A LARGE VOCABULARY SPEECH RECOGNITION SYSTEM FOR TURKISH

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

by

Cemal Yılmaz

August, 1999

TK
7895
.S65
Y55
1999

A LARGE VOCABULARY SPEECH RECOGNITION SYSTEM FOR TURKISH

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

by

Cemal Yılmaz

August, 1999

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Enis Çetin (Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mübeccel Demirekler

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Uğur Güdükbay

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Science

TK
7895
.C65
T55
1993
B 0 4 9 0 4 8

ABSTRACT

A
LARGE VOCABULARY SPEECH RECOGNITION SYSTEM
FOR TURKISH

Cemal Yılmaz
M.S. in Computer Engineering and Information Science
Supervisors: Assoc. Prof. Dr. Kemal Oflazer
and Prof. Dr. A. Enis Çetin
August, 1999

This thesis presents a large vocabulary isolated word speech recognition system for Turkish.

The triphones modeled by three-state Hidden Markov Models (HMM) are used as the smallest unit for the recognition. The HMM model of a word is constructed by using the HMM models of the triphones which make up the word. In the training stage, the word model is trained as a whole and then each HMM model of the triphones is extracted from the word model and it is stored individually. In the recognition stage, HMM models of triphones are used to construct the HMM models of the words in the dictionary. In this way, the words that are not trained can be recognized in the recognition stage.

A new dictionary model based on trie structure is introduced for Turkish with a new search strategy for a given word. This search strategy performs breadth-first traversal on the trie and uses the appropriate region of the speech signal at each level of the trie. Moreover, it is integrated with a pruning strategy to improve both the system response time and recognition rate.

Keywords: Speech recognition, triphones, Hidden Markov Model (HMM), trie-based dictionary model, trie-based search strategy

ÖZET

TÜRKÇE İÇİN GENİŞ SÖZCÜK DAĞARCIKLI KONUŞMA TANIMA SİSTEMİ

Cemal Yılmaz
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Doç. Dr. Kemal Oflazer
ve Prof. Dr. A. Enis Çetin
Ağustos, 1999

Bu tezde Türkçe için konuşmacıya bağımlı, geniş sözcük hazneli konuşma tanıma sistemi sunulmaktadır.

Bu sistemde sözcükler üçlü-fon temelli Saklı Markov Modeller ile modellenir. Bu üçlü-fonlar sistemdeki en küçük birimlerdir. Her sözcük için bu üçlü-fonların modellerinin yardımı ile sözcük modeli oluşturulur. Öğrenme safhasında, bu model bir bütün olarak eğitildikten sonra herbir üçlü-fon modeli ayrı olarak saklanır. Tanıma safhasında ise bir sözcük için gerekli olan üçlü-fon modelleri sırasıyla birbirlerine eklenerek o sözcük için gerekli model oluşturulur. Böylece öğrenme safhasında herhangi bir üçlü-fon için oluşturulan model tanıma safhasında birden fazla sözcüğün tanınması için kullanılabilir. Diğer bir deyişle, öğrenme safhasında sisteme öğretilen sözcüklerden daha fazla sözcük tanıma safhasında tanınabilir.

Bu tezde ayrıca Türkçe için "trie" yapılı sözlük modeli geliştirilmiştir. Sözlük modelinde kullanılmak üzere "trie" yapısının her düzeyinde konuşma işaretinin en uygun kısmını kullanan bir arama stratejisi geliştirilmiştir. Aynı zamanda, bu arama stratejisi sistemin tepki süresini azaltmak için arama uzayını azaltan bir strateji ile birleştirilmiştir.

Anahtar Sözcükler: Konuşma tanıma, üçlü-fonlar, Saklı Markov Modeli, "trie" yapılı sözlük modeli, "trie" yapılı arama stratejisi

ACKNOWLEDGMENTS

Foremost, I would like to express my sincerest gratitude to Prof. Enis Çetin for his continuous guidance, his precious suggestions, and his ongoing support throughout all the stages of this thesis. His assistance was particularly useful due to the fact that my advisor was abroad.

I am very grateful for the long distance support that my advisor, Assoc. Prof. Kemal Oflazer, promptly gave me. I felt his helping hands were always close to me.

I want to thank Prof. Mübeccel Demirekler and Asst. Prof. Uğur Güdükbay, the members of my jury, for reading and commenting on the thesis.

I also appreciate the work that the three trainers, Ayça Özger, Özgür Barış, and Önder Karpat, contributed to this thesis.

