &\text{for } m = M, M-1, M-2, \ldots, 2, 1 \end{aligned} \qquad \text{(Eq.3.38)}$$

3.1.2 Parcor Coefficients

Assume that $\{s(0), s(1), ..., s(t-1)\}$ is a sequence of speech signals. After that forward autoregressive (AR) model of order *m* is given as Eq.3.39.

$$s(l) = \sum_{i=1}^{m} a^m(i) s(l-i) + \varepsilon_f^m(l) \qquad \text{(Eq.3.39)}$$

$\{a^m(i)\}_{i=1}^m$ and $\varepsilon_f^m(l)$ are forward AR coefficients and forward prediction error, respectively. Similarly, backward AR model of order *m* is given as Eq.3.40.

$$s(l-m-1) = \sum_{i=1}^{m} b^m(i) s(l-i) + \varepsilon_b^m(l) \qquad \text{(Eq.3.40}$$

$\{b^m(i)\}_{i=1}^m$ and $\varepsilon_b^m(l)$ are backward AR coefficients and backward prediction error, respectively. The AR coefficients $\{a^m(i)\}_{i=1}^m$ or $\{b^m(i)\}_{i=1}^m$ can be determined so that the mean squares error minimized.

Partial autocorrelation (parcor) coefficients (Bourlard & Morgan, 1997; Carson & Berndsen, 1998; Özgür, 1997) are used in speech signal processing and often more useful than AR coefficients $\{a^m(i)\}_{i=1}^m$ or $\{b^m(i)\}_{i=1}^m$. Parcor coefficient P^m of *m* is defined as a correlation coefficient between forward and backward prediction errors in the autoregressive model of order *m*-1. Namely, it can be stated as Eq.3.41.

$$P^m = \frac{\sum_{i=m}^{t-1} \varepsilon_f^{m-1}(i) \varepsilon_b^{m-1}(i)}{\sqrt{\sum_{i=m}^{t-1} \{\varepsilon_f^{m-1}(i)\}^2 \{\varepsilon_b^{m-1}(i)\}^2}} \qquad \text{(Eq.3.41)}$$

It is well known that the parcor coefficient P^m is the same as the AR coefficient $a^m(m)$ or $b^m(m)$ of AR model of order *m*. These coefficients are calculated from autocorrelations of the sequence of signals.

There is a fast recursive algorithm for calculating AR and parcor coefficients. The algorithm can compute all AR and parcor coefficients of orders 1 through *m* with $O(m^2)$

For online computation of parcor coefficients, we can use the recursive formula with forgetting factor $0 < \alpha < 1$ to estimate the autocorrelations $r(l)$ as Eq.3.42.

$$r(l)^{(t+1)} \leftarrow (1-\alpha)r(l)^{(t)} + \alpha s(t+1)s(t+1) \qquad \text{(Eq.3.42)}$$

From the estimated autocorrelations, we can calculate the parcor coefficients with $O(m^2)$ computation.

3.1.3 Cepstrum Coefficients

Linear predictive analysis is based on a model of the vocal tract as an all-pole filter (Rabiner & Juang, 1993). The lpc coefficients are a short time measure of speech signal as the output of this all-pole filter. Although it has been designed to model speech production, it is also partially valid for musical instruments. In this case, the filter embodies the effect of resonating body of instrument, namely, its timbre.

An alternative feature for lpc coefficient is lpc derived cepstral coefficient, which can be computed simply as Eq.3.43.

$$c_n = a_n + \frac{1}{n}\sum_{i=1}^{n-1}(n-i)a_i c_{n-i} \qquad \text{(Eq.3.43)}$$

a_n is the lpc coefficient. The principal advantage of lpc derived cepstral coefficients is that they are generally decorrelated.

3.1.4 Mel Frequency Cepstral Coefficients

Because of the known variation of the ear's critical band-widths with frequency, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. Davis & Mermelstein (1980) showed that the first six eigenvectors of the covariance matrix for Dutch vowels of the speakers, expressed in terms of 17 such filter energies, accounted for 91.8 percent of the total variance. The direction cosines of this eigenvectors were very similar to a cosine series expansion on the filter energies. Additional eigenvectors showed an increasing number of oscillations of their direction cosines with respect to their original energies. This result suggested that a compact representation would be provided by a set of mel-frequency cepstrum coefficients. These cepstrum coefficients are the result of a cosine transform of the real logarithm of the short-time energy spectrum expressed on a Mel-frequency scale.

In mfcc, the main advantage is that it uses mel frequency scaling which is very approximate to the human auditory system.

Researchers have undertaken psychoacoustic experimental work to derive frequency scales that attempt to model the natural response of the human perceptual system, since the cochlea of the inner ear acts as a spectrum analyzer. The complex mechanism of the inner ear and auditory nerve implies that the perceptual attributes of sounds at different frequencies may not be entirely simple or linear in nature. AT&T Bell Labs has contributed many influential discoveries in hearing, such as critical bands. The cochlea in our auditory system acts as if it was made up of overlapping filters having bandwidths equal to the critical bandwidth (Huang & Hon, 2001). So the skill of frequency scaling is used to map linear frequency into human perception. Mel-frequency scale is such a kind of perceptually motivated scale, which is linear below 1 KHz, and logarithmic above. One mel is defined as one thousand of the pitch of a 1 KHz tone. As with all attempts, it is hoped that the mel scale more closely models the sensitivity of the human ear than a purely linear scale and provides for greater discriminatory capability between speech segments. Mel-scale frequency analysis has been widely used in current speech recognition system. It can be approximated by Eq.3.44.

$$B(f) = 1125 \ln(1 + f/700) \qquad \text{(Eq.3.44)}$$

Where *B* is the Mel-frequency scale, *f* is the linear frequency.

Given that the DFT of the input signal in Eq.3.45.

$$X_a(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k/N}, \quad 0 \le k \le N \qquad \text{(Eq.3.45)}$$

And we define mel-frequency filter bank with *p* filters m_j (*j*=1,2,…,*p*), where filter *m* is triangular filter shown in the Figure 3.6.

Each FFT magnitude coefficient is multiplied by the corresponding filter gain and the results accumulated. It can be computed as Eq.3.46.

$$m_j = \sum_{k=0}^{N-1} |X_a(k)|^2 H_j(k), \quad 0 \le j \le p \qquad \text{(Eq.3.46)}$$

Where $H_j[k]$ is the transfer function of filter *j*. The mel frequency cepstrum is then the discrete cosine transform of the p filter outputs. It's described as Eq.3.47.

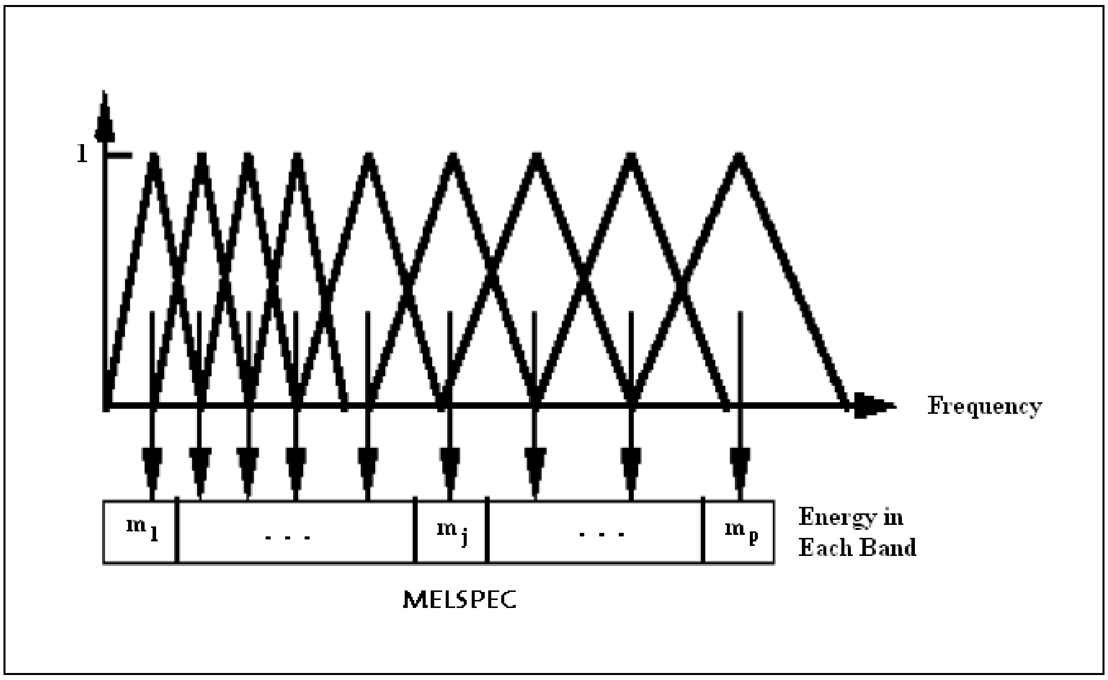


Figure 3.6 Mel frequency filter bank.

$$c_i = \sqrt{\frac{2}{N}} \sum_{i=1}^{p} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \qquad \text{(Eq.3.47)}$$

For speech recognition, normally only the first 13 Cepstrum coefficients are used (Young & Kershaw, 2000).

3.1.5 RelAtive SpecTrAl (RASTA) Features

Noisy environments decreases the performance of the speech recognition systems. RASTA (Hermansky & Morgan, 1994) was developed to eliminate the environmental factors. The method filters out distortions and noises caused by environmental factors and increases speech recognition performance.

In this method, it is replaced a common short term absolute spectrum by a spectral estimate in which every frequency channel is band pass filtered by a filter with sharp spectral zero at the zero frequency. The spectral estimate is less sensitive to slow variations in the short term spectrum.When the filtering operation is applied in the logarithmic spectral domain, the suppressed constant spectral component reflect the effect of the convolutive factors in the digital speech signal.

The algorithm of RASTA are given as the following. The algorithm is applied for each speech frame.

1. Compute the critical band spectrum and take its logarithm.
2. Estimate the temporal derivative of the log critical band spectrum using regression line through five consecutive spectral values.
3. Nonlinear processing such as applying threshold of median filtering can be done in this domain.
4. Reintegrate the log critical band temporal derivative using a first order IIR system. The pole position of this system can be adjusted to set the effective window size.
5. Add the equal loudness curve and multiply by 0.33 to simulate the power law of hearing.
6. Take the inverse logarithm (exponential function) of this relative log spectrum, yielding a relative auditory spectrum.
7. Compute an all-pole model of this spectrum.

If the derivative of Step 2 is estimated by a simple first differece, and if the full integration in Step 4 is done (pole at z=1.0), then all intermediate terms cancel and the technique is equivalent to substraction of the log spectrum of the first analysis frame from each new frame. In this special case, the RASTA technique resembles the spectral substraction or blind deconbolution techniques. The whole derivative-integration process is equivalent to a bandpass filtering of each frequency channel through an IIR filter with the transfer function as Eq.3.48.

$$H(z) = 0.1 * \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{z^{-4} * (1 - 0.98z^{-1})} \qquad \text{(Eq.3.48)}$$

The low cut-off frequency of the filter determines the fastest spectral change of the log spectrum which is ignored in the output, while the high cut-off frequency determines the fastest spectral change which is preserved.

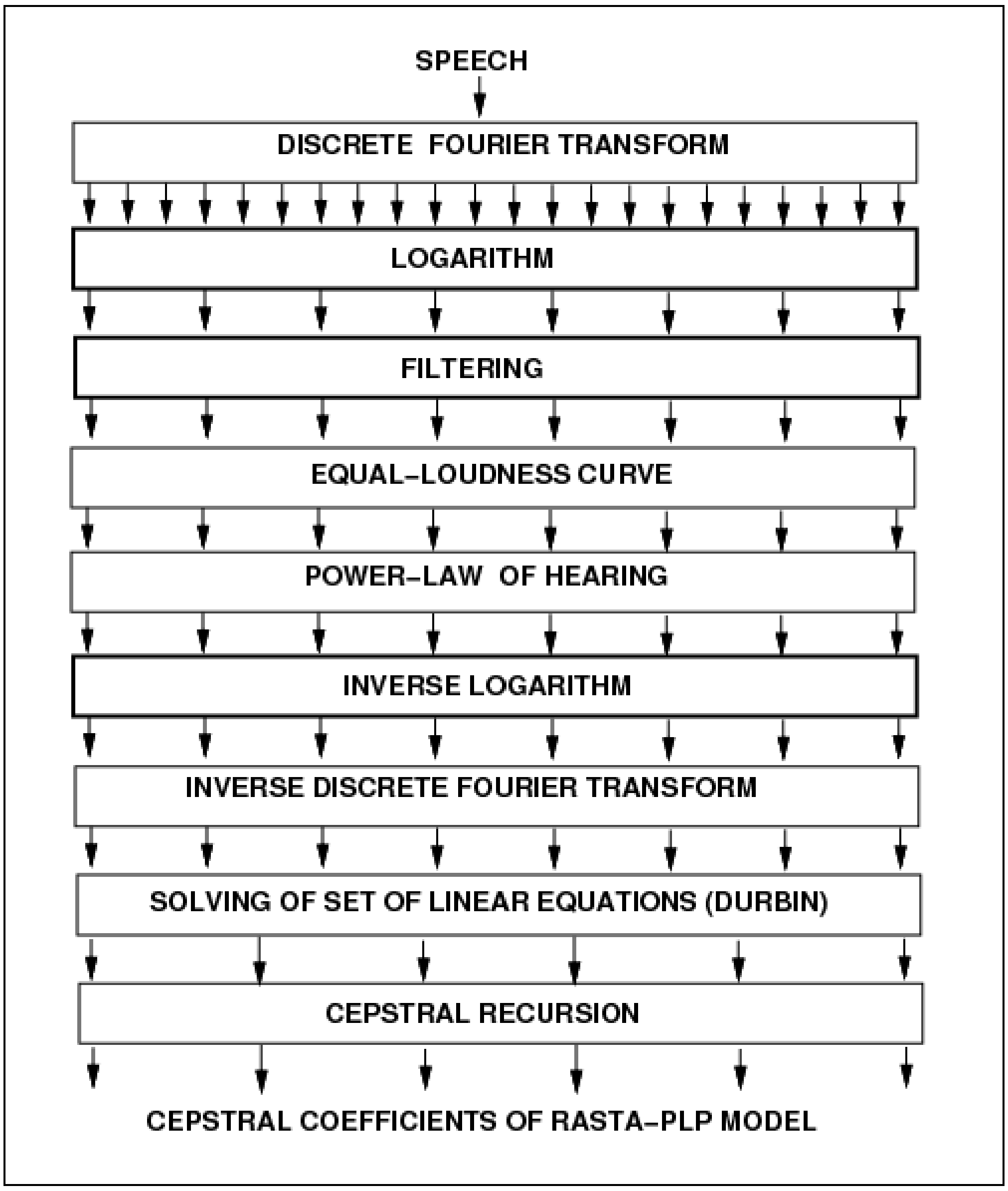

SPEECH
DISCRETE FOURIER TRANSFORM
LOGARITHM
FILTERING
EQUAL-LOUDNESS CURVE
POWER-LAW OF HEARING
INVERSE LOGARITHM
INVERSE DISCRETE FOURIER TRANSFORM
SOLVING OF SET OF LINEAR EQUATIONS (DURBIN)
CEPSTRAL RECURSION
CEPSTRAL COEFFICIENTS OF RASTA-PLP MODEL

Figure 3.7 RASTA process.

Linear distortions appear as an additive constant in the log spectrum. The high pass portion of the equivalent band-pass filter is expected to alleviate the effect of the convolutional noise introduced in the channel. The low pass filtering is expected to help in smoothing out some of fast frame to frame spectral changes present in the short time spectral estimate due to analysis artifacts. In Eq.3.48, low cut-off frequency is 0.26 Hz. The filter slope decreases 6 dB/oct from 12.8 Hz with sharp zeros at 28.9 Hz and at 50 Hz. The whole process is illutrated in Figure 3.7.

3.2 Speech Recognition Methods

3.2.1 Linear Time Alignment (*LTA*)

Linear time alignment algorithms are the simplest algorithms to implement and they can be used for both expansion and compression of the speech pattern vector. There are various ways of implementing linear algorithms, but all of them use the basic method of deleting feature vectors to shorten the speech pattern and duplicating feature vectors to length the speech pattern. An example is to duplicate or delete vectors at regular intervals along the pattern vector until the speech pattern is the correct size. An example of a linear algorithm used in conjunction with a neural network is that of Woodland (1990). Woodland achieved recognition rates of 91% for multiple speaker recognition and 88.3% for speaker independent recognition.

3.2.2 Dynamic Time Warping (*DTW*)

Dynamic time warping (DTW) is a technique that finds the optimal alignment between two time series if one time series can be “warped” non-linearly by stretching or shrinking it along its time axis. This warping between two time series can then be used to find corresponding regions between the two time series or to determine the similarity between the two time series. Dynamic time warping is often used in speech recognition to determine if two waveforms represent the same spoken phrase. In a speech waveform, the duration of each spoken sound and the interval between sounds are permitted to vary, but the overall speech waveforms must be similar. In addition to speech recognition, dynamic time warping has also been found useful in many other disciplines, including data mining, gesture recognition, robotics, manufacturing, and medicine. Dynamic time warping is commonly used in data mining as a distance measure between time series. An example of how one time series is “warped” to another is shown in Figure 3.8.

In Figure 3.8, each vertical line connects a point in one time series to its correspondingly similar point in the other time series. The lines actually have similar values on the y-axis but have been separated so the vertical lines between them can be viewed more easily. If both of the time series in Figure 3.8 were identical, all of the lines would be straight vertical lines because no warping would be necessary to 'line up' the two time series. The warp path distance is a measure of the difference between the two time series after they have been warped together, which is measured by the sum of the distances between each pair of points connected by the vertical lines in Figure 3.8. Thus, two time series that are identical except for localized stretching of the time axis will have DTW distances of zero.

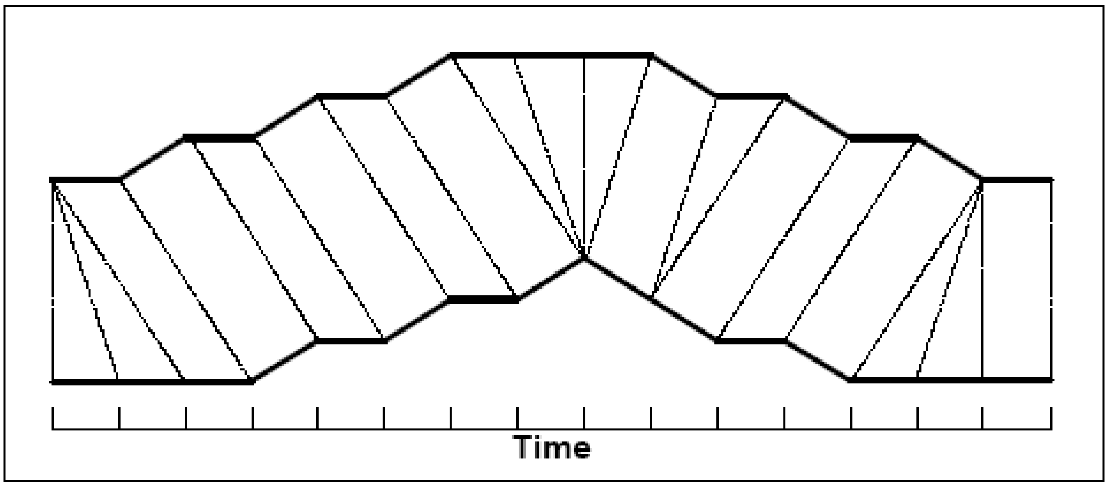


Figure 3.8 A warping between two time series.

A distance measurement between time series is needed to determine similarity between time series and for time series classification. Euclidean distance is an efficient distance measurement that can be used. The Euclidian distance between two time series is simply the sum of the squared distances from each n^{th} point in one time series to the n^{th} point in the other. The main disadvantage of using Euclidean distance for time series data is that its results are very unintuitive. If two time series are identical, but one is shifted slightly along the time axis, then Euclidean distance may consider them to be very different from each other. Dynamic time warping (DTW) was introduced (Kruskall & Liberman, 1983) to overcome this limitation and give intuitive distance measurements between time series by ignoring both global and local shifts in the time dimension.

3.2.2.1 Problem Formulation

The dynamic time warping problem is stated as Eq.3.49. Given two time series X, and *Y*, of lengths $|X|$ and $|Y|$.

$$\begin{aligned} X &= x_1, x_2, \ldots, x_i, \ldots, x_{|X|} \\ Y &= y_1, y_2, \ldots, y_3, \ldots, y_{|Y|} \end{aligned} \qquad \text{(Eq.3.49)}$$

A warp path W is constructed as Eq.3.50.

$$\begin{aligned} &W = w_1, w_2, \ldots, w_K \\ &\max(|\mathrm{X}|, |Y|) \le K < |X| + |Y| \end{aligned} \qquad \text{(Eq.3.50)}$$

where *K* is the length of the warp path and the *k*th element of the warp is $w_k = (i, j)$, where *i* is an index from time series *X*, and *j* is an index from time series *Y*. The warp path must start at the beginning of each time series at $w_1 = (1,1)$ and finish at the end of both time series at $w_K = (|X|, |Y|)$. This ensures that every index of both time series is used in the warp path. There is also a constraint on the warp path that forces *i* and *j* to be monotonically increasing in the warp path, which is why the lines representing the warp path in Figure 3.8 do not overlap. Every index of each time series must be used. Eq.3.51 states more formally.

$$w_k = (i, j), \quad w_{k+1} = (i', j') \quad i \le i' \le i+1 \quad j \le j' \le j+1 \qquad \text{(Eq.3.51)}$$

The optimal warp path is the warp path is the minimum-distance warp path, where the distance of a warp path W is as Eq.3.52.

$$Dist(W) = \sum_{k=1}^{K} Dist(w_{ki}, w_{kj}) \qquad \text{(Eq.3.52)}$$

Dist(*W*) is the distance of warp path *W*, and *Dist*(w_{ki}, w_{kj}) is the distance between the two data point indexes (one from *X* and one from *Y*) in the k^{th} element of the warp path.

3.2.3 Artificial Neural Networks (*ANN*)

The ANNs probably belong to the borderline between the Artificial Intelligence and Approximation Algorithms. The ANNs are used in universal approximation (mapping input to the output), tools capable of learning from their environment.

The Neural Networking algorithms model the brain (not necessarily - human brain) and how it processes the information. The brain is a very efficient tool. Having about 100,000 times slower response time than computer chips, it beats the computer in complex tasks, such as image and sound recognition, motion control and so on. It is also about 10,000,000,000 times more efficient than the computer chip in terms of energy consumption per operation.

The brain is a multi layer structure that works as a parallel computer capable of learning from the “feedback”. It receives from the world and changing its design by growing new neural links between neurons or altering activities of existing ones. The brain is composed of neurons, interconnected.

3.2.3.1 The Biological Neuron

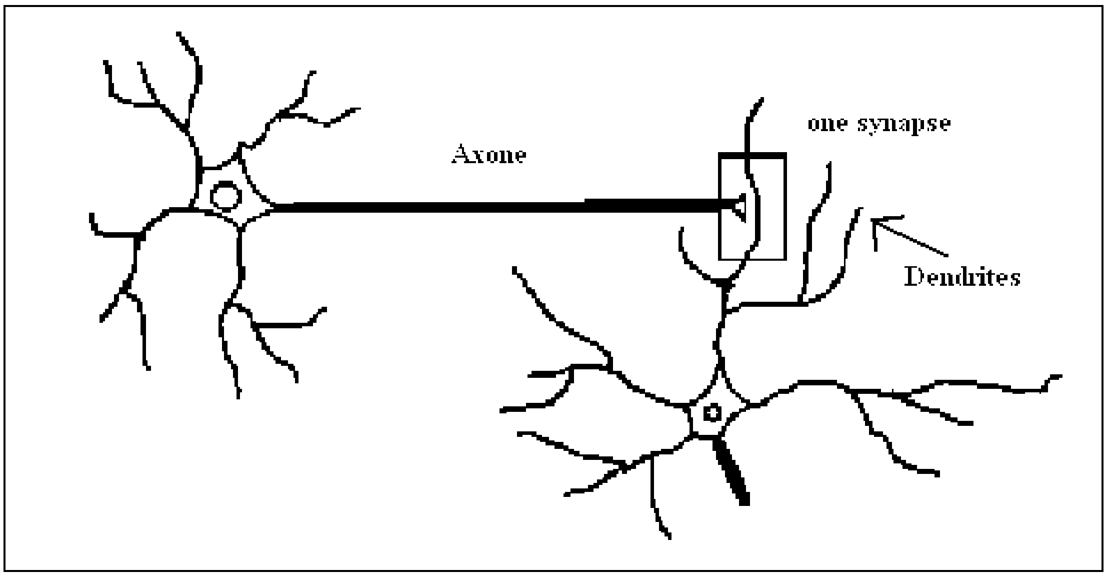


Figure 3.9 A biological neuron.

This neuron has two parts very interesting to us, called the synapse and the dendrite as shown in Figure 3.9. The dendrites are extensions of a neuron which connect to other neurons to form a neural network, while synapses are a gateway which connects to dendrites that come from other neurons. A biological neuron may thus be connected to other neurons as well as accepting connections from other neurons, and so we have the basis of a network.

Through those connections, electrical pulses are transmitted, and information is carried in the timing and the frequency with which these pulses are emitted.

So, our neuron receives information from other neurons, processes it and then relays this information to other neurons. A question which immediately arises is: of what form does this processing take? Clearly, the neuron must generate some kind of output based on the cumulative input. We still do not know the exact answer to the question as to what happens in a biological neuron. However, we do know that our neuron integrates the pulses that arrive and when this integration exceeds a certain limit, our neuron in turn emits a pulse.

Finally, one more thing that you should know is that dendrites modify the amplitude of the pulses traveling through them. This modification varies with time, as the network "learns".

So we could assume that when a connection (dendrite) is very strong, the importance of the neuron from which this connection come has an important role in the network, and on the other hand, when a connection is very narrow, the importance of the neuron from which the connection comes from is less high. Thus the neural network stores information in the pattern of connection weights.

3.2.3.2 Structure of a Neuron

Our "artificial" neuron will have inputs (all *N* of them) and one output. In Figure 3.10 the neuron has set of nodes that connect to inputs, output, or other neurons, also called synapses.

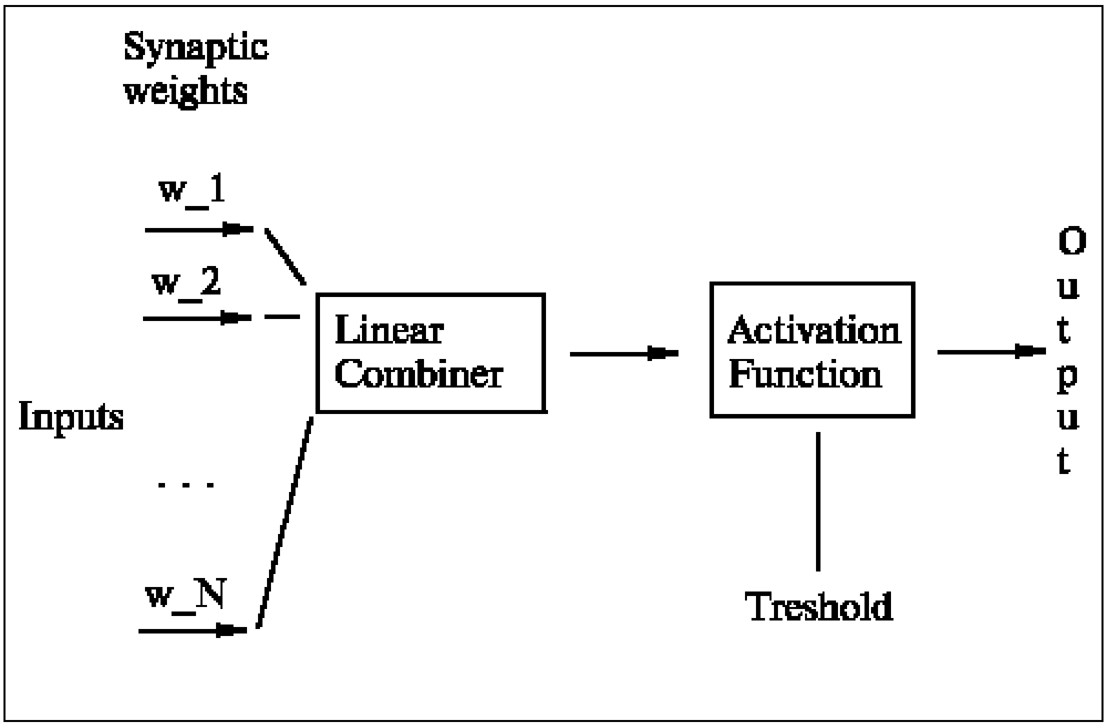


Figure 3.10 A neuron structure.

A Linear Combiner, which is a function that takes all inputs and produces a single value. A simple way of doing it is by adding together the dInput ("d" prefix means "double", we use it so that the name (*dInput*) represents the floating point number) multiplied by the Synaptic Weight (*dWeight*).

How do we make a non-linear input? By applying the Activation Function, it will take any input from minus infinity to plus infinity and squeeze it into the -1 to 1 or into 0 to 1 interval.

Finally, we have a threshold. What should the internal activity of a neuron be when there is no input? Should there be some threshold input before we have the activity? Or should the activity be present as some level (in this case it is called a bias rather than a threshold) when the input is zero?

3.2.3.3 A Neural Net

A single neuron by itself is not a very useful pattern recognition tool. The real power of neural networks comes when we combine neurons into the multilayer structures, called neural networks.

As you can see in Figure 3.11, there are 3 layers in our network. There are *N* neurons in the first layer, where *N* equals number of inputs. There are *M* neurons in the output layer, where *M* equals number of outputs. For example, when you are building the network capable of predicting the stock price, you might want the yesterday's high, low and close volume as inputs and close as the output.

You may have any number of neurons in the inner (also called "hidden") layers. If you have too few, the quality of a prediction will drop and the net doesn't have enough "brains". And if you make it too many - it will have a tendency to "remember" the right answers, rather than predicting them. Then your neural net will work very well on the familiar data, but will fail on the data that was never presented before.

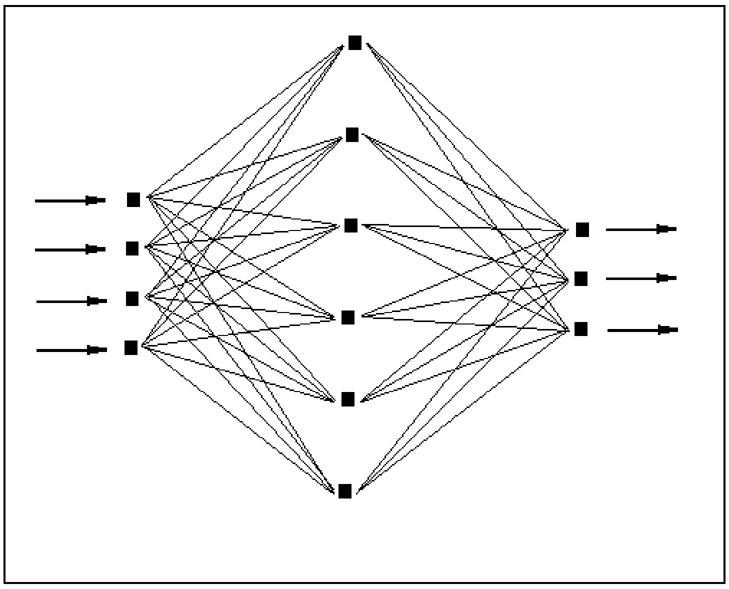


Figure 3.11 A simple neural net.

3.2.3.4 Backpropagation

A single-layer network has severe restrictions: the class of tasks that can be accomplished is very limited. In this chapter we will focus on feed-forward networks with layers of processing units. Minsky and Papert (1969) showed that a two layer feed-forward network can overcome many restrictions, but did not present a solution to the problem of how to adjust the weights from input to hidden units. An answer to this question was presented by Rumelhart, Hinton and Williams (1986), and similar solutions appeared to have been published earlier (Werbos, 1974; Parker, 1985; Le Cun, 1985). The central idea behind this solution is that the errors for the units of the hidden layer are determined by back-propagating the errors of the units of the output layer. For this reason the method is often called the back-propagation learning rule. Back-propagation can also be considered as a generalization of the delta rule for non-linear activation functions and multilayer networks.

3.2.3.4.1 Multi-layer Feed-forward Networks. A feed-forward network has a layered structure. Each layer consists of units which receive their input from units from a layer directly below and send their output to units in a layer directly above the unit. There are no connections within a layer. The N_i inputs are fed into the first layer of N_h; one hidden unit. The input units are merely 'fan-out' units; no processing takes place in these units. The activation of a hidden unit is a function F_i of the weighted inputs plus a bias, as given in Eq.3.53.

$$y_k(t+1) = F_k(s_k(t)) = F_k\left(\sum_j w_{jk}(t)y_j(t) + \theta_k(t)\right) \qquad \text{(Eq.3.53)}$$

The output of the hidden units is distributed over the next layer of N_h; 2 hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of N_o output units as shown in Figure 3.12.

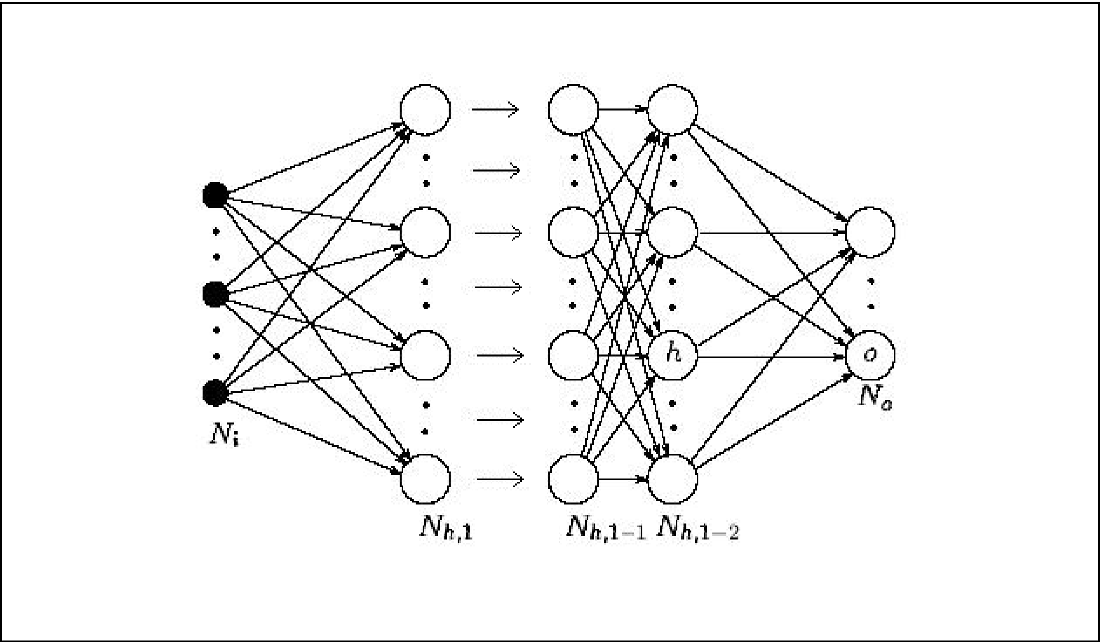


Figure 3.12 Neural network with hidden layers.

Although backpropagation can be applied to networks with any number of layers, just as for networks with binary units it has been shown (Hornik, Stinchcombe, & White, 1989; Funahashi, 1989; Cybenko, 1989; Hartman, Keeler, & Kowalski, 1990) that only one layer of hidden units success to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden units are non-linear (the universal approximation theorem). In most applications a feed-forward network with a single layer of hidden units is used with a sigmoid activation function for the units.

3.2.4 Hidden Markov Models (*HMM*)

HMM (Rabiner, 1989) is a result of the attempt to model the speech generation statistically. During the past several years it has become the most successful speech model used in speech recognition. The main reason for this success is its wonderful ability to characterize the speech signal in a mathematically tractable way.

In a HMM based speech recognition system (Ferguson, 1980; Juang & Rabiner, 1990; Pepper, Barnwell, & Clements, 1990), the HMM stage is proceeded by the preprocessing (feature parameter extraction) stages. Thus the input to the HMM is a discrete time sequence of feature parameter vectors. The parameter vectors can be supplied to the HMM, either in vector quantized form or in raw continuous form. It can be designed HMMs to handle any of the cases, but important point is how the HMM deals with the stochastic nature of the amplitudes of the feature vectors which is superimposed on the time stochasticity.

The HMM is a finite set of states, each of which is associated with a probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are “hidden” to the outside; hence the name Hidden Markov Model.

In order to define an HMM completely, following elements are needed.

- The number of states of the model, N.
- The number of observation symbols in the alphabet, M. If the observations are continuous then M is infinite.
- A set of state transition probabilities as shown in Eq.3.54.

$$A = \{a_{ij}\}$$
$$a_{ij} = p\{q_{t+1} = j \mid q_t = i\}, \quad 1 \le i,j \le N \qquad \text{(Eq.3.54)}$$

q_t denotes the current state. Transition probabilities should satisfy the normal stochastic constraints as Eq.3.55 and Eq.3.56.

$$a_{ij} \ge 0, \quad 1 \le i, j \le N \qquad \text{(Eq.3.55)}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \le i \le N \qquad \text{(Eq.3.56)}$$

- A probability distribution in each of the states as shown as Eq.3.57.

$$B = \{b_j(k)\}$$
$$b_j(k) = p\{0_t = v_k \mid q_t = j\}, \quad 1 \le j \le N \quad 1 \le i \le M \qquad \text{(Eq.3.57)}$$

v_k denotes the k^{th} observation symbol in the alphabet, and o_t the current parameter vector.

The stochastic constraints as Eq.3.58 and Eq.3.59 must be satisfied.

$$b_j(k) \ge 0, \quad 1 \le j \le N, \quad 1 \le k \le M \qquad \text{(Eq.3.58)}$$

$$\sum_{k=1}^{M} b_j(k) = 1, \quad 1 \le j \le N \qquad \text{(Eq.3.59)}$$

If the observations are continuous, then we will have to use a continuous probability density function, instead of a set of discrete probabilities. In this case we specify the parameters of the probability density function. Usually the probability density is approximated by a weighted sum of M Gaussian distributions Ω as Eq.3.60.

$$\begin{aligned} &b_j(o_t) = \sum_{m=1}^{M} c_{jm}\Omega(\mu_{jm}, \Sigma_{jm}, o_t) \\ &c_{jm} = \text{weighting coefficients} \\ &\mu_{jm} = \text{mean vectors} \\ &\Sigma_{jm} = \text{covariance matrices} \end{aligned} \qquad \text{(Eq.3.60)}$$

c_{jm} should satisfy the stochastic constraints as Eq.3.61 and Eq.3.62.

$$c_{jm} \geq 0, \quad 1 \leq j \leq N, \quad 1 \leq m \leq M \qquad \text{(Eq.3.61)}$$

$$\sum_{m=1}^{M} c_{jm} = 1, \quad 1 \leq j \leq N \qquad \text{(Eq.3.62)}$$

- The initial state distribution as shown in Eq.3.63.

$$\begin{aligned} &\pi = \{\pi_i\} \\ &\pi_i = p\{q_1 = i\}, \quad 1 \leq i \leq N \end{aligned} \qquad \text{(Eq.3.63)}$$

Therefore we can use the compact notation as Eq.3.64 to denote an HMM with probability distributions while to denote an HMM with continuous densities as Eq.3.65.

$$\lambda = (A, B, \pi) \qquad \text{(Eq.3.64)}$$

$$\lambda = (A, c_{jm}, \mu_{jm}, \Sigma_{jm}, \pi) \qquad \text{(Eq.3.65)}$$

3.2.4.1 Assumptions of HMMs

For the sake of mathematical and computational tractability, following assumptions are made in the theory of HMMs.

3.2.4.1.1 The Markov Assumption. As given in the definition of HMMs, transition probabilities are defined as Eq.3.66.

$$a_{ij} = p\{q_{t+1} = j \mid q_t = i\} \qquad \text{(Eq.3.66)}$$

In other words, it is assumed that the next state is dependent only upon the current state. This is called the Markov assumption and the resulting model becomes actually a first order HMM.

However, generally the next state may depend on past k states and it is possible to obtain such a model, called an k^{th} order HMM by defining the transition probabilities as Eq.3.67.

$$\begin{gathered} a_{i_1,i_2,\ldots,i_k,j} = p\{q_{t+1} = j \mid q_t = i_1, q_{t-1} = i_2, \ldots, q_{t-k+1} = i_k\} \\ 1 \le i_1, i_2, \ldots, i_k, j \le N \end{gathered} \qquad \text{(Eq.3.67)}$$

But, it is seen that a higher order HMM will have a higher complexity. Even though the first order HMMs are the most common, some attempts have been made to use the higher order HMMs too.

3.2.4.1.2 The Stationary Assumption. Here, it is assumed that state transition probabilities are independent of the actual time at which the transitions take place. Mathematically, it can be stated as Eq.3.68.

$$p\{q_{t_1+1} = j \mid q_{t_1} = i\} = p\{q_{t_2+1} = j \mid q_{t_2} = i\} \quad \text{for any } t_1 \text{ and } t_2 \qquad \text{(Eq.3.68)}$$

3.2.4.1.3 The Output Independence Assumption. This is the assumption that current output(observation) is statistically independent of the previous outputs (observations). We can formulate this assumption mathematically, by considering a sequence of observations as Eq.3.69.

$$O = o_1, o_2, ..., o_t \qquad \text{(Eq.3.69)}$$

Then, according to the assumption for an HMM λ as Eq.3.70.

$$p\{O \mid q_1, q_2, ..., q_T, \lambda\} = \prod_{t=1}^{T} p(o_t \mid q_t, \lambda) \qquad \text{(Eq.3.70)}$$

However, unlike the other two, this assumption has a very limited validity. In some cases this assumption may not be fair enough and therefore becomes a severe weakness of the HMMs.

3.2.4.2 Three Basic Problems of HMMs

Once we have an HMM, there are three problems of interest.

3.2.4.2.1 The Evaluation Problem. Given an HMM λ and a sequence of observations $O = o_1, o_2, ..., o_T$, what is the probability that the observations are generated by the model as Eq.3.71?

$$p\{O \mid \lambda\} \qquad \text{(Eq.3.71)}$$

3.2.4.2.2 The Decoding Problem. Given an HMM λ and a sequence of observations $O = o_1, o_2, ..., o_T$, what is the most likely state sequence in the model that produced the observations?

3.2.4.2.3 The Learning Problem. Given an HMM λ and a sequence of observations $O = o_1, o_2, ..., o_T$, how should we adjust the model parameters $\{A, B, \pi\}$ in order to maximize $p\{O \mid \lambda\}$?

Evaluation problem can be used for isolated word recognition. Decoding problem is related to the continuous recognition as well as to the segmentation. Learning problem must be solved if we want to train an HMM for the subsequent use of recognition tasks.

3.2.4.3 The Evaluation Problem and the Forward Algorithm

We have a model $\lambda = (A, B, \pi)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, and $p\{O \mid \lambda\}$ must be found. We can calculate this quantity using simple probabilistic arguments. But, this calculation involves number of operations in the order of N^T. This is very large even if the length of the sequence, T is moderate. Therefore, for this calculation we have to look for another method. Fortunately, there exists one which has a considerably low complexity and makes use an auxiliary variable, $\alpha_t(i)$ called "forward variable".

The forward variable is defined as the probability of the partial observation sequence $o_1, o_2, ..., o_T$, when it terminates at the state *i*. Mathematically, it can be stated as Eq.3.72.

$$\alpha_t(i) = p\{o_1, o_2, ..., o_t, q_t = i \mid \lambda\} \qquad \text{(Eq.3.72)}$$

Then, it is easy to see that the recursive relationship as Eq.3.73 holds.

$$\begin{aligned} &\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, \quad 1 \le j \le N, \quad 1 \le t \le T-1 \\ &\alpha_1(j) = \pi_j b_j(o_1), \quad 1 \le j \le N \end{aligned} \qquad \text{(Eq.3.73)}$$

Using this recursion we can calculate Eq.3.74.

$$\alpha_T(i), \quad 1 \le i \le N \tag{Eq2.74}$$

And then, the required probability is given by Eq.3.75.

$$p\{O \mid \lambda\} = \sum_{i=1}^{N} \alpha_T(i) \tag{Eq.3.75}$$

The complexity of this method, known as the "forward algorithm" is proportional to $N^2 T$, which is linear with respect to T whereas the direct calculation had an exponential complexity.

In a similar way, we can define the "backward variable" $\beta_t(i)$ as the probability of the partial observation sequence $o_{t+1}, o_{t+2}, \ldots, o_T$, given that the current state is *i*. Mathematically , this can be stated as Eq.3.76.

$$\beta_t(i) = p\{o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda\} \tag{Eq.3.76}$$

As in the case of $\alpha_t(i)$ there is a recursive relationship which can be used to calculate $\beta_t(i)$ efficiently as shown in Eq.3.77.

$$\begin{aligned} &\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1}), \quad 1 \le i \le N, \quad 1 \le t \le T-1 \\ &\beta_T(i) = 1, \quad 1 \le i \le N \end{aligned} \tag{Eq.3.77}$$

We can also see the Eq.3.78.

$$\alpha_t(i)\beta_t(i) = p\{O, q_t = i \mid \lambda\}, \quad 1 \le i \le N, \quad 1 \le t \le T \tag{Eq.3.78}$$

Therefore, this gives another way to calculate $p\{O \mid \lambda\}$, by using both forward and backward variables as given in Eq.3.79.

$$p\{O \mid \lambda\} = \sum_{i=1}^{N} p\{O, q_t = i \mid \lambda\} = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \qquad \text{(Eq.3.79)}$$

Eq.3.79 is very useful, especially in deriving the formulas required for gradient based training.

3.2.4.4 The Decoding Problem and the Viterbi Algorithm

In this case, we want to find the most likely state sequence for a given sequence of observations, $O = o_1, o_2, ..., o_T$ and a model, $\lambda = (A, B, \pi)$.

The solution to this problem depends upon the way "most likely state sequence" is defined. One approach is to find the most likely state q_t at *t*=*t* and to concatenate all such "q_t"s. But, some times this method does not give a physically meaningful state sequence. Therefore, we would use another method which has no such problems. In this method, commonly known as "Viterbi algorithm", the whole state sequence with the maximum likelihood is found. In order to facilitate the computation we define an auxiliary variable as Eq.3.80 which gives the highest probability that partial observation sequence and state sequence up to *t*=*t* can have, when the current state is *i*.

$$\delta_t(i) = \max_{q_1, q_2, ..., q_{t-1}} p\{q_1, q_2, ..., q_{t-1}, q_t = i, o_1, o_2, ..., o_{t-1} | \lambda\} \qquad \text{(Eq.3.80)}$$

It is easy to observe that the recursive relationship as Eq.3.81 holds.

$$\begin{aligned} &\delta_{t+1}(j) = b_j(o_{t+1})\left[\max_{1 \le i \le N} \delta_t(i) a_{ij}\right], \quad 1 \le i \le N, \quad 1 \le t \le T-1 \\ &\delta_1(j) = \pi_j b_j(o_1), \quad 1 \le j \le N \end{aligned} \qquad \text{(Eq.3.81)}$$

So, the procedure to find the most likely state sequence starts from calculation of $\delta_T(j)$, $1 \le j \le N$ using recursion in Eq.3.81, while always keeping a pointer to the "winning state" in the maximum finding operation. Finally, the state j^* is found by Eq.3.82 and starting from this state. The sequence of states is back-tracked as the pointer in each state indicates. This gives the required set of states. This whole algorithm can be interpreted as a search in a graph whose nodes are formed by the states of the HMM in each of the time instant t, $1 \le t \le T$.

$$j^* = \arg \max_{1 \le j \le N} \delta_T(j) \qquad \text{(Eq.3.82)}$$

3.2.5 Support Vector Machines (*SVM*)

SVM have been introduced as a new technique for solving pattern recognition problems (Blanz et al., 1996; Cortes & Vapnik, 1995; Osuna, Freud & Girosi, 1997; Schölkopf et al., 1996). According to the theory of SVMs (Vapnik, 1982, 1995), while traditional techniques for pattern recognition are based on the minimization on the training set, SVMs minimize the structural risk. Namely, the probability of misclassifying to be seen patterns for a fixed but unknown probability distribution of the data. This new induction principle, which is equivalent to minimize an upper bound on the generalization error, relies on the theory of uniform convergence in probability (Vapnik, 1982). What makes SVMs attractive is the ability to condense the information contained in the training set, and the use of families of decision surfaces of relatively low VC-dimension (Vapnik & Chervonenkis, 1971).

In the linear, separable case the key idea of a SVM can be explained in plain words. Given a training set *S* which contains points of either of two classes, a SVM separates the classes through a hyper-plane determined by certain points of *S*, termed "support vectors". In the separable case, this hyper-plane maximizes the margin, or twice the minimum distance of either class from the hyper-plane, and all the support vectors lie at the same minimum distance from the hyper-plane. In real cases, the two classes may not be separable and both the hyper-plane and the support vectors are obtained from the solution of a problem of constrained optimization. The solution is a trade-off between the largest margin and the lowest number of errors, trade-off controlled by a regularization parameter.

3.2.5.1 Optimal Separating Hyper-plane

Assume S is a set of points $x_i \in R^n$ with $i = 1, 2, ..., N$. Each point x_i belongs to either of two classes and thus is given a label $y_i \in \{-1, 1\}$. The goal is to establish the equation of a hyper-plane that divides S leaving all the points of the same class on the same side while maximizing the minimum distance between either of the two classes and the hyper-plane. To this purpose we need some preliminary definitions.

<u>Definition:</u> The set S is linearly separable if there exists $w \in R^n$ and $b \in R$ such that Eq.3.83 is satisfied.

$$\begin{aligned} w \cdot x_i + b &\geq 1 \quad \text{if } y_i = 1, \\ w \cdot x_i + b &\leq -1 \quad \text{if } y_i = -1 \end{aligned} \qquad \text{(Eq.3.83)}$$

In compact notation, the two inequalities as Eq.3.83 can be rewritten as Eq.3.84.

$$y_i(w \cdot x_i + b) \geq 1, \quad \text{for } i = 1, 2, ..., N \qquad \text{(Eq.3.84)}$$

The pair (w,b) defines a hyper-plane of equation as Eq.3.85 named "separating hyper-plane" as shown Figure 3.14 and 3.15.

$$w \cdot x + b = 0 \qquad \text{(Eq.3.85)}$$

If we denote with w the norm of w, the signed distance d_i of a point x_i from the separating hyper-plane (w,b) is given by Eq.3.86.

$$d_i = \frac{w \cdot x_i + b}{w} \qquad \text{(Eq.3.86)}$$

Combining inequality as Eq.3.84 and Eq.3.86, for all $x_i \in S$ we have Eq.3.87.

$$y_i d_i \geq \frac{1}{w} \qquad \text{(Eq.3.87)}$$

In Figure 3.13 and 3.14, both solid lines separate the two identical sets of open circles and triangles, but the solid line in Figure 3.14 leaves the closest points at the maximum distance. The dashed lines in Figure 3.14 identify the margin.

Therefore, $1/w$ is the lower bound on the distance between the points x_i and the separating hyper-plane (w,b).

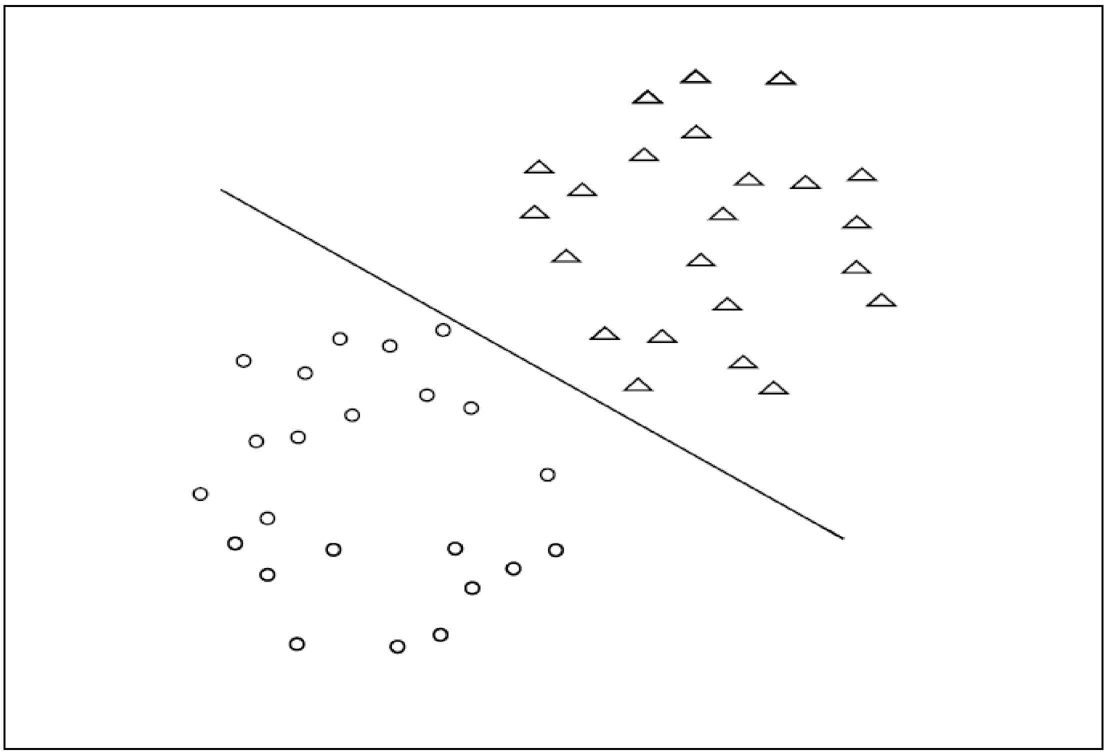


Figure 3.13 Separating hyper-plane.

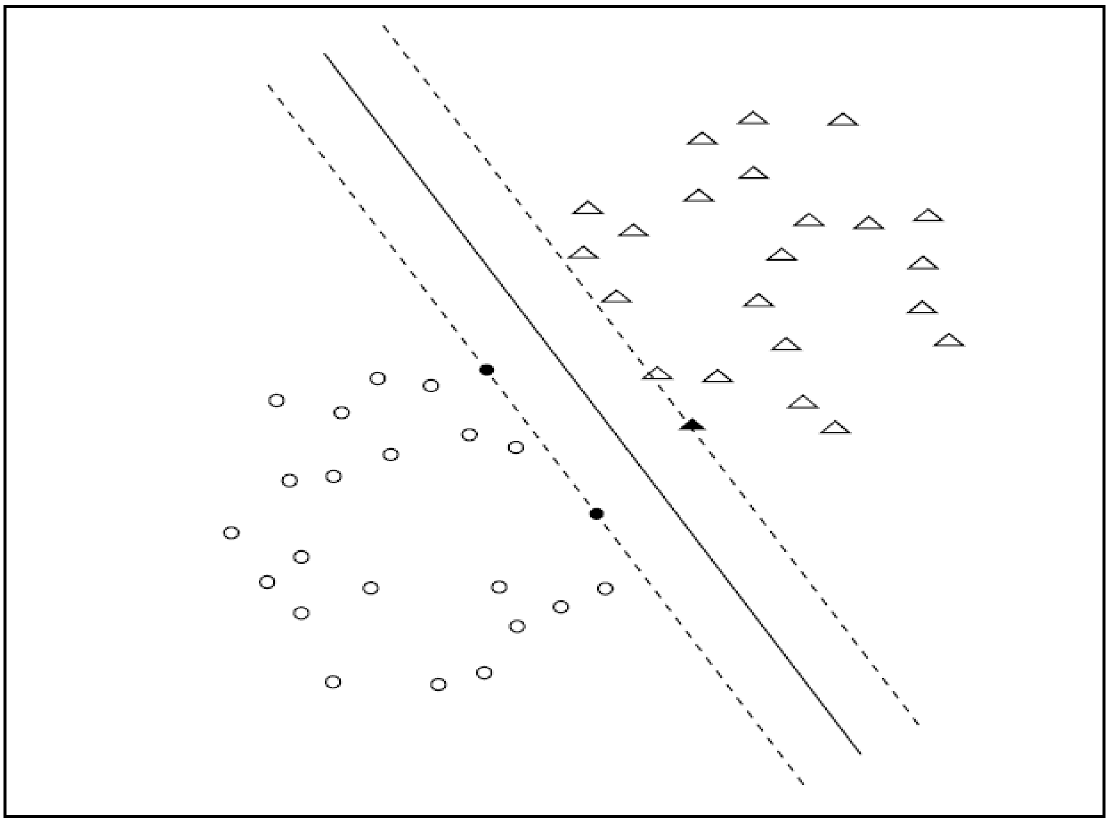


Figure 3.14 Optimal separating hyper-plane.

One might ask why not simply rewrite inequality in Eq.3.84 as shown Eq.3.88.

$$y_i(w \cdot x_i + b) \geq 0 \qquad \text{(Eq.3.88)}$$

The purpose of the "1" in the right hand side of inequality Eq.3.84 is to establish a one-to-one correspondence between separating hyper-planes and their parametric representation. This is done through the notation of canonical representation of a separating hyper-plane.

Definition: Given a separating hyper-plane (w, b) for the linearly separable set, S, the "canonical representation" of the separating hyper-plane is obtained by rescaling the pair (w, b) into the pair (w', b') in such a way that the distance of the closest point equals $1/w'$.

Through above definition, we have Eq.3.89.

$$\min_{x_i \in S}\{y_i(w' \cdot x_i + b')\} = 1 \qquad \text{(Eq.3.89)}$$

Consequently, for a separating hyper-plane in the canonical representation, the bound in Eq.3.87 is tight. In what follow we will assume that a separating hyper-plane is always given the canonical representation and thus write (w,b) instead of (w',b'). We are now in a position to define the notation of optimal separating hyper-plane.

Definition: Given a linearly separable set *S*, the "optimal separating hyper-plane" (OSH) is the separating hyper-plane which maximizes the distance of the closest point of *S*.

Since the distance of the closest point equals 1/*w*, the OSH can be regarded as the solution of the problem of maximizing 1/*w* subject to the constraint as Eq.3.84, or the problem as shown in Eq.3.90.

$$\begin{aligned} &\text{Minimize :} \quad \frac{1}{2}\mathrm{w} \cdot w \\ &\text{subject to:} \quad y_i(w \cdot x_i + b) \geq 1, \quad i = 1, 2, \ldots, N \end{aligned} \qquad \text{(Eq.3.90)}$$

Two comments are in order. First, if the pair (w,b) solves Eq.3.90, then for at least one $x_i \in S$ we have $y_i(w \cdot x_i + b) = 1$. In particular, this implies that the solution of Eq2.146 is always a separating hyper-plane in the canonical representation. Second, the parameter *b* enters in the constraints but not in the function to be minimized. The quantity 2/*w*, which measures the distance between the two classes in the direction of *w*, is named "margin". Hence, the OSH can also be seen as a separating hyper-plane which maximizes the margin as shown in Figure 3.14. We now study the properties of the solution of Eq.3.90.

3.2.5.2 Support Vectors

Eq.3.90 can be solved by means of the classical method of Lagrange multipliers (Bazaraa & Shetty, 1979). If we denote with $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)$ the N nonnegative Lagrange multipliers associated with the constraints as Eq.3.84, the solution to problem as Eq.3.90 is equivalent to determining the "saddle point" of Eq.3.91 with $L = L(w, b, \alpha)$.

$$L = \frac{1}{2} w \cdot w - \sum_{i=1}^{N} \alpha_i \{ y_i (w \cdot x_i + b) - 1 \} \qquad \text{(Eq.3.91)}$$

At the saddle point, L has a minimum for $w = \overline{w}$ and $b = \overline{b}$ and a maximum for $\alpha = \overline{\alpha}$, and thus we can write Eq.3.92 and Eq.3.93 with Eq.3.94.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i \alpha_i = 0 \qquad \text{(Eq.3.92)}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \qquad \text{(Eq.3.93)}$$

$$\frac{\partial L}{\partial w} = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ..., \frac{\partial L}{\partial w_n} \right) \qquad \text{(Eq.3.94)}$$

Substituting Eq.3.92 and Eq.3.93 into the right hand of Eq.3.91, we see that Eq.3.90 reduces to the maximization of Eq.3.95 subject to the constraint as Eq.3.92 with $\alpha \geq 0$. This new problem is called "dual problem" can be formulated as Eq.3.96.

$$\ell(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \qquad \text{(Eq.3.95)}$$

$$\text{Maximize} \quad -\frac{1}{2}\alpha \cdot D\alpha + \sum \alpha_i$$
$$\text{subject to} \quad \sum y_i \alpha_i = 0, \quad \alpha \geq 0 \qquad \text{(Eq.3.96)}$$

Both sums are for *i*=1, 2, ..., *N*, and *D* is an *N*x*N* matrix such that Eq.3.97 is satisfied.

$$D_{ij} = y_i y_j x_i \cdot x_j \qquad \text{(Eq.3.97)}$$

As for the pair $(\overline{w}, \overline{b})$, from Eq.3.96, it follows Eq.3.98 while $\overline{b}$ can be determined from the Kuhn-Tucker conditions as Eq.3.99.

$$\overline{w} = \sum_{i=1}^{N} \overline{\alpha}_i y_i x_i \qquad \text{(Eq.3.98)}$$

$$\overline{\alpha}_i (y_i (\overline{w} \cdot x_i + \overline{b}) - 1) = 0, \quad i = 1, 2, ..., N \qquad \text{(Eq.3.99)}$$

Note that the only $\overline{\alpha}_i$ that can be nonzero in Eq.3.99 are those for which the constraints as Eq.3.84 are satisfied with the inequality sign. The corresponding points x_i, termed "support vectors", are the points of *S* closest to the OSH. Given a support vector x_j, the parameter $\overline{b}$ can be obtained from the corresponding Kuhn-Tucker condition as Eq.3.100.

$$\overline{b} = y_j - \overline{w} \cdot x_j \qquad \text{(Eq.3.100)}$$

The problem of classifying a new data point x is now simply solved by computing as Eq.3.101.

$$\text{sign } (\overline{w} \cdot x + \overline{b}) \qquad \text{(Eq.3.101)}$$

In conclusion, the support vectors condense all the information contained in the training set *S* which is needed to classify new data points.

CHAPTER FOUR
TURKISH SYLLABLE *n*-GRAM ANALYSIS

4.1 Introduction

The designation of language properties is important for natural language processing (NLP) systems, such as automatic language detection, spell checking and suggestion mechanism for wrong written words and text summary generation systems. Furthermore, it is worthy in the area of cryptology, data compression, speech synthesis and speech recognition. It is difficult to construct a NLP system for Turkish language. Because some extra techniques are needed to overcome the difficulties arise from the language nature. In this manner, a new mechanism is suggested for parsing the Turkish words by syllables within this chapter.

TASA (Turkish Automatic Syllabifying Algorithm) is developed for hyphenation of Turkish words (Aşlıyan & Günel, 2005). The algorithm, which is coded with Matlab, is tested over 5 different corpora (Dalkılıç & Çebi, 2003). The test results show that the algorithm's error rate is 0% for the first 2000 Turkish words in each corpus. Then the total number of syllables and the number of syllables are calculated in Turkish for each corpus. The frequency of each syllable is analyzed. The syllable length distribution over corpora and the average syllable length are determined.

To extract the statistical information from the corpora, we use the Markov Model (Jurafsky & Martin, 2000), which is a probabilistic model. An *n*-th order Markov model looks (*n*-1)-th words into the past and calculates the probability of word sequences of the language. We apply this approach over the Turkish syllables. To obtain the *n*-gram probability (Jurafsky & Martin, 2003) from a corpus, Eq.4.1 can be used.

$$P(s_i \mid s_{i-n+1}) = \frac{C(s_{i-n+1},\ldots,s_i)}{C(s_{i-n+1},\ldots,s_{i-1})} \qquad \text{(Eq.4.1)}$$

Table 4.1 Turkish alphabet

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

As shown in Eq.4.1, $C(s_{i-n+1},...,s_i)$ is the number of occurrences of the syllable sequence $s_{i-n+1},...,s_i$ in a corpus.

Turkish language is a member of Altaic languages. The Turkish language consists of twenty nine different letters, which are given in the Table 4.1. The segments of speech uttered with a single impulse of air are called syllables. The Turkish words consist of a sequence of syllables. Morphologically, Turkish is an agglutinative language. Therefore, it is possible to generate thousands of forms from a given root word with suffixes, as Korean, Hungarian and Finnish.

An exaggerated and so popular example of a Turkish word formation (Oflazer & Bozşahin, 1994) is given as “Çekoslovakyalılaştıramadıklarımızdanmışsınızcasına”.

The word equivalent in English is “you were of those whom we could not convert to a Czechoslovakian”. In our example the root morpheme is Çekoslovakya, which is the name of the state of Czechoslovakia in English. The Syllabifying of the word is “Çe-kos-lo-vak-ya-lı-laş-tı-ra-ma-dık-la-rı-mız-dan-mış-sı-nız-ca-sı-na”.

In Turkish, the syllables have at least one and at most four letters, which are meaningless in themselves except some special cases, such as “bal”, “kol”, “dal”, “çal”, “kürk. All the possible combinations of Turkish syllable formations with some examples are given in Table 4.2, where C and V denote the consonant and vowel, respectively.

Some exceptional cases exist for the foreign words in Turkish, e.g. “tvist” imported from the English word “twist”, and for the words with accents, which alter the pronunciation of the syllables that contain them, such as “kâr”, “umumî” and “üslûp”, which mean “profit”, “general” and “style” in English, respectively. In the next section, a new algorithm, which achieves the segmentation of the Turkish words to their syllables capable of overcoming all special cases, has been given.

Table 4.2 The possible Turkish syllable structure

V	a e ı i o ö u ü
VC	ab, ac, aç, ad, ..., az, eb, ec, eç, ...
CV	ba, be, bı, bi, ..., za, ze, zı, zi, ...
CVC	bel, gel, köy, tır, ...
VCC	alt, üst, ırk, ...
CCV	bre, ...
CVCC	kurt, yurt, renk, Türk, ...

4.2 Design and Implementation of TASA

TASA is implemented with Matlab on Windows. As it is shown in Figure 4.1, TASA takes each word one by one from Turkish clean text corpus which is obtained by preprocessing the Turkish text corpus. In the preprocessing stage, punctuation marks are eliminated. Each letter is converted to lower case. There is only one blank between two words. After the preprocessing, TASA extracts the syllables from the word. The output of the TASA is Turkish Text Corpus which includes the syllables of the words.

TASA consists of two main parts as TASA-A and TASA-B as given in Figure 4.1. TASA-A divides the words into subword units. TASA-B extract syllables from subword units.

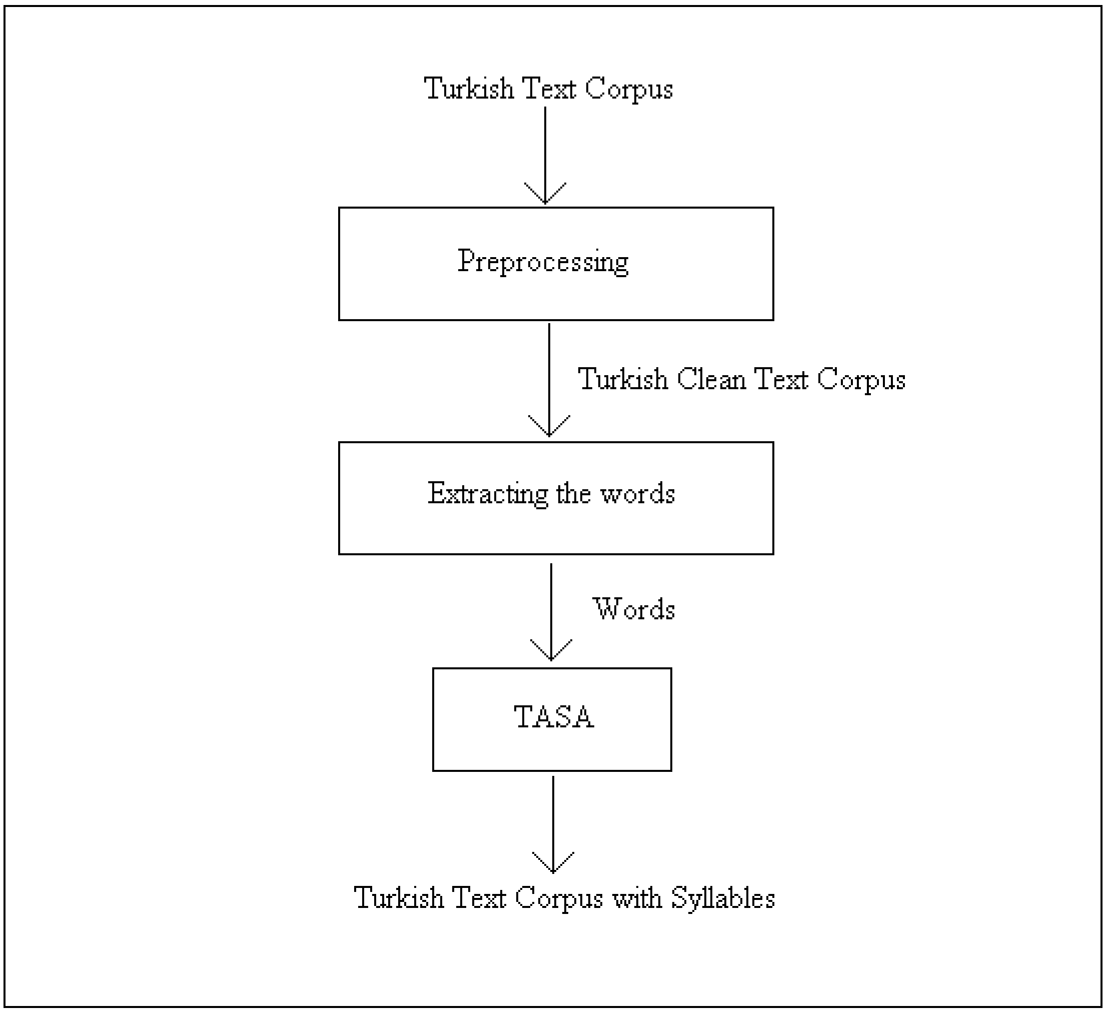


Figure 4.1 The Turkish syllable extracting system.

4.2.1 The Algorithm of TASA-A

The algoritm takes a word as an input and gives subword units as outputs. The algorithm incorporates four steps.

Step 1: If there are four consonants in the word side by side, and if these consonants are not at the beginning or end of the word, the word is divided into subword units after the second consonant.

Step 2: If there are three consonants in the word side by side, and if these consonants are not at the beginning or end of the word, the word is divided into subword units after the second consonant.

Step 3: If there are two consonants in the word side by side, and if these consonants are not at the beginning or end of the word, the word is divided into subword units after the first consonant.

Step 4: If there are two vowels in the word side by side, the word is divided into subword units after the first vowel.

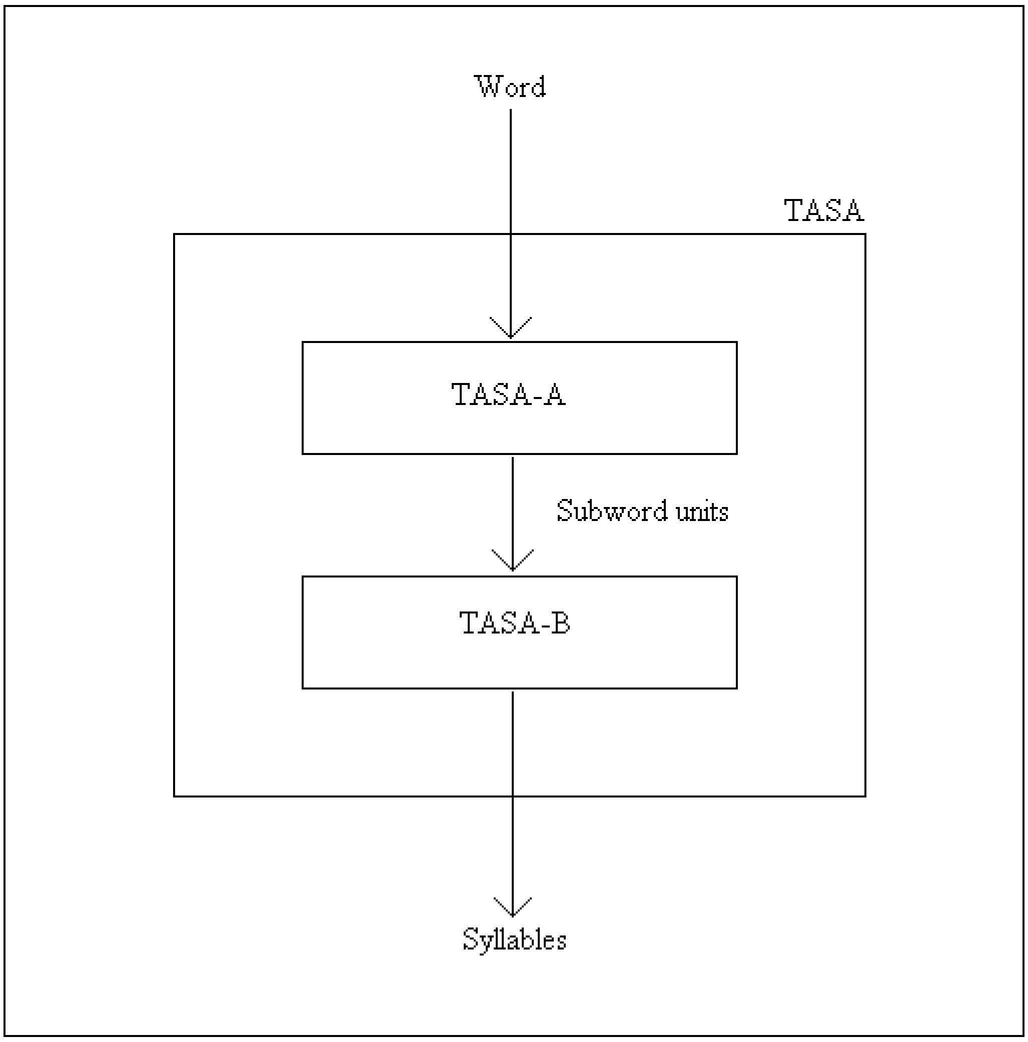


Figure 4.2 TASA structure.

4.2.2 The Algorithm of TASA-B

The algorithm takes a subword unit as an input and gives syllables of the subword unit as outputs. TASA-B consists of six cases according to the length of the subword unit.

```
**Case 1:  The length of the subword unit is 1**
If the subword unit has only one letter then,
  It is only a syllable.
**Case 2:  The length of the subword unit is 2**
If the subword unit has two letters then,
  It is only a syllable.
**Case 3:  The length of the subword unit is 3**
If the three letters of the subword unit are vowel, consonant and
vowel (VCV) respectively then,
  There are two syllables. The first vowel is the first syllable and
  the other two letters are the second syllable.
ElseIf vowel and consonant forms are VCC, CVC     and CCV then,
  The subword unit is only one syllable.
**Case 4:  The length of the subword unit is 4**
If the first two letters or the last two letters of the subword unit
are consonant then,
  The subword unit is a syllable.
Else
  If the first letter is vowel then,
    The first letter is the first syllable and the other three
    letters are the second syllable.
  Else
    The first two letters are the first syllable and the other two
    letters are the second syllable.
**Case 5:  The length of the subword unit is 5**
If the first or the last three letters of the subword unit are
consonants then,
  It is only a syllable.
ElseIf  the first and the last two letters of the subword unit are
consonants then,
  It is only a syllable.
Else
  If the first letter is vowel then,
    If the last two letters are consonants then,
      The first letter is the first syllable, and the other letters
      are the second syllable.
    Else
```

```
      The first letter is the first syllable. The next two letters
      are the second syllable. The last two letters are the third
      syllable.
  Else
    If the second letter of the subword unit is consonant then,
      The first three letters are the first syllable, and the other
      two letters are the second syllable.
    Else
      The first two letters are the first syllable, and the other
      three letters are the second syllable.
```

Case 6: The length of the subword unit is equal or greater than 6

```
If the first letter of the subword is vowel then,
  If the last three letters are consonant then,
    The first letter is a syllable. The next binary letters are
    syllables until the last five letters. The last five letters are
    a syllable.
  ElseIf the last two letters are consonant then,
    The first letter is a syllable. The next binary letters are
    syllables until the last four letters. The last four letters are
    a syllable.
  ElseIf the last letter is vowel then,
    The first letter is a syllable. The next binary letters are
    syllables.
  Else
    The first letter is a syllable. The next binary letters are
    syllables until the last three letters. The last three letters
    are a syllable.
Else
    If the second letter is vowel then,
      If the last three letters are consonant then,
        The binary letters from the beginning are syllables until the
        last five letters. The last five letters are a syllable.
      ElseIf the last two letters are consonant then,
        The binary letters from the beginning are syllables until the
        last four letters. The last four letters are a syllable.
      ElseIf the last letter is vowel then,
        The binary letters from the beginning are syllables.
      ElseIf the last two letters are vowel and consonant
       respectively then,
  The binary letters from the beginning are syllables until the
  last three letters. The last three letters are a syllable.
Else
  If the third letter is vowel then,
    If the last three letters are consonant then,
      The first three letters are a syllable. The next binary
      letters are syllables until the last five letters. The
      last five letters are a syllable.
    ElseIf the last two letters are consonant then,
      The first three letters are a syllable. The next binary
      letters are syllables until the last four letters. The
      last four letters are a syllable.
    ElseIf the last two letters are vowel and consonant
    respectively then,
      The first three letters are a syllable. The next binary
      letters are syllables until the last three letters. The
      last three letters are a syllable.
    ElseIf the last letter is vowel then,
      The first three letters are a syllable. The next binary
      letters are syllables.
    Else
      If the last three letters are consonant then,
        The first four letters are a syllable. The next binary
        letters are syllables until the last five letters. The
        last five letters are a syllable.
      ElseIf the last two letters are consonant then,
        The first four letters are a syllable. The next binary
        letters are syllables until the last four letters. The
        last four letters are a syllable.
      ElseIf the last two letters are vowel and consonant
      respectively then,
        The first four letters are a syllable. The next binary
        letters are syllables until the last three letters. The
        last three letters are a syllable.
      ElseIf the last letter is vowel then,
        The first four letters are a syllable. The next binary
        letters are syllables.
```

For example, the given word is converted into the subword units by TASA-A: “Çekos”, “lovak”, “yalılaş”, “tıramadık”, “larımız”, “dan”, “mış”, “sınız, “casına”. TASA-B takes these subword units and extracts the syllables. The syllables “Çe” and “kos” are extracted from the subunit “Çekos”. The syllables “lo” and “vak” are extracted from the subunit “lovak”. This process repeats for each subword units in the same way.

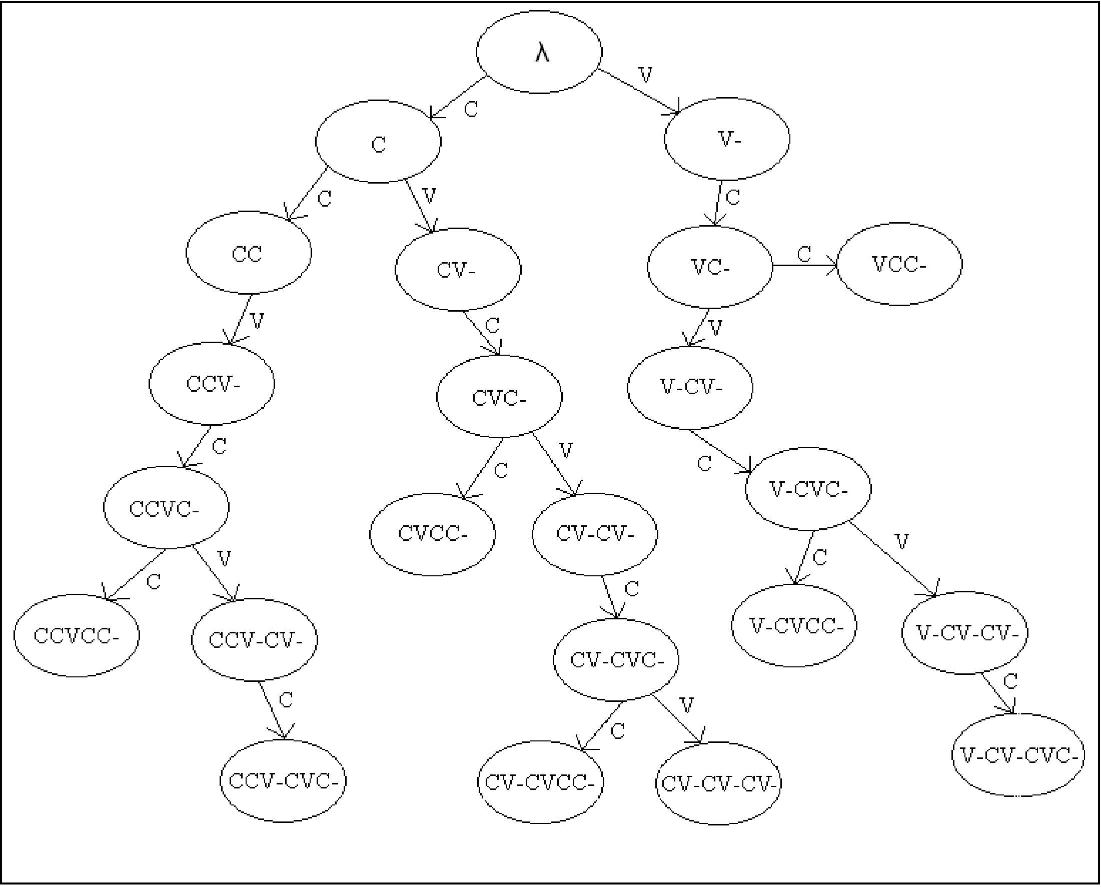


Figure 4.3 FSA structure of TASA-B.

Figure 4.3 represents the finite state automata (FSA) structure of TASA-B. C, V, λ and - is consonant, vowel, empty and syllable separator characters respectively. The finite state automata is a quintuple $M = (K, \Sigma, \Delta, s, F)$ where $\Sigma = \{\lambda, C, V, -\}$,

$K = \{\lambda, C, V-, CC, CV-, VC-, VCC-, CCV-, CVC-, V-CV-, CCVC-, CVCC-, CV-CV-, V-CVC-, CCVCC-, CCV-CV-, CV-CVC-, V-CVCC-, V-CV-CV-, CCV-CVC-, CV-CVCC-, CV-CV-CV-, V-CV-CVC-\}$, $s = \lambda$, $F = \{V-, CV-, VC-, VCC-, CCV-, CVC-, V-CV-, CCVC-, CVCC-, CV-CV-, V-CVC-, CCVCC-, CCV-CV-, CV-CVC-, V-CVCC-, V-CV-CV-, CCV-CVC-, CV-CVCC-, CV-CV-CV-, V-CV-CVC-\}$, and Δ is a finite subset of $K \mathrm{x} \Sigma^* \mathrm{x} K$. Σ^* is the set of strings from Σ.

4.3 Experimental Results

This system is composed of two stages. The first is to construct TASA, and the second is to constitute Turkish syllable statistics. After developing TASA, we extracted the syllables from the words in the five Turkish corpora. We tested TASA for the first 2000 words of each corpus, and we have encountered no words, which were spelt wrongly.

We computed the syllable length for each corpus. Syllable length is the number of letters in the syllable. Table 4.3 shows that the rate of syllable length of every corpus is approximately the same. Generally, the rate of the syllables which have two letters is 56.57% over all syllables of the corpora. The rate of the syllables which have three letters is 35.16%. The rates of the syllables which have a letter, four letters and five letters are 5.93%, 2.18% and 0.17% respectively. There are no syllables whose length is greater than five in each corpus. Table 4.4 indicates the statistics of Turkish syllable. For İmla Corpus there are 3419 different syllables.

There are a lot of studies on word *n*-grams. But we modeled syllable *n*-grams for Turkish. We computed the monogram and bigram of the corpora as shown in Table 4.5. For instance, the syllable “la” has the highest frequency. In other words, The syllable “la” is 2.4% of the syllables in this corpus. We accepted the blank between two words like a syllable. According to the Table 4.5, the probability that the syllable “a” is following the blank is the highest (1.7%) for Bilim Corpus.

We have designed and implemented TASA for the five Turkish Corpora. After testing operation, it is calculated that TASA's correct spelling rate is about 100%. After analyzing the Turkish Corpora, we have found that the rate of the syllables which have two letters is 57% over all syllables of the corpora. It is also computed the monograms and bigrams of the Turkish Corpora. TASA can be used any other Turkish Corpora, and more detailed researches can be done. Turkish syllable structure can be extracted. New compression algorithms can be generated using syllables. Furthermore, Turkish syllables can be applied to speech synthesis.

Table 4.3 The syllable length of Turkish corpora

Corpus	Syllables				
	One letter	**Two letters**	**Three letters**	**Four letters**	**Five letters**
İmla	5238	74796	72258	3020	99
Bilim	34892	304097	186073	9713	908
Pc Mag.	77740	765101	466650	39740	3338
Yeni Asır	13620	153207	92600	2881	49
Ulusal Pr.	33759	275598	160008	5157	261
Sum	164849	1572799	977589	60511	4655

Table 4.4 Statistics of some Turkish corpora

Corpus	The number of different syllables	The number of syllables	The number of words
İmla	3419	155411	48350
Bilim	3515	535683	201605
Pc Mag.	6542	1352169	515904
Yeni Asır	2324	262357	96703
Ulusal Pr.	2048	474783	158945
Sum		2780403	1021687

Table 4.5 Statistics of Bilim corpora

Monogram	Frequency	%	Bigram	Frequency	%
la	12808	2.4	blank a	9197	1.7
le	10931	2.0	da blank	5821	1.1
a	9976	1.8	blank o	5343	1.0
de	9814	1.8	de blank	5185	1.0
da	8751	1.6	blank i	5178	1.0
Sum		9.6	**Sum**		5.8

CHAPTER FIVE
DETECTING MISSPELLED WORDS IN TURKISH TEXT USING SYLLABLE *n*-GRAM FREQUENCIES

5.1 Introduction

To detect misspelled words in a text is an old problem. Today, most of word processors include some sort of misspelled word detection. Misspelled word detection is worthy in the area of cryptology, data compression, speech synthesis, speech recognition and optical character recognition (Barari & QasemiZadeh, 2005; Deorowicz & Ciura, 2005; Kang & Woo, 2001; Tong & Evans, 1996). The traditional way of detecting misspelled words is to use a word list, usually also containing some grammatical information, and to look up every word in the word list (Kukich, 1992) from dictionary.

The main disadvantage of this approach is that if the dictionary is not large enough, the algorithm will report some of correct words as misspelled, because they are not included in the dictionary. For most natural languages, the size of dictionary needed is too large to fit in the working memory of an ordinary computer. In Turkish this is a big problem, because Turkish is an agglutinative language and too many new words can be constructed by adding suffixes.

To overcome this difficulties, a new approach has been proposed for detecting misspelled words in Turkish text. We have used Turkish syllable *n*-gram frequencies which are generated from several Turkish corpora (Dalkılıc & Cebi, 2003). From the corpora we have extracted syllable monogram, bigram and trigram frequencies using TASA (see Chapter 4) (Aşlıyan & Günel, 2005). We have used these *n*-gram frequencies for calculating a word probability distribution. After that the system has decided whether a word is misspelled or not. In this approach we don't need word list. We have only Turkish syllables and their monogram, bigram and trigram frequencies. The chapter is organized as follows. In Section 5.2, we described the system architecture.We explained how syllable *n*-gram frequencies and word probability distribution are computed in Section 5.3. We discussed the empirical results of the system in Section 5.4.

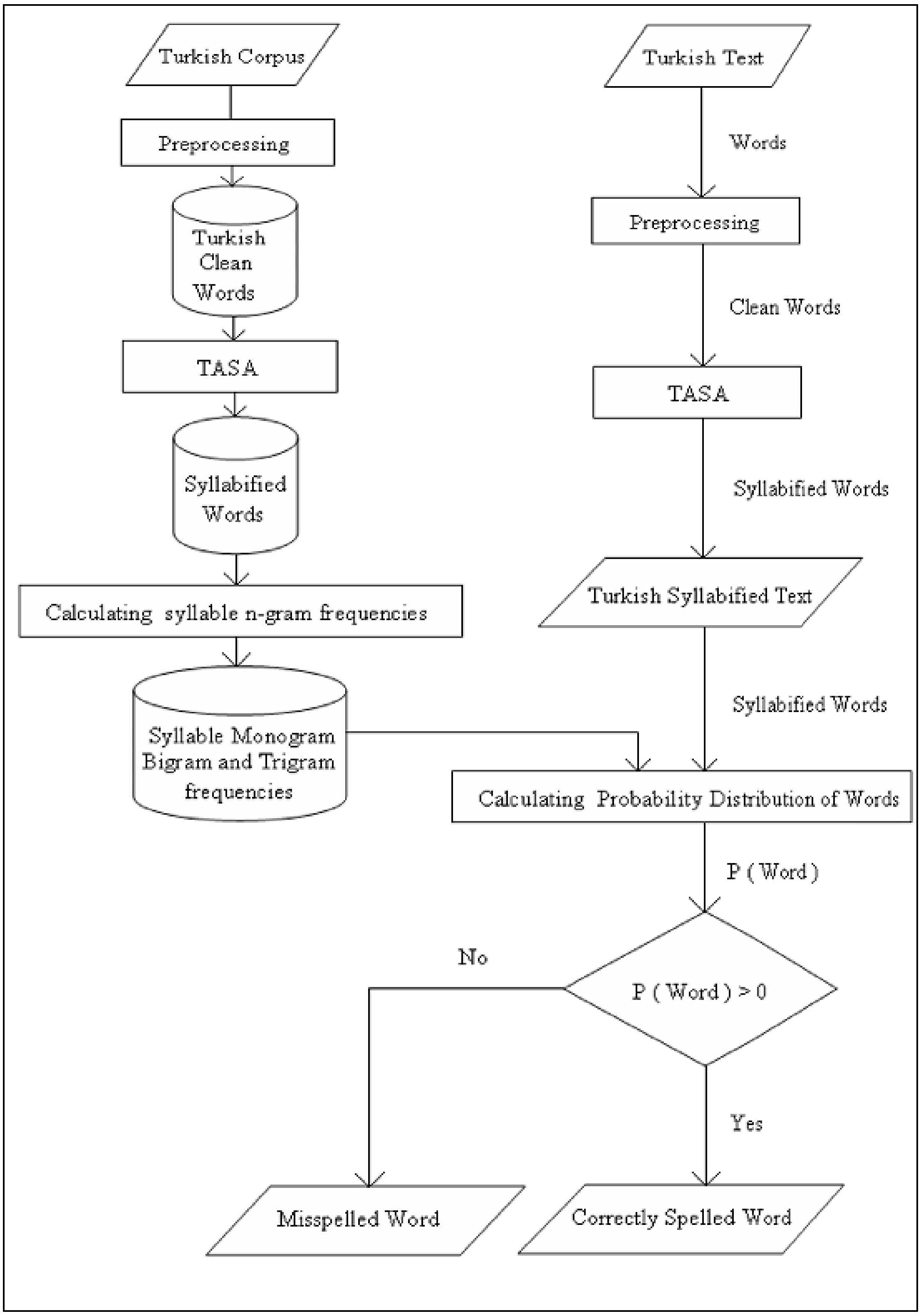


Figure 5.1 System architecture.

5.2 System Architecture

The system consists of three main components. First component is preprocessing which cleans a text. Second component is TASA, and third component is calculating probability distribution of words. As shown in Figure 5.1, the system takes words in Turkish text as input and gives the result for each word as “Misspelled Word” or “Correctly Spelled Word”.

In preprocessing component of the system, punctuation marks are cleaned. All letters in the text are converted to lower case. Blank characters between two successive words are limited with only one blank character.

In second component, TASA takes the Turkish clean text as an input and gives the Turkish syllabified text. The system divides words into syllables putting the dash character between two syllables. For example, the word “kitaplık” in Turkish text is converted into the syllabified word “ki-tap-lık” in Turkish syllabified text.

In third component, the probability distribution is calculated for each syllabified word. The system uses syllable monogram, bigram and trigram frequencies to find this probability distribution. How these *n*-gram frequencies are computed is explained in detail in the following Subsection 5.2.1.

5.2.1 Calculation of Syllable n-gram Frequencies

We have used the Turkish corpora (Dalkılıc & Cebi, 2003) which includes 304178 Turkish words and the corpora is preprocessed as seen in Figure 5.1. The system TASA syllabifies all Turkish words in the corpora.We have constructed Turkish syllable corpora from the Turkish word corpora. Turkish syllable corpora contains 900342 Turkish syllables. As shown in Table 5.1, Table 5.2 and Table 5.3, Turkish syllable monogram, bigram and trigram frequencies are calculated. For example, the frequency of the syllable monogram“la” is 21322. In Table 5.2 and

Table 5.3, “blank” represents only one blank character. We accepted blank character as syllable for the system. Table 5.2 shows the frequencies of some Turkish syllable bigram.

Table 5.1 Monogram statistics for Turkish corpus

Monogram	Frequency	%
la	21322	2.37
ma	17704	1.97
li	15124	1.68
a	13439	1.49
ta	13372	1.48
i	12827	1.42
de	11699	1.30
ra	11611	1.29
da	10930	1.21
ve	10570	1.17
ri	9618	1.07
rı	9105	1.01
me	8776	0.97
e	7312	0.81
ec	5909	0.66

Table 5.2 Bigram statistics for Turkish corpus

Bigram	Frequency	%
blank i	12880	1.43
blank a	11194	1.24
blank ve	9793	1.09
blank e	9601	1.07
li blank	9410	1.04
ve blank	8803	0.98
blank ta	8296	0.92
blank ka	7907	0.88
da blank	7429	0.82
ec blank	5857	0.65
la rı	5659	0.63
le ri	5551	0.62

Table 5.3 Trigram statistics for Turkish corpus

Trigram	Frequency	%
blank ve blank	5866	0.05
blank ta rih	5238	0.03
ta rih li	4944	0.03
rih li blank	4944	0.03
blank i liş	3437	0.02
i liş kin	3282	0.02
blank i le	3037	0.02
blank bir blank	3000	0.02
i le blank	2997	0.02
la rı blank	2979	0.02
ler ri blank	2905	0.02
blank kon sey	2896	0.02
kon sey blank	2895	0.02

5.3 Calculation of the Probability Distribution of Words

An *n*-gram is a sub-sequence of n items from a given sequence. *n*-grams are used in various areas of statistical natural language processing and genetic sequence analysis. The items in question can be letters, syllables, words according to the application.

An *n*-gram of size 1 is a "monogram"; size 2 is a "bigram"; size 3 is a "trigram"; and size 4 or more is simply called an "*n*-gram" or "(*n*−1)-order Markov model" (Zhuang et al., 2004).

An *n*-gram model predicts x_i based on $x_{i-1}, x_{i-2}, x_{i-3}, ..., x_{i-n}$. When used for language modeling independence assumptions are made so that each word depends only on the last *n* words. This Markov model is used as an approximation of the true underlying language. This assumption is important because it massively simplifies the problem of learning the language model from data.

Suppose that a word *W* in Turkish syllabified text consists of the syllable sequence $s_1, s_2, s_3, ..., s_t$. This word has *t* syllables. To obtain the *n*-gram probability distribution (Jurafsky & Martin, 2000) of the word *W*, we have used in Eq.5.1.

$$P(W) = P(s_1, s_2, ..., s_t) = \prod_{i=1}^{t} P(s_i \mid s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}) \qquad \text{(Eq.5.1)}$$

In *n*-gram model, the parameter $P(s_i \mid s_{i-n+1}, s_{i-n+2}, ..., s_{i-1})$ in Eq.5.1 can be estimated with Maximum Likelihood Estimation (MLE) (Aşlıyan & Günel, 2005) technique as shown in Eq.5.2.

$$P(s_i \mid s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}) = \frac{C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}, s_i)}{C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1})} \qquad \text{(Eq.5.2)}$$

So, we conclude as Eq.5.3.

$$P(W) = P(s_1, s_2, ..., s_t) = \prod_{i=1}^{t} \frac{C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}, s_i)}{C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1})} \qquad \text{(Eq.5.3)}$$

In Eq.5.2 and Eq.5.3, $C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}, s_i)$ is the frequency of the syllable sequence $s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}, s_i$. Furthermore, $C(s_{i-n+1}, s_{i-n+2}, ..., s_{i-1})$ is the frequency of the syllable sequence $s_{i-n+1}, s_{i-n+2}, ..., s_{i-1}$. The frequencies of these syllable sequences can be calculated from the Turkish corpora.

For bigram(n=2) and trigram(n=3) models, probability distribution $P(W)$ can be computed as shown in Eq.5.4 and Eq.5.5 respectively.

$$P(W) = P(s_1, s_2, ..., s_t) = \prod_{i=1}^{t} P(s_i \mid s_{i-1}) = \prod_{i=1}^{t} \frac{C(s_{i-1}, s_i)}{C(s_{i-1})} \qquad \text{(Eq.5.4)}$$

$$P(W) = P(s_1, s_2, ..., s_t) = \prod_{i=1}^{t} P(s_i \mid s_{i-2}, s_{i-1}) = \prod_{i=1}^{t} \frac{C(s_{i-2}, s_{i-1}, s_i)}{C(s_{i-2}, s_{i-1})} \qquad \text{(Eq.5.5)}$$

For example, according to bigram model we can calculate the probability distribution of a word in Turkish text. Assume that we have a text which includes some words as “... Bu gün okulda, şenlik var...”. This text is converted to syllabified text as “... Bu gün o-kul-da, şen-lik var...”. Syllables in the words are delimited with dash character. Assume that the word *W*=”okulda” in the text is taken for computing its probability distribution and *W* can be written as the syllable sequence $W = s_1, s_2, s_3 =$”o”, “kul”, “da”. Here, $s_1 =$“o”, $s_2 =$“kul”, $s_3 =$“da”. We accepted blank character as a syllable.We call this syllable as λ. So, assume that syllable monogram frequencies are *C*(“λ”)=0.003, *C*(“o”)=0.002, *C*(“kul”)=0.004 and syllable bigram frequencies are *C*(“λ”,“o”)=0.0001, *C*(“o”,“kul”)=0.0002, *C*(“kul”,“da”)=0.0003. We have calculated P(“okulda”) using bigram model. We have found that the probability distribution of the word “okulda” is 0.0002475 as shown in Eq.5.6.

$$\begin{aligned} P(W) = P(\text{"okulda"}) = P(s_1, s_2, s_3) &= P(\text{"o"}, \text{"kul"}, \text{"da"}) \\ &= \prod_{i=1}^{3} P(s_i \mid s_{i-1}) = \prod_{i=1}^{3} \frac{C(s_{i-1} \mid s_i)}{C(s_{i-1})} \qquad \text{(Eq.5.6)} \\ &= \left(\frac{C(\text{"}\lambda\text{"}, \text{"o"})}{C(\text{"}\lambda\text{"})} \right) \left(\frac{C(\text{"o"}, \text{"kul"})}{C(\text{"o"})} \right) \left(\frac{C(\text{"kul"}, \text{"da"})}{C(\text{"kul"})} \right) \end{aligned}$$

5.4 Testing the System

We have designed and implemented two systems to detect misspelled words in Turkish text. One uses monogram and bigram frequencies. The size of monogram database is 41 kilobytes and our monogram consists of 4141 different syllables. The size of bigram and trigram databases are 570 and 2858 kilobytes respectively. While the bigram database includes 46684 syllable pairs, the trigram database consists of 183529 ternary syllables. The other uses bigram and trigram frequencies. We have tested these two systems. To test the systems, we have two Turkish texts. One is correctly spelled text which includes 685 correctly spelled words. The other is misspelled text which has 685 misspelled words. These two texts have same words. Namely, misspelled words are generated with putting errors on the correctly spelled words. These error types are substitution, deletion, insertion, transposition and split word errors. The systems takes correctly spelled and misspelled texts as input and gives the results for each word as “correctly spelled word” or “misspelled word”. As it is shown in Figure 5.1, probability distributions are calculated for each word. If the probability distribution of a word is equal to zero, system decides that the word is misspelled. If it is greater than zero, system decides that the word is correctly spelled.

The system works with Intel-based NT, Windows 2000, XP, Windows 2003 Server systems with 512MB RAM and it has been developed using Borland C++ Builder Professional.

We have first tested the system on the correctly spelled text using monogram and bigram frequencies. The system determines 602 correctly spelled words from the correctly spelled text, so the words are correctly recognized with 88% success rate. Also, 589 misspelled words within the misspelled text are decided successfully by the system. Namely, the system which is tested on the misspelled text correctly recognized the words with 86% success rate.

Finally we have tested the system on the correctly spelled text using bigram and trigram frequencies. The system determines 671 of 685 correctly spelled words from the correctly spelled text. The success rate on correctly recognition of the words is 98%. Furthermore, 664 of 685 misspelled words within the misspelled text are decided successfully by the system. Thus, the system which is tested on the misspelled text correctly recognized the words with 97% success rate. The system can analyze 75000 words per second.

In conclusion, we have designed and implemented a system which decides whether or not a word is misspelled in Turkish text. Firstly, three databases of syllable monogram, bigram and trigram frequencies are constructed using the syllables that are derived from five different Turkish corpora. Then, the system takes words in Turkish text as an input and computes the probability distribution of words using syllable monogram, bigram and trigram frequencies from the databases. If the probability distribution of a word is zero, it is decided that this word is misspelled. For testing the system, we have constructed two text databases with the same words. One text database has 685 misspelled words. The other has 685 correctly spelled words. The words from these text databases are taken as inputs for the system. The system produces two results for each word: “Correctly spelled word” or “Misspelled word”. The system that is designed with monogram and bigram frequencies has 86% success rate for the misspelled words and has 88% success rate for the correctly spelled words. According to the system designed with bigram and trigram frequencies, there is 97% success rate for the misspelled words and there is 98% success rate for the correctly spelled words.

CHAPTER SIX
SPEECH RECOGNITION EXPERIMENTS

In this thesis, we have designed and implemented speaker dependent isolated word speech recognition systems using the methods as LTA, DTW, ANN, HMM and SVM. In the applications, we have used the speech signal features as mfcc, lpc, parcor, cepstrum, rasta and the mixture of mfcc, lpc and parcor.

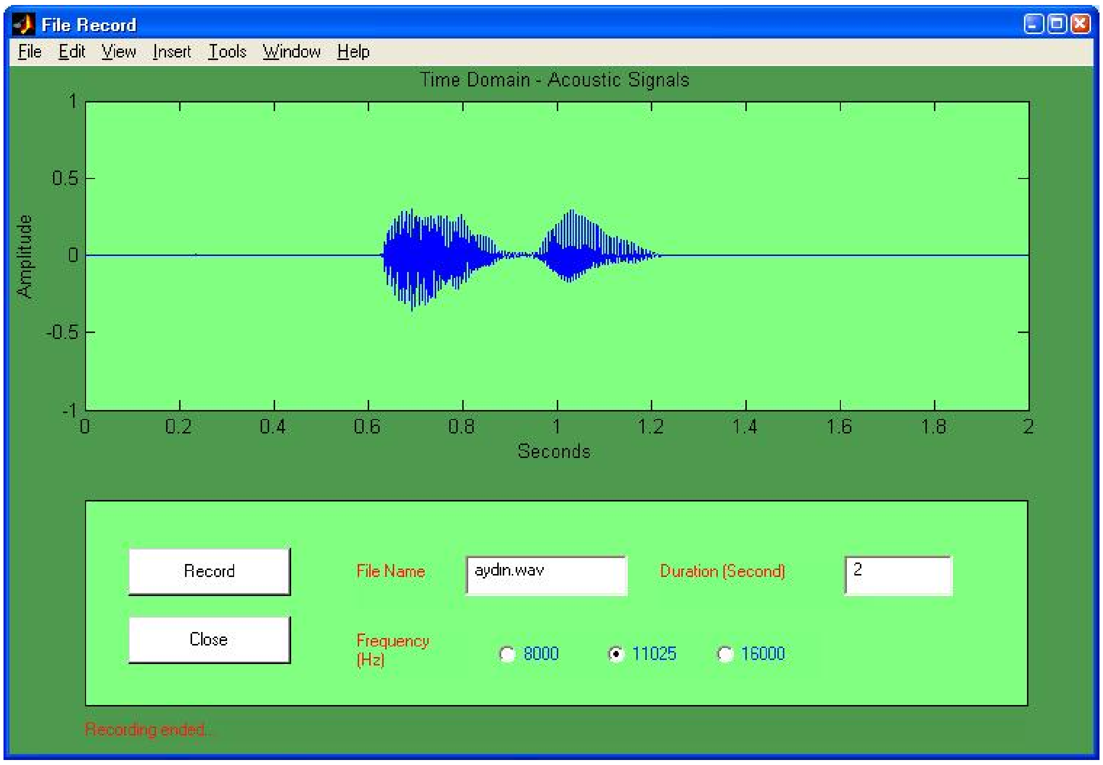
The speech recognition applications have been executed on the computer which has the following features: Pentium Centrino 1.6 CPU, 768 MB RAM, 40 GB harddisk, Windows XP Operating System, a sound card and a microphone. The codes of the applications have been written with Matlab 6.5.

6.1 System Databases

System dictionary consists of 200 different Turkish words which are shown in Appendix A. Using this dictionary we have constructed two databases. One database has been used for training and the other is for testing of the system.

The training speech database (approximately 2.7 hours of 250 MB speech material) involves 5000 Turkish word utterences (25x200) which were recorded by a male speaker. Each word in the dictionary was recorded 25 times using the recording program as shown in Figure 6.1.

The testing speech database was constructed by recording every word in the dictionary 10 times. Total number of utterences is 2000 (about 1.1 hours of 100 MB speech material).



File Name: aydin.wav

Duration (Second): 2

Frequency (Hz): 8000 [✓ 11025] 16000

Recording ended...

Figure 6.1 The user interface of wave file recording program.

The recording procedure took place in a noise-free environment. A head-mounted close-talking microphone was used. The format of the file recording is WAVE file format. The waveforms of the utterences are encoded using Pulse Code Modulation (PCM) coding format, 11025 Hz sampling rate, 2 bytes per sample. The utterences are recorded in 2 seconds time duration.

6.2 Preprocessing of the System

After the digitization of the word speech signal, we have applied preemphasis filter to spectrally flatten the signal as explained in Section 2.5. For the speech signal the syllable end-point detection is applied as explained in Subsection 6.2.1. After that each syllable utterence is divided into frames of 20 ms by frameblocking. To reduce the signal discontinuity at the ends of each block, Hamming window is applied for each frame as mentioned in Section 2.5.

6.2.1 Word and Syllable End-point Detection

An important problem in speech processing is to detect the presence of speech in a background of noise. This problem is often referred to as the end-point location problem (Rabiner & Sambur, 1975). The accurate detection of a word's start and end-points means that subsequent processing of the data can be kept to a minimum.

A major cause of errors in isolated-word automatic speech recognition systems is the inaccurate detection of the beginning and ending boundaries of test and reference patterns (Junqua, Mak & Reaves, 1997). It is essential for automatic speech recognition algorithms that speech segments are reliably separated from non-speech.

The reason for requiring an effective end-point algorithm is that the computation for processing the speech is minimum when the end-points are accurately located (Savoji, 1989).

In syllable end-point detection operation, the speech signals are taken and after processing them, the number of syllables and the indexes of syllable end-points have been detected. Namely, the beginning and end indexes are computed on the digital speech signal.

After sampling the sound wav files, the mean of the speech signal as a vector is calculated and translated to $y = 0$ axis. Assume that y_n is a speech signal. The new speech signal, which is focused on $y = 0$ axis, is $y'_n = y_n - mean(y_n)$. After that, the voiced and unvoiced parts of the speech signal are approximately computed with the slope between the beginning value of the digital sound and the maximum value of the sound. This slope is the threshold slope. The utterence is divided into windows which have 350 samples. If the slope, which is calculated between two windows, which are one after the other, is greater than the threshold slope, this means that the voiced part of the sound begins at the index. However, these beginning and end index of the voiced part are nearly true, but not certain value. We have used the distance data of zero-crossing index of sound vector because of obtaining more accurate results. We have constructed a new vector which represents the zero-crossing distances, then we have defined a threshold (say zero-crossing threshold=100). The beginning index is found earlier but not certainly true index. Now this index goes on one by one to the first index if the zero-crossing distance is between 1 and zero-crossing threshold. Zero values of the vector are not taken into account. In the same way, the end index of the voiced part is calculated. At the end, we find the voiced part exactly.

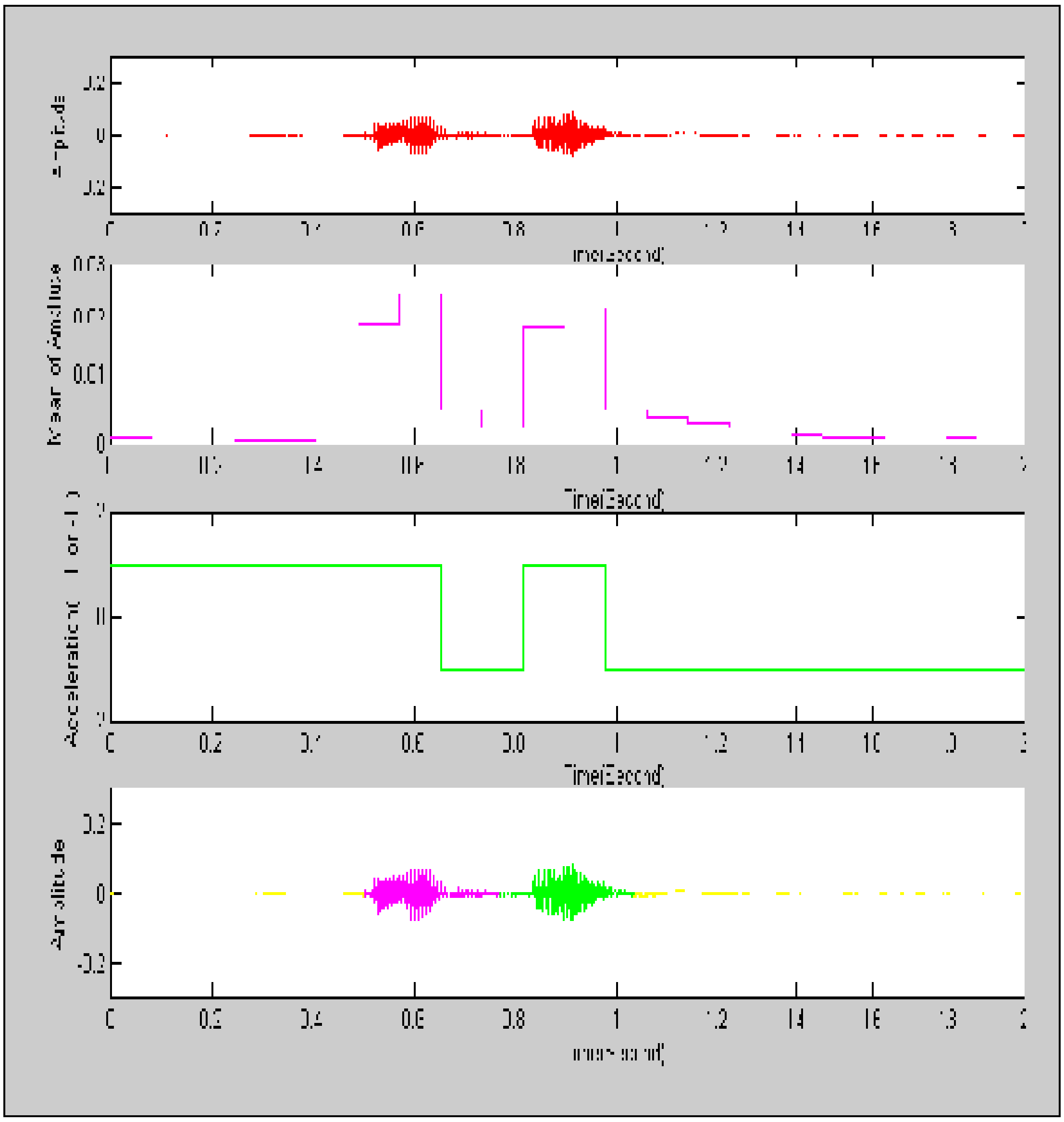


Figure 6.2 The process of syllable endpoint detection.

To discover syllable end-points, the windows that consist of 900 samples are generated without overlapping. The mean values of these windows are computed and assembled for constructing a new mean vector as shown in Figure 6.2. The slope between one element and the next element of the mean vector is determined, and if the slope is zero or greater than zero, a new vector's value is +1. Otherwise, the value is -1. Using these vectors, the boundaries of syllables on the sound vector are obtained approximately. The samples between 500 samples backward and 500 samples forward from the found syllable end-points are divided into windows which include 20 samples. After that, the middle index of the window which has the minimum mean is syllable end-point. Finally, we can calculate the beginning and end index of the syllables for each word before processing them. Now we have the number of syllables of the word and their end-point indexes.

According to the number of syllable in a word using syllable end-point detection algorithm which is mentioned in the Subsection 6.2.1.2., we have found that accuracy result is approximately 99%. For example, the word which has five syllables is successfully divided into five syllables and the end-points of the syllables are detected.

6.2.1.1 Word End-point Detection Algorithm

1. x in Eq.6.1 is digital sound vector. $N = 22050$ (N is the number of samples in the utterence)

$$x = (x_1, x_2, x_3 \ldots, x_N) \qquad \text{(Eq.6.1)}$$

2. λ is the mean of the values of first 200 samples in x. $\widetilde{x}$ is a vector which translated to the axis $y = 0$.

$$\lambda = (\sum_{i=1}^{200} x_i) / 200 \qquad \text{(Eq.6.2)}$$

$$\tilde{x} = x - \lambda = (x_1 - \lambda, ..., x_N - \lambda) = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_N) \qquad \text{(Eq.6.3)}$$

3. M is the maximum value of the vector $\tilde{x}$. I is the index of maximum value of the vector $\tilde{x}$. E_b and E_s are the beginning and end threshold values respectively.

$$[M, I] = \max(\tilde{x}) \qquad \text{(Eq.6.4)}$$

$$E_b = M / I, \qquad E_s = M / (N - I) \qquad \text{(Eq.6.5)}$$

4. The vector $\tilde{x}$ is divided into windows which consist of 350 samples. The vector $\bar{x}$ is the mean vector of above windows.

$$\bar{x} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_p) \text{ and } p = N / 350 \qquad \text{(Eq.6.6)}$$

$$\bar{x}_i = \left(\sum_{k=i*350}^{(i+1)*350-1} \tilde{x}_k \right) / 350 \ , \quad i = 1,2,...,p \qquad \text{(Eq.6.7)}$$

5. For $i = 1,2,...,p-1$, $\bar{x}_E$ and $\bar{x}_{E_i}$ are calculated as shown in Eq.6.8.

$$\bar{x}_E = (\bar{x}_{E_1}, \bar{x}_{E_2}, ..., \bar{x}_{E_{p-1}}) \text{ and } \bar{x}_{E_i} = \bar{x}_{i+1} / \bar{x}_i \qquad \text{(Eq.6.8)}$$

6. S_b is the beginning index of the sound vector.

```
For $r = 1$ to $p - 1$
    if $\bar{x}_{E_r} > E_b$ then $S_b = r * 350$
End
```

7. S_s is the end index of the sound vector.

```
For r = $x_{E_{p-1}}$ DownTo 1
  if $1/\bar{x}_{E_r} > E_s$ then $S_s = r * 350$
End
```

8. The beginning and end indexes are approximately determined from Step 6 and 7. To decide exactly the end-points of the sound, the zero-crossing indexes are fixed. Using the sound vector $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_N)$, the zero-crossing vector $z = (z_1, z_2, ..., z_{N-1})$ is generated.

```
For $k = 2$ To $N$
  if $\tilde{x}_{k-1} / \tilde{x}_k < 0$ then
    $z_{k-1} = 1$
  else
    $z_{k-1} = 0$
End
```

9. After the distances between one after the other zero-crossing indexes are computed, new distance vector of zero-crossing $\tilde{z}_k = (\tilde{z}_1, \tilde{z}_2, ..., \tilde{z}_{N-1})$ is calculated as the followings.

```
For $k = 1$ To $N - 1$
  if $z_k = 1$ and after the index k, its first value of the following indexes is 1,
    ($z_h = 1$) then
    $\tilde{z}_k = h - k$
  else
    $\tilde{z}_k = 0$
    if $z_k = 0$ then $\tilde{z}_k = 0$
```

```
End
```

10. The threshold value of zero-crossing is accepted as $T = 100$. SB is the value at the index which the sound begins.

```
$SB = S_b$
For $k = S_b$ DownTo 1
    if $\tilde{z}_k > 0$ and $\tilde{z}_k < T$ then $SB = k$
    if $\tilde{z}_k = 0$ then continue
    if $\tilde{z}_k > T$ then break
End
```

11. SS is the value at the index which the sound ends.

```
$SS = S_s$
For $k = S_s$ To $N-1$
    if $\tilde{z}_k > 0$ and $\tilde{z}_k < T$ then $SS = k$
    if $\tilde{z}_k = 0$ then continue
    if $\tilde{z}_k > T$ then break
End
```

6.2.1.2 Syllables End-point Detection of the Words

After detecting the end-points (SB and SS) of the words, the end-points of the syllables in the words are determined with the following algorithm.

1. $n = (n_1, n_2, ..., n_k) = (\tilde{x}_{SB}, \tilde{x}_{SB+1}, ..., \tilde{x}_{SS})$

2. The vector n is divided into windows, which have 900 samples, without overlapping. The vector $\overline{n}$ is the mean vector of each window above.

$$\overline{n} = (\overline{n}_1, \overline{n}_2, ..., \overline{n}_p) \text{ and } p = k/900$$

$$\overline{n}_i = \left(\sum_{m=i*900}^{(i+1)*900-1} n_m \right) / 900, \quad i = 1,2,...p \qquad \text{(Eq.6.9)}$$

3. The slope vector is composed by computing the slopes between the values of the vector $\overline{n}$ which follow one after another.

$$\overline{n}_E = (\overline{n}_{E_1}, \overline{n}_{E_2}, ..., \overline{n}_{E_{p-1}}) \text{ and } \overline{n}_{E_i} = \overline{n}_{i+1} / \overline{n}_i, \ i = 1,2,..., p-1 \qquad \text{(Eq.6.10)}$$

4. Using the slope vector, we calculate the vector $a = (a_1, a_2, ..., a_{p-1})$ which has the values, +1 and -1. Namely, the increasing and decreasing positions are determined.

```
For $k = 1$ To $p-1$
    if $n_{E_k} \geq 0$ then $a_k = 1$
    else $a_k = -1$
End
```

5. H is the number of syllables in the word.

```
$H = 0$
    For $k = 2$ To $p-1$
    if $a_{k-1} = 1$ and $a_k = -1$ then $H = H + 1$
End
```

6. The values of the middle indexes, which include the value -1 in the vector a, are approximately the end-points of syllables. There are $H-1$ syllable end-points. The syllable end-point vector $s = (s_1, s_2, ..., s_{H-1})$ is calculated as

shown in the following. The values s_i are the indexes which are the values of the vector $\tilde{x}$.

```
For $k = 1$ To $H - 1$
    if the middle index of the indexes, which have the $k$-th value -1 that
    follows one after the other, is $w$ then
        $s_k = SB + 900 * w$
End
```

7. Until now, the beginning point SB and end point SS is detected. The vector s represents the end-points of syllables. To find the syllable end-points more accurately the following algorithm is performed, and the vector $\tilde{s} = (\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_{H+1})$ is attained.

```
$\tilde{s}_1 = SB$ and $\tilde{s}_{H+1} = SS$
For $i = 1$ To $H - 1$
    The windows with 20 samples between $s_i - 500$ and $s_i + 500$ are
    constructed, and the mean values are computed for each windows.
    if the middle index of the window, which has the smallest mean, is $q$
    then
        $\tilde{s}_{i+1} = q$
End
```

8. The vector $\tilde{s}$ which represents the syllables end-points in the word sound vector $\tilde{x}$ is decided accurately. There are H syllables in the word. The beginning index of k-th syllable is $\tilde{s}_k$ and the end index is $\tilde{s}_{k+1}$.

6.3 Feature Extraction

After preprocessing the speech signal, we have the syllable end-points of the word. The syllable utterences are framed with Hamming window as explained in

Subsection 2.5. The length of one frame is 20 ms with 10 ms shift (overlapping time=10 ms). 10 features are computed from each frame. These features are lpc, parcor, cepstrum, mfcc and rasta. The number of frames is not constant, but the number of frames is normalized to 30 frames with the length of 10 as shown in Figure 6.3. The normalized features are used only for the speech recognition methods as ANN, HMM and SVM. $s(n)$ is the syllable feature vector. $x(10, m)$ is the syllable feature matrix. For normalized features, m is 30 as illustrated in Figure 6.3. In Figure 6.4, the time duration for each speech feature is shown , and the fastest speech feature extraction algorithm among these features is mfcc.

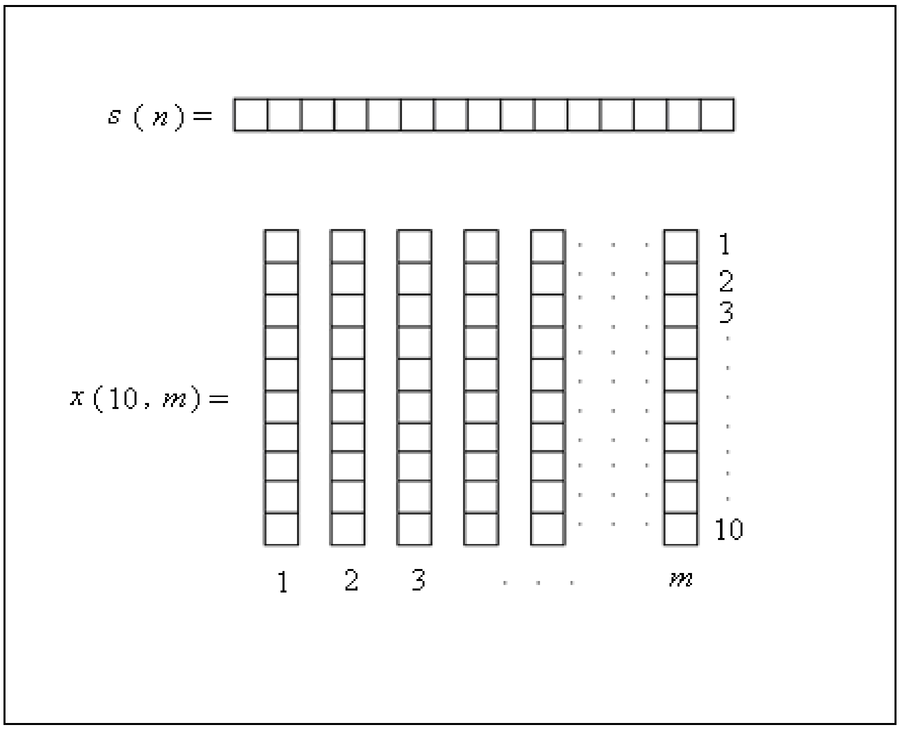


Figure 6.3 Feature extraction.

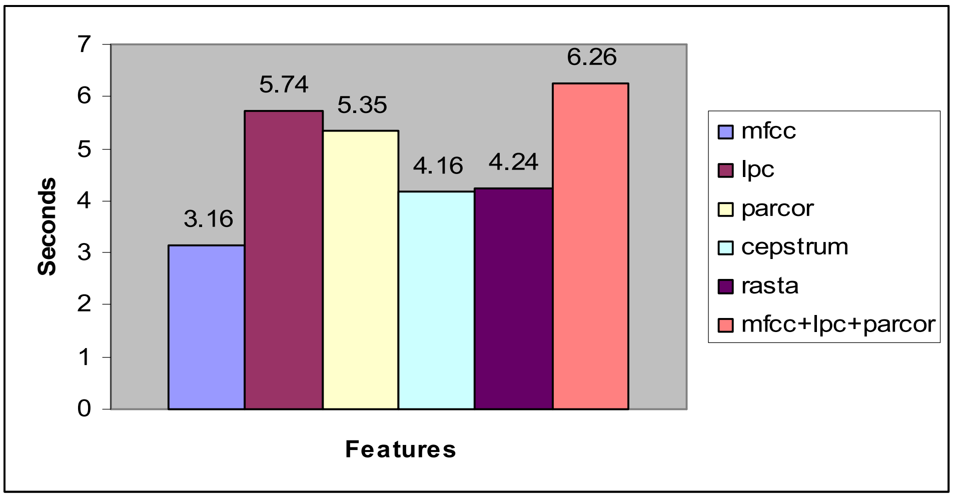


Figure 6.4 Feature extraction time duration.

6.4 Experiments with Linear Time Alignment

One of the speech recognition methods is linear time alignment. When the lengths of two vectors are different, this method provides us to compare the vectors using a distance metric according to their similarities.

Assume that x in Eq.6.11 is the digital speech signal which will be recognized, and y is one of the digital speech signals in the template database which was constructed from the syllable utterences of the dictionary before the recognition operation. n and m are sample numbers in the speech vector. n and m are usually different. To decide the similarity of these vectors we have used Euclidean distance metric, $d(x,y)$ as shown in Eq.6.12. So, the best match speech signal of x can be found using this metric. x and y are compared with each other after their lengths are made same.

$$\begin{aligned} x &= x(1), x(2), ..., x(n) \\ y &= y(1), y(2), ..., y(m) \end{aligned} \qquad \text{(Eq.6.11)}$$

$$d(x,y) = \left(\sum_{r=1}^{n} (x(r) - y(f(r)))^2 \right)^{1/2} \qquad \text{(Eq.6.12)}$$

$$f(r) = round\left(\left(\frac{m-1}{n-1}\right)r + \frac{n-m}{n-1}\right), \qquad r = 1,2,...,n \qquad \text{(Eq.6.13)}$$

$f(r)$ in Eq.6.13 is the function which makes same length for the given r values. The results of $f(r)$ are the positive integers between 1 and *m*. Figure 6.5 illustrates the function $f(r)$.

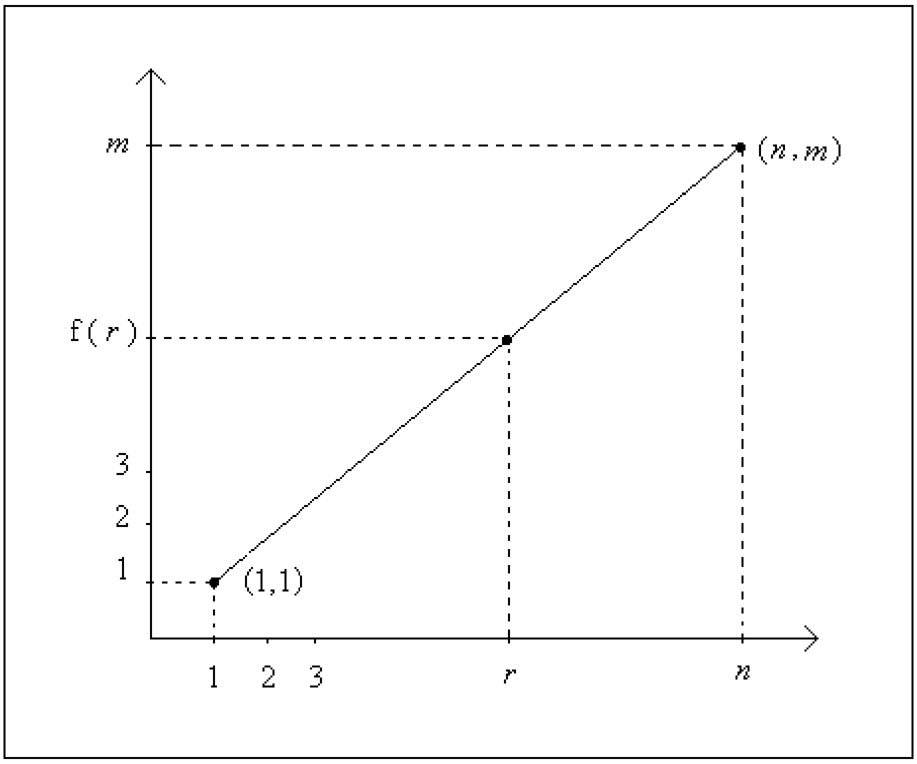
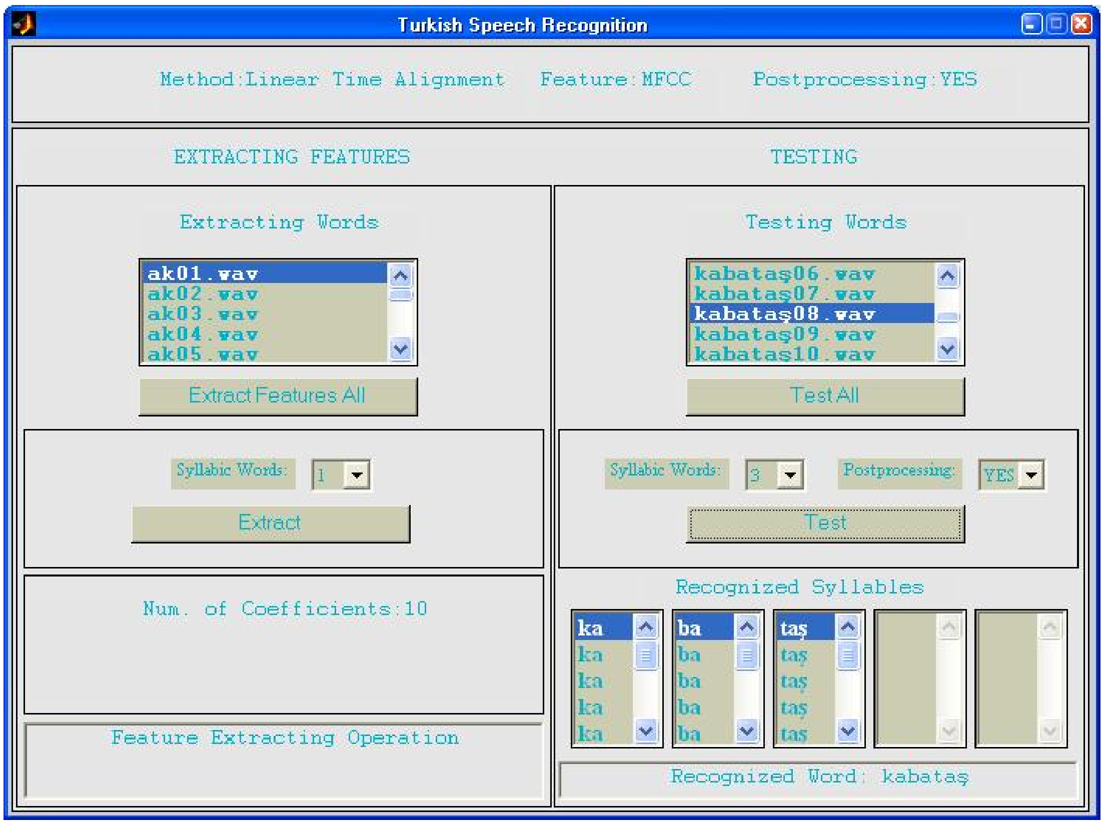


Figure 6.5 The function for linear time alignment.



Method:Linear Time Alignment Feature:MFCC Postprocessing:YES

ak01.wav
ak02.wav
ak03.wav
ak04.wav
ak05.wav

kabataş06.wav
kabataş07.wav
kabataş08.wav
kabataş09.wav
kabataş10.wav

Syllabic Words: 1

Syllabic Words: 3 Postprocessing: YES

Num. of Coefficients:10

ka	ba	taş
ka	ba	taş
ka	ba	taş
ka	ba	taş
ka	ba	taş

Recognized Word: kabataş

Figure 6.6 The user interface of the speech recognition system.

Using linear time alignment, the system, which has the user interface as shown in Figure 6.6, detects the recognized syllables of the word which have smallest distances. After concatenation of the recognized syllables, the recognized word is found by the system if the postprocessing is not carried out. Section 6.5 explains how the postprocessing works.

6.4.1 Word Error Rate

The most widely used evaluation measure for speech recognition performance is Word Error Rate (WER) (Hunt, 1990; McCowan et al., 2005). The general difficulty of measuring the performance lies on the fact that the recognized word sequence can have different length from the reference word sequence. The WER is derived from the Levenshtein distance, working at word level instead of character.

This problem is solved by first aligning the recognized word sequence with the reference sequence using dynamic string alignment. The word error rate can then be computed as in Eq.6.14.

$$WER = 100\left(\frac{E}{N}\right) \qquad \text{(Eq.6.14)}$$

where E is the number of wrongly detected words, and N is the number of words in the reference set.

Table 6.1 Experimental results of speech recognition using LTA (without postprocessing)

Speech Recognition Experiments (No Postprocessing)	FEATURES					
	MFCC	LPC	PARCOR	CEPSTRUM	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	48	20	47	18	26
Two Syllabic Words (WER %)	22	54	40	75	27	37
Three Syllabic Words (WER %)	5	39	15	52	5	11
Four Syllabic Words (WER %)	6	42	21	64	6	16
Five Syllabic Words (WER %)	8	47	21	56	8	14
Total Words (WER %)	12.2	46	23.4	58.8	12.8	20.8

After testing of the system using linear time alignment, we have the WER results as shown in Table 6.1. The system is tested according to 6 different features. One of them is the mixture of the features which consist of mfcc, lpc and parcor features (4 mfcc, 4 lpc and 4 parcor features are concatenated). In addition, the system is evaluated according to *n*-syllabic words in the dictionary (*n*=1, 2, 3, 4, 5). The best result for LTA is gained with the mfcc feature as given in Figure 6.7. It is followed by rasta feature. Three syllabic words are the most successful words in the dictionary to be detected correctly. We can compare the systems which use and does not use postprocessing. Namely, the applications with the postprocessing, which are described in Section 6.5, improve the system accuracy rate about 14% using linear time alignment.

6.5 The Postprocessing of the System

After the syllables of the word utterence are recognized using the speech recognition method, and the most similar 10 syllables are put in order, the recognized syllables are concatenated and generated the recognized word. We can find the most similar words in order by concatenation of the most similar syllables. From the uppermost recognized words, it can be determined whether or not the word is Turkish (see Chapter 5). If the word is Turkish, it is the recognized word of the system. If these words are not Turkish, the system does not recognize any word.

For example, as shown in Table 6.2, the recognized syllables are ordered. Hence, the most similar syllables as “kı”, “tap” and “lik” have been found. These syllables are concatenated, and the most similar word as “kıtaplik” is constructed. But, the system decides that the word is not Turkish word. Then, the next most similar word is concatenated, and it is determined whether or not the word is Turkish. This process is continued until a Turkish word is found in these concatenated words. In this example, the word “kitaplık” which is generated from the syllables “ki”, “tap” and “lık” is detected by the system. Therefore, it is the recognized word using the postprocessing.

Table 6.2 The most similar syllables

The order of the most similar syllables	**Recognized Syllables**		
	“ki”	**“tap”**	**“lık”**
1.	kı	tap	lik
2.	ki	tap	lak
3.	ki	tep	lık
4.	ki	ta	lik
5.	kı	ta	lık

Table 6.3 Experimental results of speech recognition using LTA (with postprocessing)

Speech Recognition Experiments (Postprocessing)	FEATURES					
	MFCC	LPC	PARCOR	CEPSTRUM	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	48	20	47	18	26
Two Syllabic Words (WER %)	19	32	20	49	12	17
Three Syllabic Words (WER %)	1	14	5	18	4	2
Four Syllabic Words (WER %)	2	11	6	31	6	5
Five Syllabic Words (WER %)	2	16	6	23	6	4
Total Words (WER %)	8.8	24.2	11.4	33.6	9.2	10.8

Table 6.3 displays the WER of the system with the features for LTA. Postprocessing is used in these applications. It can be seen that WER decreases when the number of syllables in the word utterence ascends. Note that we can not apply postprocessing operation for one syllabic words.

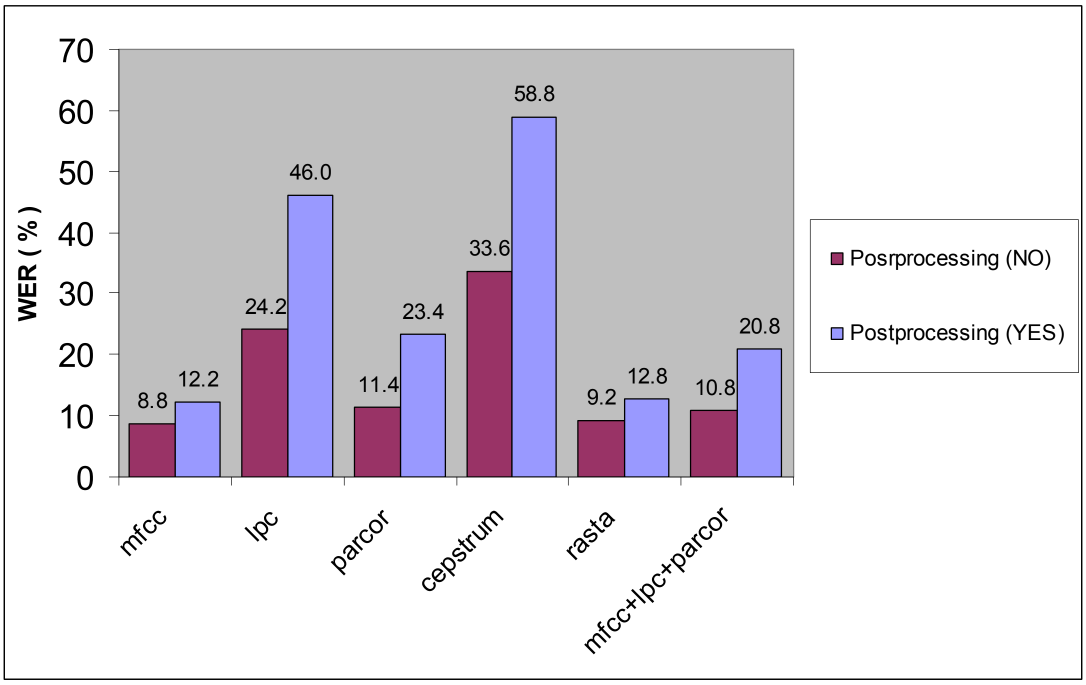


Figure 6.7 WER results of system using LTA.

6.5.1 Postprocessing Algorithm For Three Syllabic Word

The following algorithm is a function which takes the recognized syllables as inputs and which gives the recognized word as an output. The algorithm is explained for only three syllabic words.

```
If the word utterence is three syllabic word then
   Assume that st1, st2 and st3 are the matrix variables (column
   matrices) for the first, second and third recognized syllables of
   the word utterence respectively.
   Depth=10
   TotalDepth=3*Depth+1
   MaxDepth= TotalDepth
   For i=1 To Depth
      If MaxDepth < i+1+1, break end
      For j=1 To Depth
        If MaxDepth < i+j+1, break end
        For m=1 To Depth
          if MaxDepth maxder < i+j+m, break end
          kelime: st1{i}, st2{j} and st3{m} are concatenated, and
                the word is assigned to the variable kelime
          if kelime is Turkish word then
           TurkishWord{i+j+m}= kelime;
           MaxDepth =i+j+m;
          end
        end
      end
   end
  if MaxDepth < TotalDepth
    return TurkishWord {maxder};
  else
    return 'No-recognition';
  end
end
```

6.6 Experiments with Dynamic Time Warping

DTW is based on the principle of matching a speech signal converted into a feature matrix against a set of reference templates. The templates are simply feature matrix examples of each syllable of a word in the vocabulary of the system. Consequently, DTW is normally used for recognition of isolated words. The similarity between a template and the unknown speech is assumed to be inversely proportional to the minimum cost alignment. This is normally evaluated by calculating a local distance between each input features and all feature matrices of the reference template. Calculating the minimum cost alignment is then a matter of finding the optimal path from the bottom left-hand to the top right-hand corner of the matrix. Namely, the path that accumulates the lowest sum of local distances and does not stray too far away from the diagonal. The standard asymmetric dynamic programming decision rule adds a penalty for both horizontal and vertical deviances from the diagonal.

6.6.1 DTW Algorithm

The following algorithm takes two speech feature matrices as inputs to calculate the distance of them. The output of this algorithm is the distance of these features. These two matrices consist of n and m frames respectively. Each frame has ten speech features.

```
DTW_Distance (s[1..10,1..n], t[1..10,1..m], d[1..n,1..m])
    Define a matrix variable as DTW[0..n,0..m]
    Define variables as i, j, cost
    For i=1 To m
        DTW[0,i] = infinity
    end
    For i=1 To n
        DTW[i,0] = infinity
    end
    DTW[0,0] = 0
    For i=1 To n
        For j=1 To m
            cost = distance(s[1..10,i],t[1..10,j])
            DTW[i,j] = cost + minimum(DTW[ i-1, j   ],
                                      DTW[ i  , j-1 ],
                                      DTW[ i-1, j-1 ])
       end
    end
    return DTW[n,m]
```

In Table 6.4 and Table 6.5, the WER results are given for dynamic time warping. If we evaluate the system, we can say that the best result for DTW is obtained with the mfcc feature. It is followed by rasta feature. Three syllabic words are the most successful words in the dictionary to be detected correctly. The system accuracy rate as displayed in Figure 6.8 is increased with postprocessing operation about 9% using DTW.

Table 6.4 Experimental results of speech recognition using DTW (without postprocessing)

Speech Recognition Experiments (No Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	15	24	22	18	18
Two Syllabic Words (WER %)	21	48	44	27	25
Three Syllabic Words (WER %)	4	18	16	12	15
Four Syllabic Words (WER %)	6	26	21	18	17
Five Syllabic Words (WER %)	6	23	21	17	16
Total Words (WER %)	10.4	27.8	24.8	18.4	18.2

Table 6.5 Experimental results of speech recognition using DTW (with postprocessing)

Speech Recognition Experiments (Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+PARCOR
One Syllabic Words (WER %)	15	24	22	18	19
Two Syllabic Words (WER %)	12	28	24	12	14
Three Syllabic Words (WER %)	0	9	5	4	6
Four Syllabic Words (WER %)	0	13	7	6	7
Five Syllabic Words (WER %)	2	12	6	6	8
Total Words (WER %)	5.8	17.2	12.8	9.2	10.8

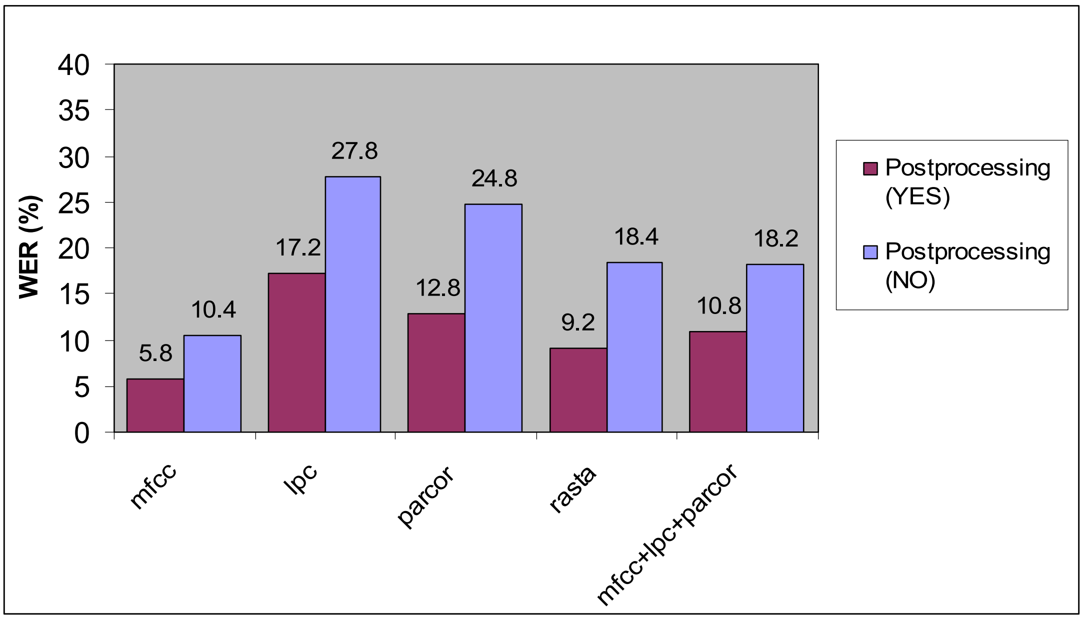


Figure 6.8 WER results of system using DTW.

6.7 Experiments with Artificial Neural Networks

A Neural Network (NN) is a computer software that simulates a simple model of neural cells in humans. The purpose of this simulation is to acquire the intelligent features of these cells.

Backpropagation networks are a popular type of network that can be trained to recognize different patterns including images, signal, and text. We have used backpropagation networks for our speech recognition system.

6.7.1 Sigmoid Function

The function as Eq.6.15 is called a Sigmoid function. The coefficient a is a real number constant. In NN applications, a is usually chosen between 0.5 and 2. As a starting point, we can use a=1 and modify it later when we are fine-tuning the network. Note that $s(0) = 0.5$, $s(\infty) = 1$, $s(-\infty) = 0$ (The symbol ∞ means infinity).

$$s(x) = \frac{1}{(1 + e - ax)} \qquad \text{(Eq.6.15)}$$

The sigmoid function will convert values less than 0.5 to 0, and values greater than 0.5 to 1. The Sigmoid function is used on the output of neurons.

6.7.2 Neuron

In NNs, a neuron is a model of a neural cell in humans. This model is simplistic, but as it turned out, is very practical. The neuron has been thought as a program or a class that has one or more inputs and produces one output. The inputs simulate the signals that a neuron gets, while the output simulates the signal which the neuron generates. The output is calculated by multiplying each input by a different number which is called weight, adding them all together, then scaling the total to a number between 0 and 1.

Figure 6.9 shows a simple neuron with:

1. Three inputs $[x_1, x_2, x_3, ..., x_{300}]$. The input values are usually scaled to values between 0 and 1.
2. 300 input weights $[w_1, w_2, w_3, ..., w_{300}]$. The weights are real numbers that usually are initialized to some random numbers. The weights are variables of type real. We can initialize to a random number between 0 and 1.
3. One output z. A neuron has only one output. Its value is between 0 and 1, it can be scaled to the full range of actual values.

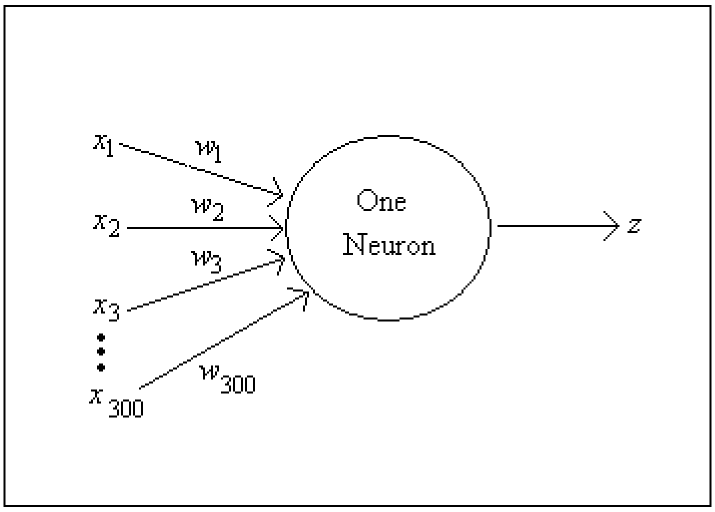


Figure 6.9 One neuron structure.

$$d = (x_1 * w_1) + (x_2 * w_2) + (x_3 * w_3) + \ldots + (x_{300} * w_{300}) \qquad \text{(Eq.6.16)}$$

In a more general manner, for *n* number of inputs, *d* is defined as Eq.6.17.

$$d = \sum_{i=1}^{n} x_i * w_i \qquad \text{(Eq.6.17)}$$

Let θ be a real number which is known as a threshold. Experiments have shown that best values for θ are between 0.25 and 1. θ is just a variable of type real that is initialized to any number between 0.25 and 1.

$$z = s(d + \theta) \qquad \text{(Eq.6.18)}$$

In Eq.6.18, the output *z* is the result of applying the sigmoid function on $(d + \theta)$. In NN applications, the challenge is to find the right values for the weights and the threshold.

6.7.3 Backpropagation

The structure of our system is shown in Figure 6.10. This NN consists of four layers: Input layer with 300 neurons, first hidden layer with 30 neurons, second hidden layer with 10 neurons and output layer with one neuron.

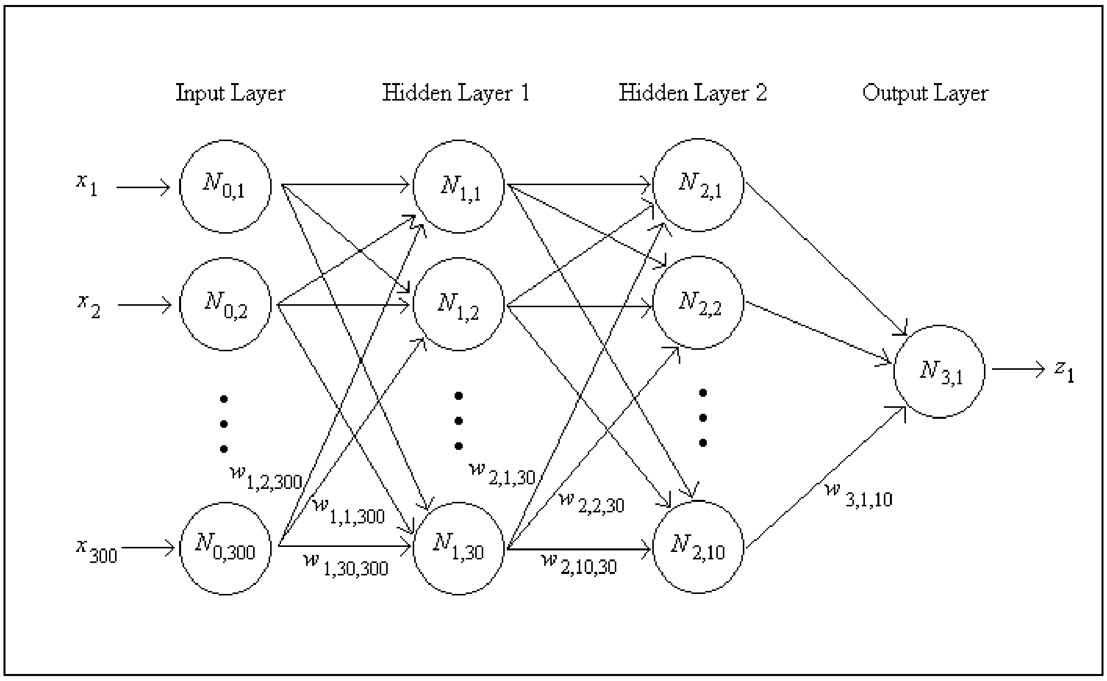


Figure 6.10 Backpropagation network of our system.

The output of a neuron in a layer goes to all neurons in the following layer. Each neuron has its own input weights. The weights for the input layer are assumed to be 1 for each input. In other words, input values are not changed. The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input. The Backpropagation NN must have at least an input layer and an output layer. It could have zero or more hidden layers.

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. In general, the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance.

6.7.3.1 Supervised Learning

The Backpropagation NN works supervised training. The training can be summarized as the following algorithm.

1. Start by initializing the input weights for all neurons to some random numbers between 0 and 1.
2. Apply input to the network.
3. Calculate the output.
4. Compare the resulting output with the desired output for the given input. This is called the error.
5. Modify the weights and threshold θ for all neurons using the error.
6. Repeat the process until the error reaches an acceptable value (the error, 0.006), which means that the NN was trained successfully, or if we reach a maximum count of iterations, which means that the NN training was not successful.

The challenge is to find a good algorithm for updating the weights and thresholds in each iteration (step 5) to minimize the error.

Changing weights and threshold for neurons in the output layer is different from hidden layers. For the input layer, weights remain constant at 1 for each input neuron weight.

For the training operation, we define the following:

1. The Learning Rate, λ: A real number constant, 0.02 for our system.
2. The change, Δ: For example Δx is the change in x.

6.7.3.2 Output Layer Training

Let *z* be the output of an output layer neuron. Let *y* be the desired output for the same neuron, it should be scaled to a value between 0 and 1. This is the ideal output which we like to get when applying a given set of input. Then the error, *e*, will be as Eq.6.19.

$$e = z * (1 - z) * (y - z) \qquad \text{(Eq.6.19)}$$

$$\Delta\theta = \lambda * e \qquad \text{(Eq.6.20)}$$

$$\Delta w_i = \Delta\theta * x_i \qquad \text{(Eq.6.21)}$$

$\Delta\theta$ represents the change in θ. Δw_i is the change in weight *i* of the neuron. In other words, for each output neuron, calculate its error *e*, and then modify its threshold and weights using Eq.6.19, Eq.6.20 and Eq.6.21.

6.7.3.3 Hidden Layer Training

Consider a hidden layer neuron as shown in Figure 6.11. Let z be the output of the hidden layer neuron. Let m_i be the weight at neuron N_i in the layer following the current layer. This is the weight for the input coming from the current hidden layer neuron. Let e_i be the error e at neuron N_i. Let r be the number of neurons in the layer following the current layer. (In Figure 6.11, $r = 3$).

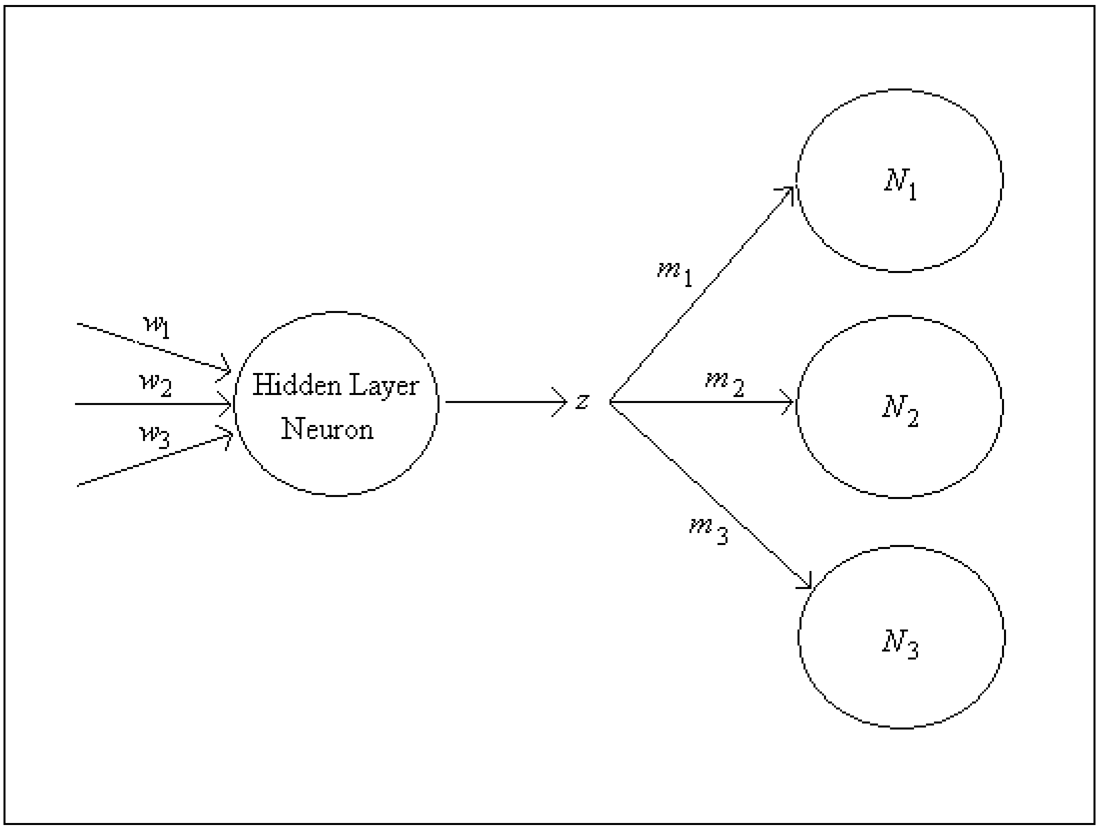


Figure 6.11 Hidden layer training.

$$g = \sum_{i=1}^{r} m_i * e_i \qquad \text{(Eq.6.22)}$$

$$e = z * (1 - z) * g \qquad \text{(Eq.6.23)}$$

$$\Delta\theta = \lambda * e \qquad \text{(Eq.6.24)}$$

$$\Delta w_i = \Delta\theta * x_i \qquad \text{(Eq.6.25)}$$

e is the error at the hidden layer neuron. $\Delta\theta$ is the change in θ. Δw_i is the change in weight *i*. In calculating g, we used the weight m_i and error e_i from the following layer, which means that the error and weights in this following layer should have already been calculated. This implies that during a training iteration of a Backpropagation NN, we start modifying the weights at the output layer, and then we proceed backwards on the hidden layers one by one until we reach the input layer. It is the method of proceeding backwards which gives this network its name Backward Propagation.

Table 6.6 Experimental results of speech recognition using ANN (without postprocessing)

Speech Recognition Experiments (No Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	42	29	27	27
Two Syllabic Words (WER %)	33	51	29	37	44
Three Syllabic Words (WER %)	16	35	20	29	26
Four Syllabic Words (WER %)	18	39	31	33	26
Five Syllabic Words (WER %)	18	40	27	29	30
Total Words (WER %)	21	41.4	27.2	31	30.6

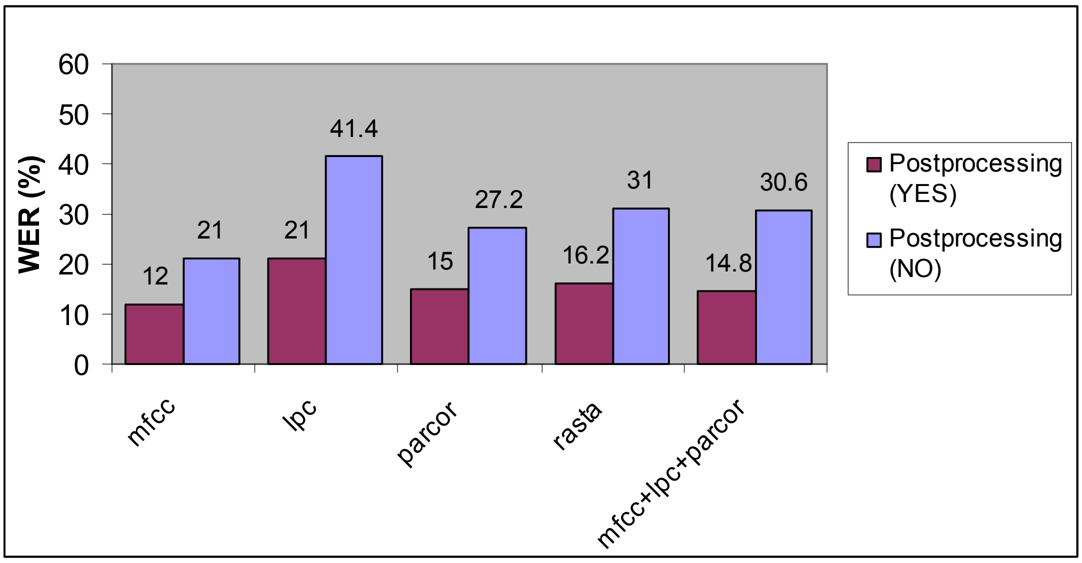


Figure 6.12 WER results of system using ANN.

Table 6.7 Experimental results of speech recognition using ANN (with postprocessing)

Speech Recognition Experiments (Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	42	29	27	27
Two Syllabic Words (WER %)	18	32	27	25	24
Three Syllabic Words (WER %)	6	12	6	9	7
Four Syllabic Words (WER %)	8	10	8	13	9
Five Syllabic Words (WER %)	8	9	5	7	7
Total Words (WER %)	12	21	15	16.2	14.8

In Table 6.6 and Table 6.7, the WER results are given for artificial neural network. If we evaluate the system, we can say that the best result for ANN is obtained with the mfcc feature. Three syllabic words are the most successful words in the dictionary to be detected correctly. The system accuracy rate as shown in Figure 6.12 is increased with postprocessing operation about 15% using ANN.

6.8 Experiments with Hidden Markov Models

In acoustic modeling part of the speech recognition system, we have modeled each syllable of the word in the dictionary from the syllable features. How we constructed HMM models is explained by the following subsections.

6.8.1 Constructing Hidden Markov Models

Calculating the parameters of hidden markov model $\lambda = (A, B, \pi)$ is one of the hardest problem in HMM. A, B and π parameters are calculated to satisfy an optimization criterion. Our optimization criterion is based on maximizing $P(O, \lambda)$ where O represents the training observations. In order to do that the Baum-Welch method also known as expectation-maximization (EM) (Rabiner & Juang, 1993) method is used. Before going any further, the form of the observation symbol probability distribution $B = \{b_j(k)\}$, needs to be made explicit. One can characterize observations as discrete symbols chosen from a finite alphabet and use a discrete probability density within each state of the model. On the other hand, feature parameters extracted from the speech signals can take continuous values. Therefore continuous observation densities are used to model feature parameters directly.

The output distributions are represented by Gaussian Mixture Densities as shown in Eq.6.26.

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} N(o_t, \mu_{jk}, U_{jk}),\ 1 \le j \le N \qquad \text{(Eq.6.26)}$$

where o_t is the observation vector being modeled, M is the number of mixture values used for each state (we used three mixture values ($M = 1$) for each state), N represents a Gaussian probability density function (pdf), and c_{jk} is the mixture coefficients for the k^{th} mixture in state j as Eq.6.27.

$$\sum_{k=1}^{M} c_{jk} = 1,\ c_{jk} \ge 0 \quad \begin{array}{l} 1 \le j \le N, \\ 1 \le k \le M \end{array} \qquad \text{(Eq.6.27)}$$

The Gaussian pdf N has a mean vector μ_{jk} and covariance matrix U_{jk} for the k^{th} mixture component in state j as Eq.6.28.

$$N(o_t, \mu_{jk}, U_{jk}) = \frac{1}{\sqrt{(2\pi)^n |U_{jk}|}} e^{-\frac{1}{2}(o_t - \mu_{jk})' U_{jk}^{-1} (o_t - \mu_{jk})} \qquad \text{(Eq.6.28)}$$

where n is the dimensionality of the observation vector o_t. On our case n is 10 feature parameters are extracted from each frame of the speech signal. Suppose that an HMM model contains just one state j and one mixture value k is used for this state. Then the parameter estimation would be easy. The maximum likelihood estimation of μ_{jk} and U_{jk} would be simple averages as follows:

$$\begin{aligned} \hat{\mu}_{jk} &= \frac{1}{T}\sum_{t=1}^{T} o_t \\ \hat{U}_{jk} &= \frac{1}{T}\sum_{t=1}^{T} (o_t - \mu_{jk})(o_t - \mu_{jk})' \end{aligned} \qquad \text{(Eq.6.29)}$$

where T is the number of observations. In practice, HMM models contain more than one state; each of which has more than one mixture component. Also, for a given model and observation sequence, there are no direct assignments of observation vectors to the individual states, as the underlying state sequence is not known. But Eq.6.28 and Eq.6.29 can be used if some approximate assignments of observation vectors to the states could be done.

Now we define the variable $\xi_t(i,j)$ (Rabiner & Juang, 1993) to help us the parameter estimation algorithm. The variable $\xi_t(i,j)$ is the probability of being in state i at time t, and state j at time $t+1$, given the model λ and observation sequence O as shown in Eq.6.30.

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \lambda) \qquad \text{(Eq.6.30)}$$

The variable $\xi_t(i,j)$ can be rewritten by using the definitions of the forward and backward variables as Eq.6.31.

$$\begin{aligned}\xi_t(i,j) &= \frac{P(q_t = i, q_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}\end{aligned} \qquad \text{(Eq.6.31)}$$

A posteriori variable $\gamma_t(i)$ is defined for making the parameter estimation algorithm tractable as Eq.6.32.

$$\gamma_t(i) = P(q_t = i \mid O, \lambda) \qquad \text{(Eq.6.32)}$$

That is, as the probability of being in state i at time t, given the observation sequence O, and the model λ. Then we can express $\gamma_t(i)$ as Eq.6.33.

$$\begin{aligned}\gamma_t(i) &= P(q_t = i \mid O, \lambda) \\ &= \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)} \\ &= \frac{P(O, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)}\end{aligned} \qquad \text{(Eq.6.33)}$$

Eq.6.33 can be rewritten by the aid of forward and backward variables as Eq.6.34.

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \qquad \text{(Eq.6.34)}$$

We can relate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over j as Eq.6.34.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \qquad \text{(Eq.6.35)}$$

The summation of $\gamma_t(i)$ over the time index t can be interpreted as the expected number of times that state i is visited. Similarly, summation of $\xi_t(i,j)$ over t can be interpreted number of transitions from state i to j as Eq.6.36 and Eq.6.37.

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i \qquad \text{(Eq.6.36)}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from state } i \text{ to } j \qquad \text{(Eq.6.37)}$$

If the current model is defined as $\lambda = (A, B, \pi)$ then a set of reasonable reestimation formulas for the parameters of the model can be defined as in Eq.6.38, Eq.6.39 and Eq.6.40.

$$\overline{\pi}_j = \text{expected number of times in state } i \text{ at time } t = 1 \qquad \text{(Eq.6.38)}$$

$$\begin{aligned} \overline{a}_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \end{aligned} \qquad \text{(Eq.6.39)}$$

$$\overline{b}_j(o_t) = \sum_{k=1}^{M} \overline{c}_{jk} N(o_t, \overline{\mu}_{jk}, \overline{U}_{jk}), \quad 1 \le j \le N \qquad \text{(Eq.6.40)}$$

In Eq.6.40, the equations for the reestimation of the coefficients $\overline{c}_{jk}, \overline{\mu}_{jk}$ and $\overline{U}_{jk}$ are given as Eq.6.41, Eq.42 and Eq.6.43.

$$\overline{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} \qquad \text{(Eq.6.41)}$$

$$\overline{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) o_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \qquad \text{(Eq.6.42)}$$

$$\overline{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(o_t - \mu_{jk})(o_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \qquad \text{(Eq.6.43)}$$

where $\gamma_t(j,k)$ is the probability of being in state j at time t with the k^{th} mixture component accounting for o_t as Eq.6.44.

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jk} N(o_t, \mu_{jk}, U_{jk})}{\sum_{m=1}^{M} c_{jm} N(o_t, \mu_{jm}, U_{jm})} \right] \qquad \text{(Eq.6.44)}$$

At the end of these computation, a reestimated model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is obtained and either $\overline{\lambda} = \lambda$, that is $P(O \mid \overline{\lambda}) = P(O \mid \lambda)$, or model $\overline{\lambda}$ is more likely than λ, that is, $P(O \mid \overline{\lambda}) > P(O \mid \lambda)$ (Rabiner, 1989).

In this way, we can iteratively use $\overline{\lambda}$ in place of λ and repeat the reestimation calculations to improve the probability of O being observed from the model until some limiting point is reached.

In this thesis, Turkish syllables are used as the smallest unit for speech recognition. Each syllable is represented by a three state left to right HMM model.

6.8.2 Training and Recognition with HMM

After a spoken word is divided into syllable sequences, each syllable can be represented with a sequence of observation vectors O, defined as $O = o_1, o_2, ..., o_T$ where o_t is the t^{th} observation vector and T is the number of observation vectors for single syllable utterence. Then the syllable recognition can be defined as Eq.6.45.

$$s^* = \arg\max_i P(s_i \mid O) \qquad \text{(Eq.6.45)}$$

where s_i is the i^{th} syllable in the dictionary and s^* is the desired syllable. By using Bayes' rule, $P(s_i \mid O)$ can be expressed as Eq.6.46.

$$P(s_i \mid O) = \frac{P(O \mid s_i) P(s_i)}{P(O)} \qquad \text{(Eq.6.46)}$$

Prior probabilities $P(s_i)$ are taken to be equal to each other for all s_i in this thesis. Therefore the most probable syllable depends only on the likelihood $P(O \mid s_i)$. It is not feasible for a given observation sequence O, to directly estimate the joint probability $P(o_1, o_2, ..., o_T \mid s_i)$ for each syllable s_i as discussed previous section. For HMM models, $P(O \mid s_i)$ is replaced by estimating the HMM model parameters of the syllable s_i

Syllables are used as the smallest unit for training and recognition of the syllables. The training of these models and their usages in recognition phase will be introduced next subsection. First, the spoken word is divided into syllables. The features of syllable utterences are extracted and observation feature vectors $O = o_1, o_2, ..., o_T$ are calculated. The trained HMM models of syllables are used to construct the HMM model of a syllable s supplied by the dictionary. The Viterbi algorithm is then applied to the HMM model with the feature vectors to get the probability $p(s \mid O)$. Then, the recognized syllables are concatenated by each other in order. After that, the recognized word is decided using postprocessing operation.

6.8.3 The Training Process of HMM

There are several training algorithms for HMM which were introduced in some studies (Furui, 1980; Kenny, Lenning & Mermelstein, 1990; Nadas, Nahammoo & Picheny, 1988; Pepper, Barnwell & Clements, 1980). The training algorithm with four steps is as the followings:

1. Construct the HMM model topology of the syllable.
2. Guess initial set of model parameters for the HMM model.
3. Improve the HMM model.
4. Save the individual HMM models for each syllable in the word separately.

In that way, the syllables in the spoken word can be recognized. For instance, once the word “kitap” is trained, we have two HMM models for syllables /ki/ and /tap/. In the recognition stage, we can use the models for syllables.

In the algorithm above and throughout this subsection, it is assumed that an HMM model is trained by using twenty five utterence of a word. The use of multiple observation sequence adds no additional complexity to the algorithm above. Step 3 is simply repeated for each distinct training sequence. In this thesis, 200 words have been trained with their syllables.

For the rest of this subsection, assume that a training word has T frames in the speech signal, N_S syllables, N states and the HMM model $\lambda = (A, B, \pi)$.

6.8.4 Initial Guess of the HMM Model Parameters

Training algorithms always start with an initial guess. This section introduces the strategy to guess initial parameters for the HMM model $\lambda = (A, B, \pi)$. The training syllables of the word may already have been trained. If that is the case, these trained models are used as the initial guess, otherwise, the model parameters A, B and π are initially guessed as follows:

Initial guess of the state transition probability distribution, A, is given in Figure 6.13.

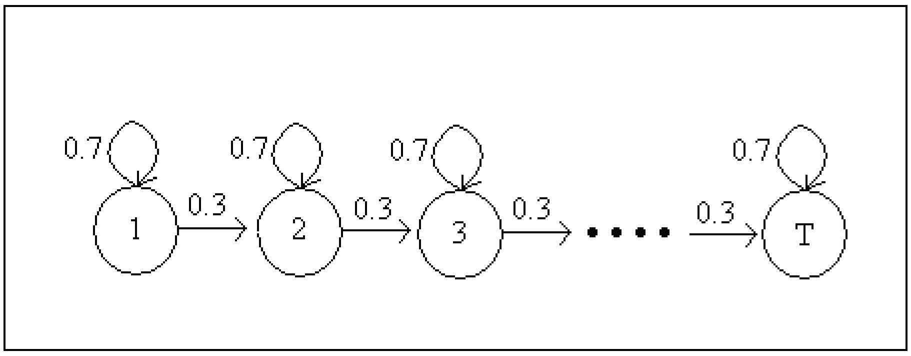


Figure 6.13 Initial estimate of the state transition probability distribution, A.

The transition goes from the rightmost state to itself because there is no other state on the left. Since the sum of the outgoing transition probabilities for a state must be 1, the probability of taking this transition is 1.

For initial guess of the observation symbol probability distribution, B, some approximate assignments of observation vectors to the individual states must be done. In this thesis, feature vectors extracted from the speech signal are distributed on the states of an HMM uniformly.

After assigning the feature vectors on syllables, we distribute feature vectors for a syllable over its three states such that , first one fifth of the feature vectors are assigned to the first state, next three fifths are assigned to the second state, and the last one fifth are assigned to the third state. Although this distribution is static, later in the improvement of HMM models, the Viterbi algorithm is used to modify this static distribution of the feature vectors on the states. Figure 6.14 shows the distribution of feature vectors on the states.

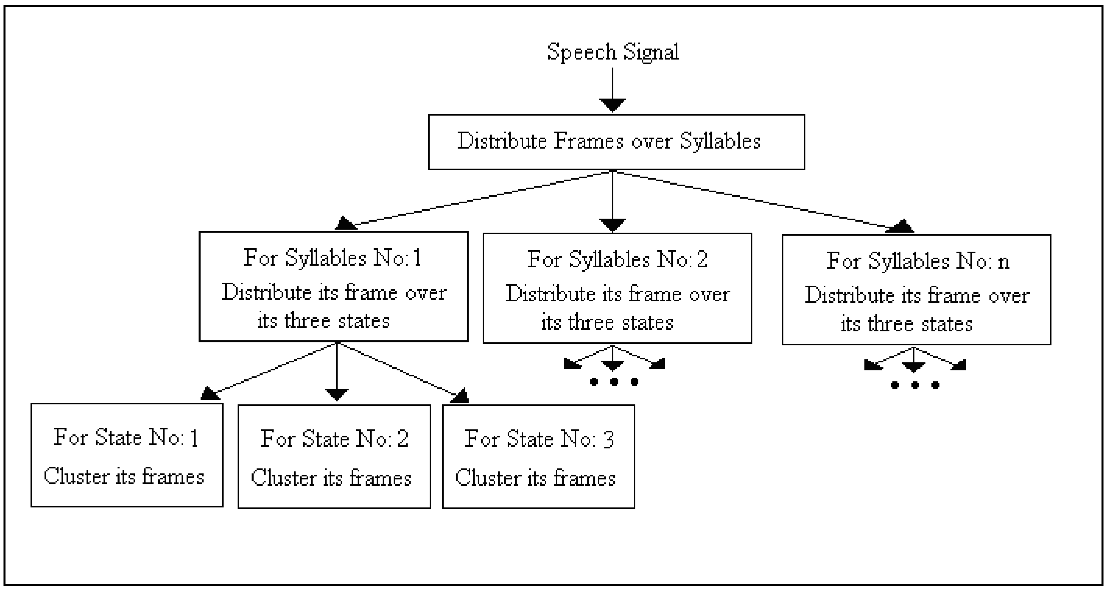


Figure 6.14 Distribution of feature vectors on the states.

In this thesis, continuous observation densities with three mixture values $(M = 3)$ for each state are used. K-means clustering algorithm (Juang & Rabiner, 1990) is used to cluster the feature vectors within each state j into a set of M clusters (using an Euclidean distance measure), where each cluster represents one of the M mixtures of the $b_j(o_t)$. These mixture values are used with the gives observation sequence O and the Eq.6.26 to compute the observation symbol probability distribution, B, for each state.

For initial state distribution, π, since the HMM model for a syllable has only one starting state. So, we have $\pi_i = 1$ and $\pi_i = 0$ for all i where $i = 2,3,...,N$.

At the end of these computations, we have the initial HMM model $\lambda = (A, B, \pi)$ for the training syllable.

6.8.5 Improving the HMM Model

Improvement of the model $\lambda = (A, B, \pi)$ means that the parameters of the model have to be reestimated to get an improved model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$. There are three main steps in the improvement of a HMM as shown in Figure 6.13.

First, we find the optimum state sequence for the given model $\lambda = (A, B, \pi)$ and given observation sequence O by using Viterbi algorithm. Optimum state sequence determines which state emits which frames. Therefore, we can consider the Viterbi algorithm as an another way of distributing feature vectors on the states of an HMM model such that $P(O \mid \lambda)$ is maximized.

Second, for each state, K-Means clustering algorithm is used to reestimate the clusters of its feature vectors according to the number of mixture used. The clusters may change since we change the distribution of the feature vectors on the states in the first step.

Finally, the equations between Eq.6.39 and Eq.6.44 are used to reestimate the parameters of the model $\lambda = (A, B, \pi)$ to get the new improved model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$. Note that the parameter π did not change because the new model should have also one start state.

If the iteratively use $\overline{\lambda}$ in place of λ and repeat the procedure, the probability of O being observed from the HMM model is improved until some limiting point is reached.

Viterbi algorithm is used to get better distribution of feature vectors on the states. After the better distribution, *K*-Means clustering algorithm is used to get better estimation of clusters for each of the model. Better clustering for each state means better estimation of the observation symbol probability distribution, B, and better estimate of the state transition probability distribution, A.

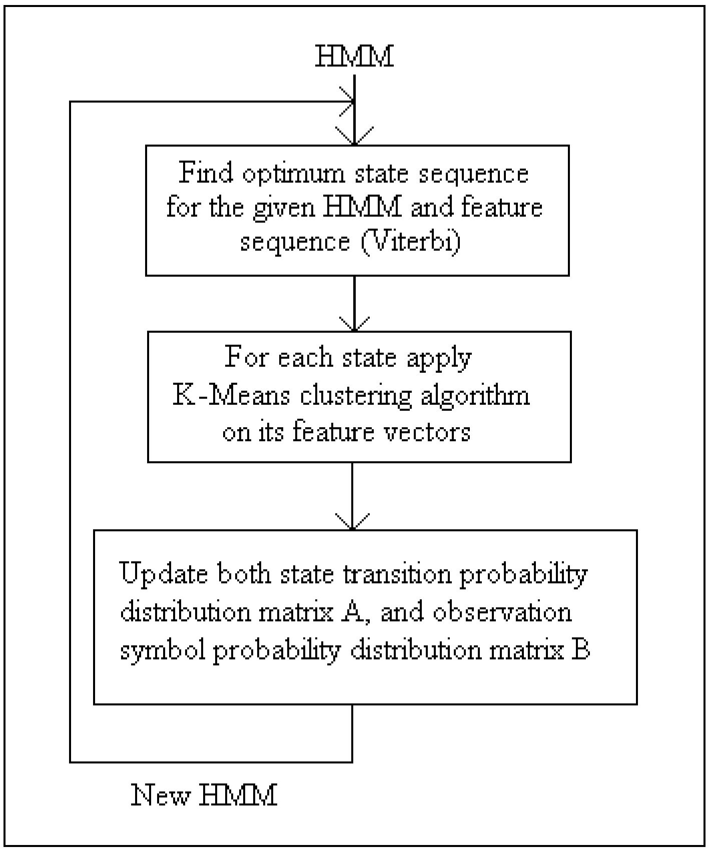

HMM
Find optimum state sequence for the given HMM and feature sequence (Viterbi)
For each state apply K-Means clustering algorithm on its feature vectors
Update both state transition probability distribution matrix A, and observation symbol probability distribution matrix B
New HMM

Figure 6.15 The improvement algorithm for a HMM model.

6.8.6 Recognition Process

In order to apply the Viterbi algorithm in the recognition stage, the observation symbol probability distribution $B = \{b_j(k)\}$ must be computed. The computation of B is on the order of $O(T.N.M)$ where T is the number of observation sequence, N is the number of states in the HMM model, and M is the number of mixture values used for each state.

In the training stage, after the creation of each syllable model, a space of mixture values is created by using the three mixture values of each state in every model. The space is 10-dimensional because we have 10-dimensional feature vectors. Later, this space is divided into C classes by the aid of the *K*-Means clustering algorithm. In this clustering, the similarity criterion is Euclidean distance in 10-dimensional space.

In the recognition stage, each observation vector is assigned to one of the clusters in the codebook by the help of the *K*-Means clustering algorithm. $s_j(\kappa_k)$ is the value of the probability density function of emitting an observation vector at state j which is assigned to the cluster κ_k (k^{th} cluster) and it is defined in as Eq.6.47.

$$s_j(\kappa_k) = \sum_{m=1}^{M} c_{jm} N(\kappa_k, \mu_{jm}, U_{jm}), \qquad \begin{matrix} 1 \le j \le N \\ 1 \le k \le C \end{matrix} \qquad \text{(Eq.6.47)}$$

where c_{jm} is the mixture coefficient for the m^{th} mixture in state j. N represents the Gaussian pdf with mean μ_{jm} and covariance matrix U_{jm} given in Eq.6.28. The computation of $s_j(\kappa_k)$ is performed at the training stage and it is inserted into the model of each syllable. Suppose that $d(o_t)$ stores the cluster number which the observation vector o_t is assigned that is $1 \le d(o_t) \le C$. Then, the computation of the observation symbol probability distribution, B, is just a table look-up process as expressed in Eq.6.48.

$$b_j(o_t) = s_j(d(o_t)) \qquad \begin{matrix} 1 \le j \le N, \\ 1 \le t \le T \end{matrix} \qquad \text{(Eq.6.48)}$$

Using Viterbi algorithm, the syllables in the word utterence are recognized and so by adding the syllables in order, the recognized word is found.

Table 6.8 Experimental results of speech recognition using HMM (without postprocessing)

Speech Recognition Experiments (No Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+PARCOR
One Syllabic Words (WER %)	14	20	18	15	22
Two Syllabic Words (WER %)	37	55	42	42	54
Three Syllabic Words (WER %)	23	29	26	48	39
Four Syllabic Words (WER %)	51	55	53	52	54
Five Syllabic Words (WER %)	48	57	48	53	52
Total Words (WER %)	34.6	43.2	37.4	42	44.2

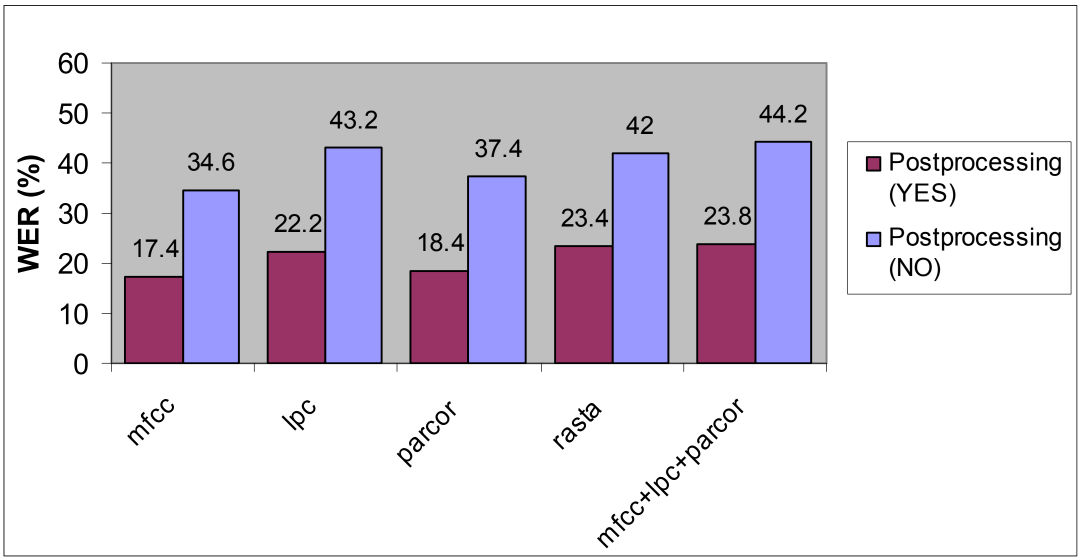


Figure 6.16 WER results of system using HMM.

Table 6.9 Experimental results of speech recognition using HMM (with postprocessing)

Speech Recognition Experiments (Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	14	20	18	15	22
Two Syllabic Words (WER %)	21	27	20	28	30
Three Syllabic Words (WER %)	7	6	7	15	11
Four Syllabic Words (WER %)	22	27	23	32	28
Five Syllabic Words (WER %)	23	31	24	27	28
Total Words (WER %)	17.4	22.2	18.4	23.4	23.8

In Table 6.8 and Table 6.9, the WER results are given for hidden markov model. If we evaluate the system, we can say that the best result for HMM is obtained with the mfcc feature. It is followed by parcor feature. Three syllabic words are the most successful words in the dictionary to be detected correctly. The system accuracy rate as shown in Figure 6.16 is increased with postprocessing operation about 19% using HMM.

6.9 Experiments with Support Vector Machines (SVM)

SVM is a technique of obtaining the optimal boundary of two sets in a vector space independently on the probabilistic distributions of training vectors in the sets. Its fundamental idea is quite simple; locating the boundary that is most distant from the vectors nearest to the boundary in both of the sets. This idea is a traditional one, however, recently has attracted much attention again. This is because of the introduction of kernel method, which is equivalent to a transformation of the vector space for locating a nonlinear boundary.

6.9.1 Basic Support Vector Machine

At first, we assume a linearly separable problem, as shown in Figure 6.17. Our aim is finding the optimal boundary hyperplane which exactly separates one set from the other. Note that our "optimal" boundary hyperplane should classify not only the training vectors, but also unknown vectors in each set. In the first session of this topic, the classification method by estimating probabilistic distributions of the vectors was explained. However, an accurate estimation is difficult since the dimension of vectors is often much higher than the number of training vectors. It was referred as "curse of dimensionality".

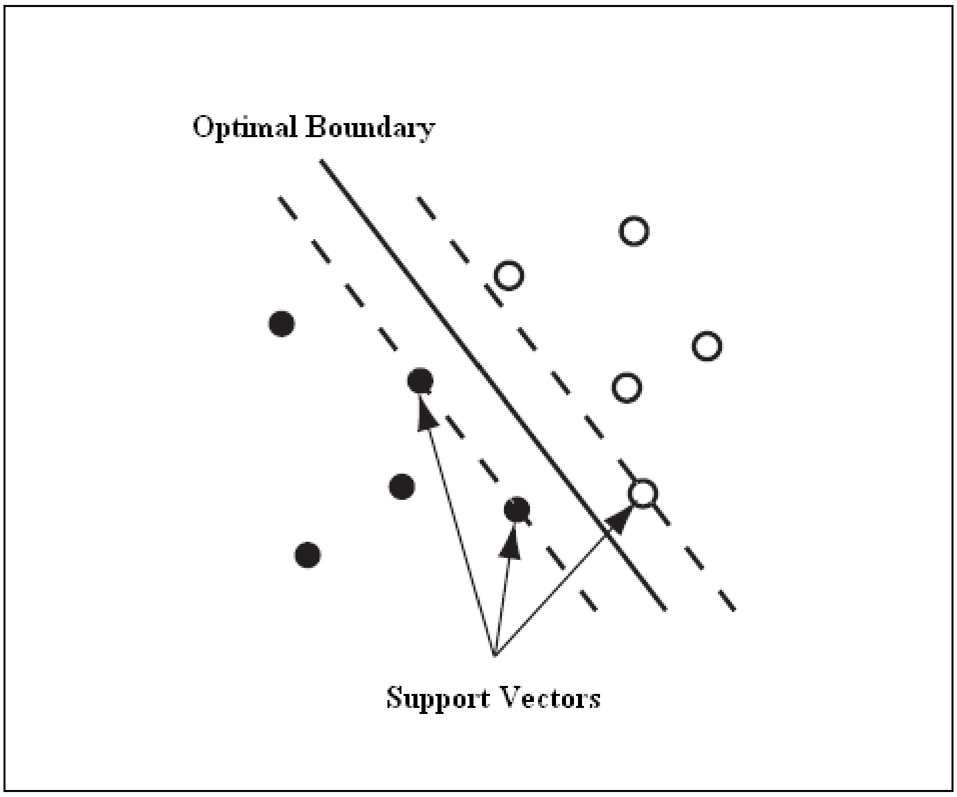


Figure 6.17 Optimal boundary with SVM.

In this approach, the "optimal" boundary is defined as the most distant hyperplane from both sets. In other words, this boundary passes the midpoint between these sets.

Although the distribution of each set is unknown, this boundary is expected to be the optimal classification of the sets, since this boundary is the most isolated one from both of the sets. The training vectors closest to the boundary are called support vectors.

Such boundary is defined to be passing through the midpoint of the shortest line segment between the convex hulls of the sets and is orthogonal to the line segment.

Let x be a vector in a vector space. A boundary hyperplane is expressed as one of the hyperplanes

$$w^T x + b = 0 \qquad \text{(Eq.6.49)}$$

where w is a weight coefficient vector and b is a bias term. The distance between a training vector x_i and the boundary, called "margin", is expressed as follows:

$$\frac{\left| w^T x_i + b \right|}{\|w\|} \qquad \text{(Eq.6.50)}$$

Since the hyperplanes expressed by Eq.6.49 where w and b are multiplied by a common constant are identical, we introduce a restriction to this expression, as follows:

$$\min_i \left| w^T x_i + b \right| = 1 \qquad \text{(Eq.6.51)}$$

$$\frac{1}{\|w\|^2} = \frac{1}{w^T w} \qquad \text{(Eq.6.52)}$$

The optimal boundary maximizes the minimum of Eq.6.50. By the restriction of Eq.6.51, this is reduced to maximization of Eq.6.52. Consequently, the optimization is formalized as Eq.6.53.

$$\begin{array}{ll} \text{minimize} & w^T w \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 \end{array} \qquad \text{(Eq.6.53)}$$

where y_i is 1 if x_i belongs to one set and –1 if x_i belongs to the other set. If the boundary classifies the vectors correctly as Eq.6.54 and it is identical to the margin.

$$y_i(w^T x_i + b) \geq 0 \qquad \text{(Eq.6.54)}$$

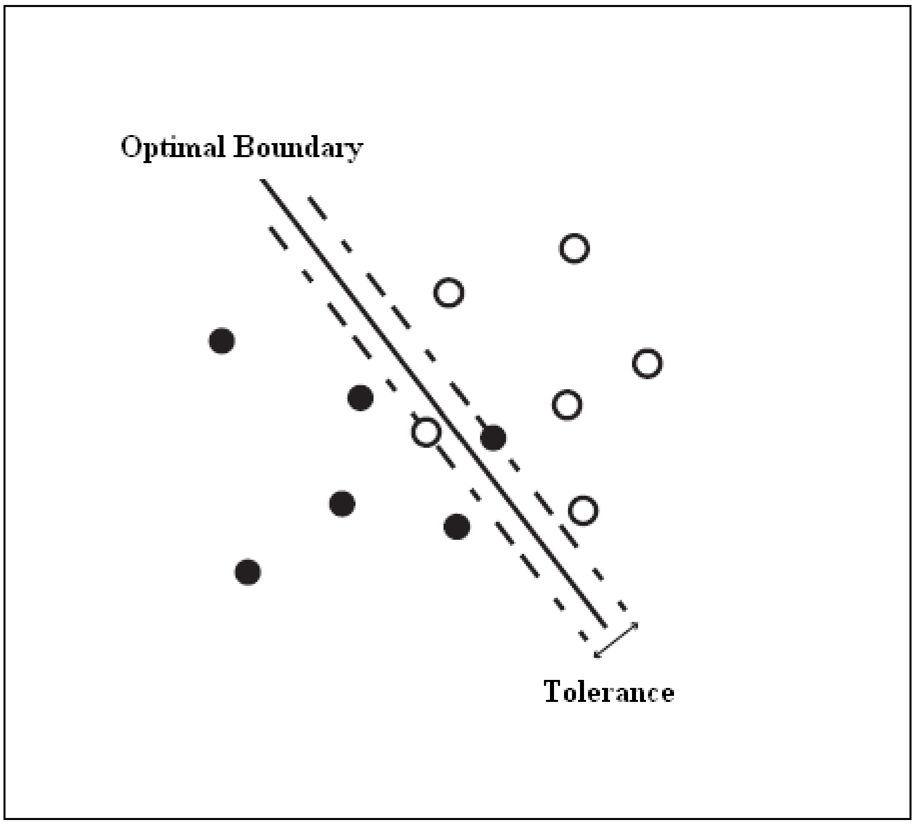


Figure 6.18 Linearly nonseparable case.

This conditional optimization is achieved by Lagrange's method of indeterminate coefficient. Let us define a function as Eq.6.55.

$$L(w,b,\alpha_i) = \frac{1}{2} w^T w - \sum_i \alpha_i \left[y_i (w^T x_i + b) - 1 \right] \tag{Eq.6.55}$$

where $\alpha_i \geq 0$ are the indeterminate coefficients. If w and b take the optimal value, the partial derivatives as Eq.6.56 are zero.

$$\begin{aligned} \frac{\partial L}{\partial w} &= w - \sum_i \alpha_i y_i x_i \\ \frac{\partial L}{\partial b} &= -\sum_i \alpha_i y_i \end{aligned} \tag{Eq.6.56}$$

Setting the derivatives of Eq.6.56 to zero, we get Eq.6.57 and Eq.6.58.

$$w = \sum_i \alpha_i y_i x_i \tag{Eq.6.57}$$

$$\sum_i \alpha_i y_i = 0 \tag{Eq.6.58}$$

Rewriting Eq.6.55, we get Eq.6.59.

$$L(w,b,\alpha_i) = \frac{1}{2} w^T w - \sum_i \alpha_i y_i w^T x_i - b \sum_i \alpha_i y_i + \sum_i \alpha_i \tag{Eq.6.59}$$

Substituting Eq.6.57 and Eq.6.58 to Eq.6.59, we get Eq.6.60.

$$
\begin{aligned}
L(w,b,\alpha_i) &= \frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)^T \left(\sum_j \alpha_j y_j x_j\right) \\
&\quad - \sum_i \alpha_i y_i \left(\sum_j \alpha_j y_j x_j\right)^T x_i + \sum_i \alpha_i \\
&= -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i
\end{aligned}
\qquad \text{(Eq.6.60)}
$$

The contribution of the second term of Eq.6.55 should be minimum, and L should be maximized subject to α. Consequently, the optimization is reduced to a quadratic programming problem as Eq.6.61.

$$
\begin{aligned}
&\text{maximize} \quad -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \\
&\text{subject to} \quad \sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_i = 0, \ \ \alpha_i \geq 0
\end{aligned}
\qquad \text{(Eq.6.61)}
$$

The above discussion is applicable to the case of linearly separable sets only. If the sets are not linearly separable, a hyperplane exactly classifying the sets does not exist, as explained in the previous subsection.

The method called "soft margin" is a solution to such case. This method replaces the restriction in Eq.6.53 with Eq.6.62.

$$
\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \xi_i \qquad \text{(Eq.6.62.)}
$$

where ξ_i called slack variables, are positive variables that indicate tolerances of misclassification. This replacement indicates that a training vector is allowed to exist in a limited region in the erroneous side along the boundary, as shown in Figure 6.18. Several optimization functions are proposed for this case.

$$
\text{minimize} \quad w^T w + C\sum_i \xi_i \qquad \text{(Eq.6.63)}
$$

The second term of Eq.6.63 is a penalty term for misclassification, and the constant C determines the degree of contribution of the second term.

6.9.2 Kernel Method

The soft margin method is an extension of the support vector machine within the linear framework. The kernel method explained here is a method of finding truly nonlinear boundaries.

The fundamental concept of kernel method is a deformation of the vector space itself to a higher dimensional space. We consider the linearly nonseparable example presented in the previous subsection, as shown in Figure 6.19. If the two-dimensional space is transformed to the three-dimensional one as shown in Figure 6.20, "black" vectors and "white" vectors are linearly separable.

Let Φ be a transformation to a higher dimensional space. The transformed space should satisfy that the distance is defined in the transformed space and the distance has a relationship to the distance in the original space. The kernel function $K(x, x')$ is introduced for satisfying the above conditions. The kernel function satisfies Eq.6.64.

$$K(x, x') = \Phi(x)^T \Phi(x') \qquad \text{(Eq.6.64)}$$

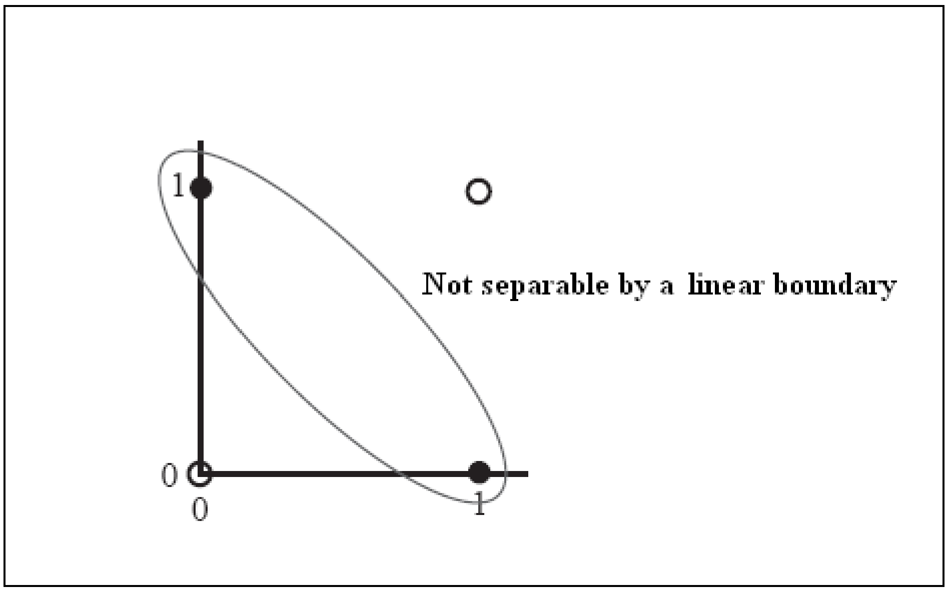


Figure 6.19 Transformation to higher dimensional space (not separable by linearly)

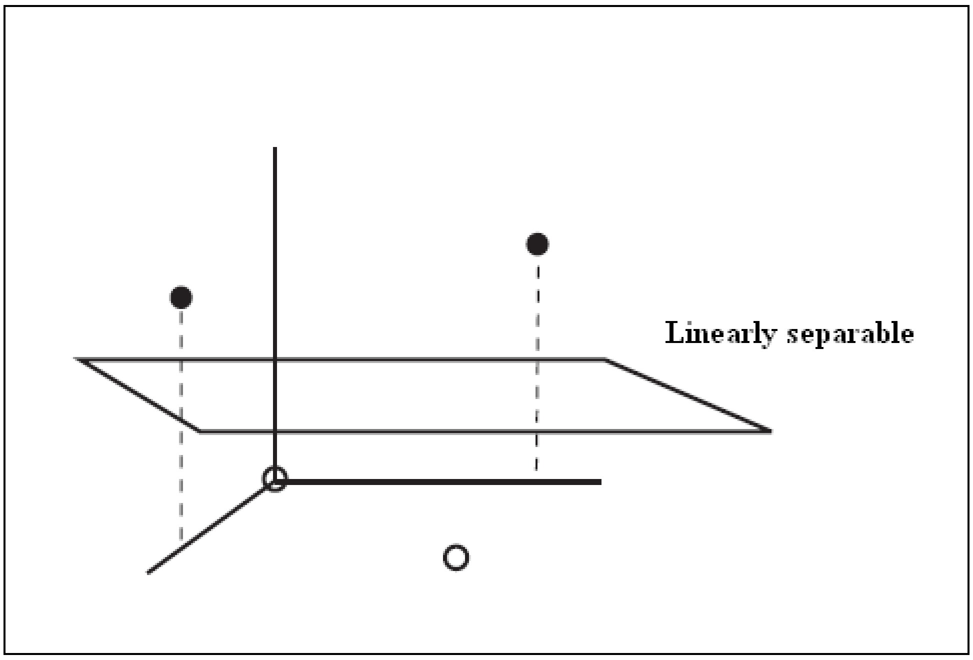


Figure 6.20 Transformation to higher dimensional space (linearly separable).

Eq.6.65 indicates that the kernel function is equivalent to the distance between x and x' measured in the higher dimensional space transformed by Φ. If we measure the margin by the kernel function and perform the optimization, a nonlinear boundary is obtained. Note that the boundary in the transformed space is obtained as Eq.6.65.

$$w^T \Phi(x) + b = 0 \qquad \text{(Eq.6.65)}$$

Substituting Eq.6.57 into the above equation with replacing x with Φ(x), we get Eq.6.66.

$$\sum_i \alpha_i y_i \Phi(x_i^T)\Phi(x) + b = \sum_i \alpha_i y_i K(x_i, x) + b = 0 \qquad \text{(Eq.6.66)}$$

The optimization function of Eq.6.61 in the transformed space is also obtained by substituting $x_i^T x_j$ with $K(x_i, x_j)$. These results mean that all the calculation can be achieved by using $K(x_i, x_j)$ only, and we do not need to know what Φ or the transformed space actually is.

A sufficient condition for satisfying Eq.6.64 is that K is positive definite. One example of such kernel functions is known as Eq.6.67 (Gaussian Kernel or Radial Basis Function Kernel, RBF).

$$K(x, x') = \exp(-\frac{\|x - x'\|^2}{\sigma^2}) \qquad \text{(Eq.6.67)}$$

The term empirical risk means the misclassification rate for known training vectors. It is not what we want to minimize; Our objective is minimizing the misclassification rate for all vectors in each set, including unknown vectors. This misclassification rate is called expected risk.

In case of linearly separable problems, there exists a boundary hyperplane that makes the empirical risk zero. The concept of support vector machine to find the boundary with the largest margin is equivalent to selecting a hyperplane minimizing the expected risk, from the set of hyperplanes that makes the empirical risk zero. This is formally explained in the framework of structural risk minimization with the concept of Vapnik-Chervenenkis (VC) dimensionality.

Table 6.10 Experimental results of speech recognition using SVM (without postprocessing)

Speech Recognition Experiments (No Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	37	24	25	25
Two Syllabic Words (WER %)	33	45	39	42	41
Three Syllabic Words (WER %)	10	23	13	29	16
Four Syllabic Words (WER %)	13	32	22	37	23
Five Syllabic Words (WER %)	10	30	18	27	23
Total Words (WER %)	17.2	33.4	23.2	32	25.6

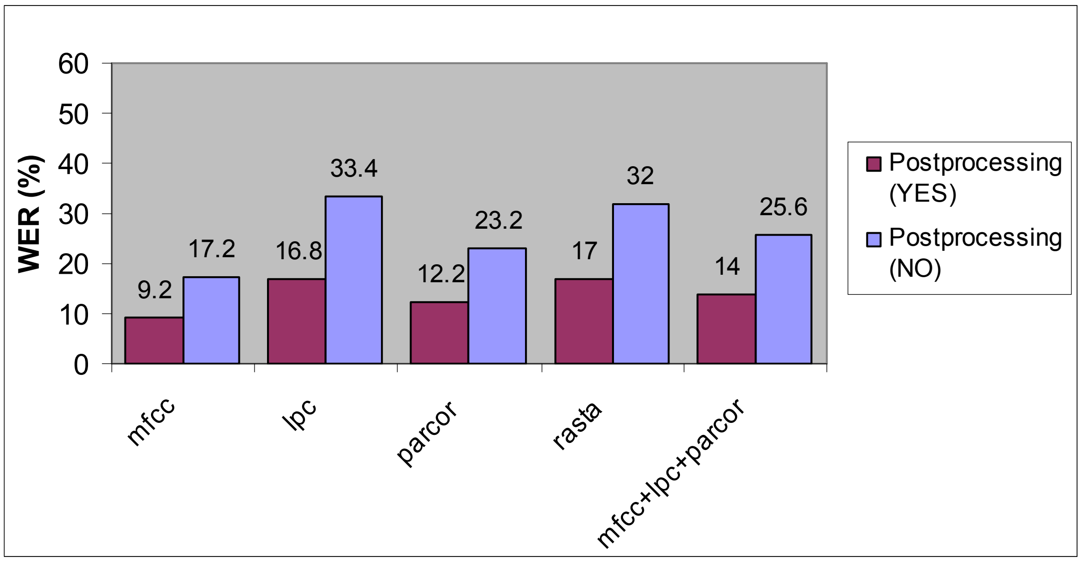


Figure 6.21 WER results of system using SVM.

Table 6.11 Experimental results of speech recognition using SVM (with postprocessing)

Speech Recognition Experiments (Postprocessing)	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
One Syllabic Words (WER %)	20	37	24	25	25
Two Syllabic Words (WER %)	18	27	24	27	26
Three Syllabic Words (WER %)	1	8	3	9	3
Four Syllabic Words (WER %)	5	8	6	15	9
Five Syllabic Words (WER %)	2	4	4	9	7
Total Words (WER %)	9.2	16.8	12.2	17	14

In Table 6.10 and Table 6.11, the WER results are given for support vector machine. If we evaluate the system, we can say that the best result for SVM is obtained with the mfcc feature. It is followed by parcor feature. Three and five syllabic words are the most successful words in the dictionary to be detected correctly. The system accuracy rate as shown in Figure 6.21 is increased with postprocessing operation about 13% using SVM.

6.10 Overall System Evaluation

According to the results of WER of the system, as shown in Table 6.12, the most successful feature and speech recognition method are mfcc and DTW (5.8% WER) respectively. The postprocessing which we proposed improves approximately 14% of the system accuracy.

LTA and DTW do not need training operation. They use the extracted features. The best feature is mfcc because its extraction time duration is the lowest. ANN, HMM and SVM need training, and they construct a model for each syllable in words using training. The method which has the shortest time duration is HMM as shown in Table 6.15, but there must be much more training words to be more successful for this method. In addition, SVM's accuracy rate is quite remarkable although its training time duration is short.

Table 6.12 Experimental results of speech recognition (with postprocessing)

Speech Recognition Methods	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
LTA (WER %)	8.8	24.2	11.4	9.2	10.8
DTW (WER %)	5.8	17.2	12.8	9.2	10.8
ANN (WER %)	12	21	15	16.2	14.8
HMM (WER %)	17.4	22.2	18.4	23.4	23.8
SVM (WER %)	9.2	16.8	12.2	17	14

Table 6.13 Syllable recognition results

Speech Recognition Methods	FEATURES				
	MFCC	LPC	PARCOR	RASTA	MFCC+LPC+ PARCOR
LTA (WER %)	7.9	22.1	11.2	8.7	11.3
DTW (WER %)	5.5	18.3	9.4	8.9	9.2
ANN (WER %)	9.1	24.6	14.1	10.8	13.2
HMM (WER %)	15.8	32.3	18.5	16.2	17.8
SVM (WER %)	7.8	22.7	10.9	9.1	10.2

As shown in Table 6.13, DTW is the most successful method for syllable recognition.

Table 6.14 Representative recognition WER for some isolated word recognizers

Authors	Environment	Vocabulary	Error Rates
Martin, 1975	Actual baggage handling application	34 words	WER 1.5%
Itakura, 1975	Telephone speech	200 Japanese words	WER 2.7%
Scott, 1977	Speaker independent	10 digits and 4 control words	WER 4%
Nippon electronic, 1978	Speaker dependent	10 digits	WER 0.2%
Nedim Karaca, 1999	Speaker independent.	130 words	WER 26.5%
Nuri İkizler, 2003	Speaker independent.	All Turkish syllables	Syllable ER 40%
Ebru Arısoy, Helin Dutağacı, 2006	Speaker independent.	Turkish letters	Letter error rates 20%
Özgül Salor, Bryan L. Pellom, 2007	Speaker independent.	Turkish phones	Phone recognition ER %29.2
Engin Avcı, 2007	Speaker independent.	15 words	WER 8%
Our system	Speaker dependent	200 words	WER 5.8%

Table 6.16 displays average testing time duration for each syllable. The fastest method is HMM, and it is fallowed by ANN, SVM, LTA and DTW respectively. Although the method, which has the best accuracy rate, is DTW, it has the longest time duration for testing operation.

Table 6.15 Average training time for one syllable

	Training Time (Seconds)
ANN	1102.5
HMM	17.5
SVM	578.2

Table 6.16 Average testing time for one syllable

	Testing Time (Seconds)
LTA	18.1
DTW	57.3
ANN	7.4
HMM	4.7
SVM	8.6

CHAPTER SEVEN
CONCLUSIONS

In this thesis, we have developed syllable based isolated word Turkish speech recognition systems using the speech recognition methods as LTA, DTW, ANN, HMM and SVM. These speaker dependent systems use the features as mfcc, lpc, parcor, cepstrum, rasta and the mixture of mfcc, lpc and parcor. We trained the system using ANN, HMM and SVM. Syllable models of the words in the dictionary are constructed syllable databases to compare the word utterence. The system firstly recognizes the syllables of the word utterence. Recognized word is found by the concatenation of the recognized syllables.

To use in postprocessing stage of the system, we have firstly designed and implemented TASA. TASA's correct spelling rate is about 100%. Then, we calculated Turkish syllable *n*-gram frequencies for some Turkish corpora.

In addition, using syllable *n*-gram frequencies, we have developed a system which decides whether or not a word is misspelled in Turkish text. The system takes words as inputs. The system produces two results for each word: "Correctly spelled word" or "Misspelled word". According to the system designed with bigram and trigram frequencies, the success rate is 97% for the misspelled words, and 98% for the correctly spelled words.

In postprocessing operation, after the recognized word is constructed by concatenating of the recognized syllables, the system decides whether it is Turkish word or not. If the word is Turkish word, then it is new recognized word. This postprocessing increases the accuracy rate of the system approximately 14%.

After testing of the middle scaled speech recognition system, we have seen that the most successful method is DTW whose word error rate is about 5.8%. It can be said that the best feature for the speech recognition is mel frequency cepstral coefficients.

7.1 Future Directions

By combination of some speech recognition methods, the system will be extended to syllable based hybrid system. In addition, speaker independent systems will be constructed.

Turkish is an agglutinative language. We can generate many words from a word by adding suffixes. Therefore, word based speech recognition systems are not adequate for Turkish to develop large scaled speech recognition systems. If the syllables of all Turkish words are modeled, large scaled system will be developed.

REFERENCES

Abdulla, W., Chow, D., & Sin, G. (2003). Cross-words reference template for DTW-based speech recognition systems. *In Proc. IEEE TENCON*, *4*, 1576-1579.

Anderson, C. W. & Kirby, M. J. (2003). EEG subspace representations and feature selection for brain-computer interfaces. *Proceedings of the first IEEE workshop on Computer Vision and Pattern Recognition for Human Computer Interaction*, Medison, Wisconsin.

Arısoy, E. & Dutağacı, H. (2006). A unified language model for large vocabulary continuous speech recognition of Turkish. *Signal Processing*. *86*(10), 2844-2862.

Artuner, H. (1994). The design and implementation of a Turkish speech phoneme clustering system. Ph. D. Thesis. Hacettepe University, Ankara.

Aşlıyan, R. & Günel, K. (2005). Design and implementation for extracting Turkish syllables and analysing Turkish syllables. *INISTA (International Symposium on Innovations in Inttelligent Systems and Applications)*, İstanbul.

Aşlıyan, R., Günel, K. & Yakhno, T. (2007). Detecting misspelled words in Turkish text using syllable n-gram frequencies, *Lecture Notes in Computer Science (LNCS), Pattern Recognition and Machine Intelligence*, *4815*, 553-559.

Avcı, E. (2007). An automatic system for Turkish word recognition using discrete wavelet neural network based on adaptive entropy. *Arabian Journal for Science and Engineering*. *32*, 239-250.

Ayuso, A. J. R. & Soler, J. M. L. (Eds.). (1993). *Speech recognition and coding*: *New advances and trends*. Berlin: Springer-Verlag.

Baker, J. (1975). The Dragon system, an overview. *IEEE Trans. ASSP*, *23*(1), 24-29.

Barari, L. & QasemiZadeh, B. (2005). CloniZER spell checker adaptive language independent spell checker. *AIML 05 Conference CICC*, Cairo Egypt, 19-21.

Bazara, M. & Shetty, C. M. (1979). *Nonlinear programming*. New York: John Wiley.

Becchetti, C. & Ricotti, L. P. (1999). *Speech recognition*: *Theory and C++ implementation*. Chichester: John Wiley & Sons.

Blanz, V., Schölkopf, B., Bulthoff, H., Burges, C., Vapnik, V. N. & Vetter, T. (1996). Comparison of view-based object recognition algorithms using realistic 3D models. *In Proc of ICANN'96. LNCS. 1112*, 251-256.

Boite, R. & Kunt, M. (1987). *Traitement de la parole*. Lausanne: Pres Politechnique Romandes.

Bourlard, H. A. & Morgan, N. (1997). *Connectionist speech recognition*: *A hybrid approach*. Massachusetts: Kluwer Academic Publishers.

Bridle, J. & Brown, M. (1979). Connected word recognition using whole word templates. *Proceedings of the Institute of Acoustics Autumn Conference*, 588-595.

Carson, J. & Berndsen, J. (1998). *Time map phonology*: *Fine state models and event logics in speech recognition*. Dordrecht: Kluwer Academic Publishers.

Cole, R., Rudnicky, R., Zue, V., & Reddy, D. (1980). Speech as patterns on paper. In R. Cole, (Ed.). *Perception and Production of Fluent Speech* (3-50). Hillsdale: Lawrence Erlbaum Ass.

Cortes, C. & Vapnik, V. N. (1995). Support vector network. *Machine Learning*. *20*, 1-25.

Courant, R. & Hilbert, D. (1981). *Methods of mathematical physics*. New York: John Wiley.

Cybenko, G. (1989). Approximation by super position of a sigmoidal function. *Mathematics of control, signals and system*, *2* (4), 303-314.

Dalkılıç, G. & Çebi, Y. (2003). Creating a Turkish corpus and determining word length. *DEÜ Mühendislik Fakültesi, Fen ve Mühendislik Dergisi*, *5*(1), 1-7.

Davis, K., Biddulph, R., & Balashek, S. (1952). Automatic recognition of spoken digits. *J. Acous. Soc. Ame.*, *24*, 3-50.

Davis, S. B. & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE trans. on Acoustic, Speech, and signal Processing*, *28* (4), 357-366.

Denes, P. (1959). The design and operation of the mechanical speech recognizer at University College London. *Journal of the British Institute of Radio Engineers*, *19*, 211-229.

Deorowicz, S. & Ciura M. G. (2005). Correcting spelling errors by modelling their causes. *International Journal of Applied Mathematics and Computer Science*.15(2), 275-285.

Dreyfus-Graf, J. (1952). Letyposonographe phonetique ou phonetographe. *Bulletin Technique Des PTT Suisses*, *12*, 363-379.

Dudley, H. & Balashek, S. (1958). Automatic recognition of phonetic patterns in speech. *J. Acoustic Soc. Am.*, *30*, 721-733.

Ferguson, J. (Ed.). (1980). *Hidden Markov Models for speech.* IDA-CRD, Princeton, New Jersey.

Fisher, W., Doddington, G., & Goudie-Marshall, K. (1986). The DARPA speech recognition database: Specifications and status. *In DARPA Workshop on speech Recognition*, 93-99.

Forgie, J. & Forgie, C. (1959). Results obtained from a vowel recognition computer program. *J. Acoust. Soc. Ame.*, *31*, 1480-1489.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Furui, S. (1980). A training procedure for isolated word recognition systems. *IEEE Transactions on Accoustic, Speech, and Signal Processing, ASSP-28. 2*.

Gupta, L., Molfese, D., Tammana, R., & Simos, P. (1996). Nonlinear alignment and averaging for estimating the evoked potential. *In IEEE Transactions on Biomedical Engineering*, *43* (4), 346-356.

Hakkani, D. Z., Oflazer, K. & Tür, G. (2000). Statistical morphological disambiguation for agglutinative languages. Technical Report, Bilkent University, Ankara.

Hartman, E. J., Keeler J. D., & Kowalski, J. M. (1990). Layered neural networks with gaussian hidden units as universal approximations. *Neural Computation*, *2*, 210-215.

Haton, J. P. (1974). A practical application of a real time isolated word recognition. *IEEE Trans. ASSP*, *6*(22), 416-419.

Haton, J. P. & Pierrel, J. M. (1976). Organization and operation of a connected speech understanding system at lexical, syntactical and semantical levels. *In ICASSP*, 430-433.

Hermansky, H., Morgan, N., Bayya, A. & Kohn, P. (1991). Compensation for the effect of the communication channel in auditory-like analysis of speech. Proc. of Eurospeech'91, 1367-1371.

Hermansky, H. (1998). Should recognizers have ears. *Speech communication*, *25*(3), 3-27.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359-366.

Huang, X. & Hon, A. A. H. (2001). *Spoken language processing*: *A guide to theory, algorithm, and system development*. New York: Prentice Hall.

Hunt, M. J. (1990). Figures of merit for assessing connected word recognisers. *Speech Communication*, *9*, 229-236.

Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Trans. ASSP*, *23*, 67-72.

Jakobson, R., Fant, G., & Halle, M. (1952). *Preliminaries to speech analysis* (1st ed.). Cambridge, MA: MIT Press.

Jelinek, F. (1976). Continuous speech recognition using statistical methods. *Proc. IEEE*, *64*(4), 532-556.

Jelinek, F. (1998). Statistical methods for speech recognition. Cambridge: MIT Press.

Juang, B. H. & Rabiner, L. R. (1990). The segmental K-means algorithm for estimating parameters of hidden markov models. *IEEE Transactions on Accoustic, Speech, and Signal Processing*. *38*(9).

Junqua, J. C., Mak, B. & Reaves, B. (1997). A robust algorithm for word boundary detection in the presence of noise. *IEEE Transactions on Speech and Audio Processing*, *2*(3), 406-412.

Jurafski, D. & Martin, J. H. (2000). *Speech and language processing*. New Jersey: Prentice Hall.

Kang, S. S. & Woo, C. W. (2001). Automatic segmentation of words using syllable bigram statistics. *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, Tokyo, Japan, 729-732.

Karaca, N. (1999). Realization of a Turkish isolated word speech recognition system under noisy environments. Ph. D. Thesis. Hacettepe University, Ankara.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Accoustics, Speech and Signal Processing, ASSP*, *35*(3), 400-4001.

Kenny, P., Lenning, M. & Mermelstein, P. (1990). A linear predictive HMM for vector-valued observation with applications to speech recognition. *IEEE Transactions on Accoustic, Speech, and Signal Processing*, *38*(2).

Koç, A. (2002). Acoustic feature analysis for robust speech recognition. M. S. Thesis. Boğaziçi University, İstanbul.

Kruskall, J. & Liberman, M. (1983). *The theory and practice of sequence comparison*. Massachusetts: Addison-Wesley Publishing.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*. *24*(4), 377-439.

Le Cun, Y. (1985). Une proc´edure d’apprentissage pour r´eseau `a seuil assymetrique. *Cognitiva 85*, 599–604.

Lee, C. & Rabiner, L. (1989). A frame synchronous network search algorithm for connected word recognition. *IEEE Trans. ASSP*, *37*(11), 1649-1658.

Lesser, V., Fennell, R., Erman, L., & Reddy, D. (1975). Organization of the HEARSAY II speech understanding system. *IEEE Trans. ASSP*, *23*(1), 11-23.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE Trans. ASSP Magazine*, *4*(2), 4-22.

Lowerre, B. (1976). The harpy speech recognition system. Technical Report. Carnegie Mellon University.

Makhoul, J. & Schwartz, R. (1994). State of the art in continuous speech recognition. In D. Roe, & J. Wilpon, (Eds.). *Voice Communication Between Humans and Machine* (165-198). Washington: National Academy Press.

Martin, T., Nelson, A., & Zadel, H. (1964). Speech recognition by feature abstraction techniques. Technical Report. Air Force Avionics Lab.

McCowan, I., Moore, D., Dines, J., Gatica-Perez, D., Flynn, M., Wellner, P., et al. (2005). On the use of information retrieval measure for speech recognition evaluation. IDIAP-RR 73, IDIAP.

Mengüşoğlu, E. (1999). Bir Türkçe sesli ifade tanıma sisteminin kural tabanlı tasarımı ve gerçekleştirimi. Yüksek Lisans Tezi. Hacettepe Üniversitesi, Ankara.

Meral, O. (1996). Speech recognition based on pattern comparison techniques. M.S. Thesis. İstanbul Technical University, İstanbul.

Mercier, G. (1977). A multipurpose speech understanding system. *In ICASSP*, 815-818.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Nadas, A., Nahammoo, D. & Picheny, M. A. (1988). On a model robust training method for speech recognition. *IEEE Transactions on Accoustic, Speech, and Signal Processing*. *36*(8).

Nagata, K., Kato, Y., & Chiba, S. (1963). Spoken digit recognizer for Japanese language. *NEC Res. Develop.*, *6*, 2.

Oflazer, K. (1994). Two-level description of Turkish morphology. *Literary and Linguistics*. *9*(2), 137-148.

Oflazer, K. & Bozşahin, H. C. (1994). Turkish natural language processing initiative: An overview. *In Proc. of the Third Turkish Symposium on Artifical Intelligence*, Middle East Technical University, Turkey.

Olson, H. & Bellar, H. (1956). Phonetic typewriter. *J. Accous. Soc. Ame.*, *2*, 1072-1081.

Osuna, E., Freud, R. & Girosi, F. (1997). Training support vector machines: An applications to face detection. *In CVPR97*. 130-136.

Özkan, Ö. (1997). Implementation of speech recognition for connected numerals. M.S. Thesis. Middle East Technical University, Ankara.

Parker, D. (1985). Learning logic. MIT Technical Report, TR-47, Cambridge, MA.

Pepper, D., Barnwell, T. P. & Clements, M. A. (1990). Using a ring parallel processor for hidden markov model training. *IEEE Transactions on Accoustic, Speech, and Signal Processing*. *36*(2).

Rabiner, L. & Sambur, M. R. (1975). An algorithm for determining the end-points of isolated utterances. *The Bell System Technical Journal*, *54*(2), 297-315.

Rabiner, L., Levinson, S., Rosenborg, A., & Wilpon, J. (1979). Speaker independent recognition of isolated words using clustering techniques. *IEEE Trans. ASSP*, *27*(4), 336-349.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, *77*(2), 257-286.

Rabiner, L. & Juang, B. H. (1993). *Fundamentals of speech recognition*, New York: Prentice-Hall.

Reddy, D. (1966). An approach to computer speech recognition by direct analysis of the speech wave. Technical Report, Stanford University.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Parallel distributed processing*: *Explorations in the microstructure of cognition*. Cambridge, MA: MIT Press.

Sakai, T. & Doshita, S. (1962). The phonetic typewriter. *In IFIP Congress*, 445-449.

Sakoe, H. & Chiba, S. (1971). A dynamic programming approach to continuous speech recognition. *In 7th ICA*, 20C-13.

Sakoe, H. (1979). Two level DP matching a dynamic programming based pattern matching algorithm for connected word recognition. *IEEE Trans. ASSP*, *27*(6), 588-595.

Salor, Ö & Pellom, B. L. (2007). Turkish speech corpora and recognition tools developed by porting SONIC: Towards multilingual speech recognition. *Computer Speech and Language*. *21*(4), 580-583.

Salvador, S. (2004). Learning states for detecting anomalies in time series. M.S. Thesis. Dept. Of Computer Sciences, Florida Institute of Technology, Florida.

Savoji, M. H. (1989). End-pointing of speech signals. *Speech Communication*, *8*(1), 46-60.

Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T. & Vapnik, V. N. (1996). Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. on Signal Processing*. *45*(11), 2758-2765.

Slutsker, G. (1968). Nelinejnyp method analiza recevych signalov. *Trudy NIIR*, *2*, 76-82.

Suzuki, J. & Nakata, K. (1961). Recognition of Japanese vowels - Preliminary to the recognition of speech. J. *Radio Research Lab.*, *37*(8) 193-212.

Tong, X. & Evans, D. A. (1996). A statistical approach to automatic OCR error correction in context. *Proceedings of the Fourth Workshop on Very Large Corpora*, Copenhagen. Denmark, 88-100.

Vapnik, V. N. & Chervonenkis, A. J. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*. *16*, 264-280.

Vapnik, V. N. (1982). *Estimation of dependencies based on empirical data*. New York: Springer-Verlag.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Velichko, V. & Zagoruyko, N. (1970). Automatic recognition of 200 words. *Int. J. Man-Machine Studies*, *2*, 223.

Vicens, P. (1969). Aspects of speech recognition by computer. Ph. D. Thesis. Computer Science, Stanford University, California.

Vintsyuk, T. (1968). Speech discrimination by dynamic programming. *Kibernetika*, *Cybernetics*, *4*(1), 81-88.

Werbos P. J. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. Thesis. Harvard University, Cambridge.

Wilpon, J. (1994). Applications of voice processing technology in telecommunications. In D. Roe, & J. Wilpon, (Eds.). *Voice Communication Between Humans and Machines* (280-310). Washington: National Academy Press.

Wiren, J. & Stubbs, H. (1956). Electronic binary selection system for phoneme classification. *J. Acoustic Soc. Ame.*, *28*(6), 1082-1091.

Wolf, J. & Woods, W. (1977). The HWIM speech understanding system. *In ICASSP*, 784-787.

Woodland, P. C. (1990). Isolated word speech recogniton based on connectionist techniques. *Br. Telecom. Technol. J.*, *8*(2), 61-66.

Yılmaz, C. (1999). A large vocabulary speech recognition system for Turkish. M. S. Thesis. Bilkent University, Ankara.

Young, S. & Kershaw, D. (2000). *The HTK book* (3rd ed.). Cambridge: Cambridge University Press.

Zhuang, L., Bao, T., Zhu, X.,Wang, C. & Naoi, S. (2004). A chinese OCR spelling check appoarch based on statistical language models. *IEEE International Conference on Systems, Man and Cybernetics*. 4727-4732.

APPENDIX A
DICTIONARY WORDS

Table A.1 shows the system words which we have used for training and testing of the applications.

Table A.1 The system dictionary words.

abajur	boyutlandırmak	elektrikçi	mükemmeliyet
abaküs	burun	elektroteknik	mütemadiyen
aceleci	bülten	endüstrileşme	naftalin
acemice	can	evcilleştirme	nakliyat
acımasız	caz	faks	nefeslenmek
acil	cesaretli	fark	neşelendirmek
aç	cevizli	farklılaştırma	neşeli
açıklamak	cezalı	faydalı	nicelemek
adaletsizlik	coğrafya	felaketzede	nitelendirmek
adapazarı	cumhuriyet	felek	not
ağaçlandırmak	çabalamak	ferman	nur
ahşap	çabuklaştırmak	feza	of
ak	çağla	fındık	ok
akarsu	çakır	fikir	oksijen
akça	çal	fiyasko	okuryazarlık
akıcılık	çalgı	fotokopi	organizasyon
akıllanmak	çalım	gazetecilik	ormanlık
akreditasyon	çalışkan	gecekondu	ot
aks	çalışmak	habersiz	pim
aktar	çam	halıcı	plak
akvaryumculuk	çay	hareketli	plan
alçı	çekim	ihtiyarlamak	prens
alt	çeşitlilik	inandırma	programcılık
anıtlaştırmak	çiçekçilik	iştahsızlık	radyoelektrik
aydın	çimenlik	iyotlu	radyoloji
baba	çobanpüskülü	izcilik	renk
badanacılık	dalgınlaşma	kabataş	resimlendirme
bağlam	damga	kafkasyalı	rey
bahçıvan	danışmak	kahramanlık	ring
bahçıvanlık	dansimetre	kalorimetre	risk
bakla	dargın	kamburlaştırmak	robotlaştırmak
bal	dayanışma	kan	sabunlaştırmak
baltık	defter	kap	samimiyetlik
bar	deha	kapitülasyon	sevindirmek
bardak	delgi	karikatürcü	simülasyon
basamaklı	demokrasi	kılavuz	siyasetname
başarısızlık	deneme	kitabevi	sosyoloji

başbakanlık	denetleyici	kundura	şekerleme
belirlemek	denizaltı	lösemi	tank
beneklenmedi	denizyıldızı	macun	tarz
benzerlik	dert	maç	taş
benzeşim	dev	maden	tatbikat
bereket	divan	mafya	termodinamik
biçimsel	doğru	maharet	uygarlaştırmak
bilet	doksan	makas	ücretlendirme
bilimsel	doktor	malümat	yabancılık
biyosfer	durak	mart	yaz
boncuk	ehliyet	mat	ziyaret
borç	eldiven	mert	ziyaretçi
bordo	elektrik	misafirlik	zor