Lastly, I feel a final word must go to my family for all their deep love and care. In particular, I want to thank my brother Can for making me laugh even at the most difficult times and for giving me the inspiration to carry on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Speech is the most natural way of communication among human beings. Therefore the use of speech in the communication with the computers are important from the point of human beings. The production and the recognition of the speech by computers have been an active research area for years [18]. The widespread use of speech communication machines like telephone, radio, and television has given further importance to speech processing. The advances in digital signal processing technology lead the use of speech processing in many different application areas like speech compression, enhancement, synthesis, and recognition. In this thesis, the problem of speech recognition is considered and a speech recognition system is developed for Turkish.

Considerable progress has been made in speech recognition in the past 15 years. Many successful systems have emerged (see [2] and [16]). The difficulties in the speech recognition systems can be observed in four dimensions: (1) speaker dependency (speaker dependent or independent), (2) the type of utterance (continuous or isolated), (3) the size of the vocabulary (small, medium, or large vocabulary), and (4) the noise present in the environment.

In a speaker dependent speech recognition system, single speaker is used to train the system and the system should be used specifically for recognizing trainer's speech. Such systems can also recognize the speech of other speakers with possibly very high error rate. A speaker independent system is trained

with multiple speakers (including both males and females) and then they are used to recognize the speech of many speakers including those who may not have trained the system.

In a continuous speech recognition system, speakers utter the sentences in a most natural manner like in the real life. In this case, the difficulty is to detect the boundaries in the speech signal [7] and to model the coarticulatory effects and sloppy articulation [17]. However, in an isolated word recognition system, speakers must pause between the words. Therefore, finding the boundaries of the words is relatively easy when compared to the continuous utterances.

Based on their vocabulary size, speech recognition systems can be divided into three main categories: *small*, *medium*, or *large* vocabulary systems. Small vocabulary systems typically have 2-99 words, medium vocabulary systems have 100-999 words while large vocabulary systems have over 1000 words. As the vocabulary size increases, the number of confusable words increases and this leads to degraded performance [2].

In this thesis,

- a speaker dependent, large vocabulary, isolated word speech recognition system for noise-free environments is developed for Turkish,
- a new dictionary model for Turkish speech recognition systems which allows for fast and accurate search algorithm is presented, and
- a new search strategy for Turkish which improves both the system response time and recognition rate is introduced.

Chapter 2 introduces the speech processing techniques used in this thesis to (1) extract feature parameters characterizing the speech signal, and (2) find the end points of a discrete utterance.

Chapter 3 reviews the Hidden Markov Models (HMM) used in the speech recognition problem. The HMM approach is a statistical method of characterizing the spectral properties of the frames of a pattern. The underlying assumption of the HMM in speech recognition is that the speech signal can be

well characterized as a parametric random process and the parameters can be determined in a precise and well-defined manner.

Designs of the word model and the dictionary model are challenging problems in large vocabulary speech recognition systems. They must be designed in a manner allowing fast and accurate search over the dictionary. Chapter 4 introduces the design details of the word and dictionary model. It also gives a brief description of recognition problems that may occur in Turkish. The word and dictionary models are designed according to the needs of the Turkish relevant to the speech recognition problem. The triphone based HMM models are used in modeling the words [10]. The dictionary model is based on the trie structure in which each node contains a triphone.

Chapter 5 introduces the techniques and algorithms used in the training and recognition stage. In the training stage, the HMM model of a word is constructed and trained as a whole then the individual triphone models which make up the word are saved individually. In the recognition stage, these triphone models are used to construct the HMM models of the words in the dictionary then the Viterbi algorithm is applied to these models with the feature sequence fed to the system. The details of the search strategy for a word in the dictionary are also discussed in Chapter 5.

Chapter 6 gives the conclusions and the directions for future research.

Chapter 2

Speech Processing

Speech processing techniques are employed in several different application areas such as compression, enhancement, synthesis and recognition [17]. In isolated speech recognition, such techniques are used to process the speech signal to (1) extract feature vectors characterizing the speech signal, and (2) determine the endpoints of a discrete utterance.

This chapter introduces the feature extraction and end point detection algorithms in Sections 2.1 and 2.2, respectively.

2.1 Feature Extraction

A key assumption made in the design of most speech recognition systems is that the segment of a speech signal can be considered as stationary over an interval of few milliseconds. Therefore the speech signal can be divided into blocks which are usually called frames. The spacing between the beginning of two consecutive frames is in the order of 10 m-secs, and the size of a frame is about 25 m-secs. That is, the frames are overlapping to provide longer analysis windows. Within each of these frames, some feature parameters characterizing the speech signal are extracted. These feature parameters are then used in the training and recognition stage.

In the past, several parameters were used to model the speech signals. The Linear Prediction (LP) coefficients are the earliest feature parameters [17]. Unfortunately, their performance were not good so other feature parameters like Line Spectral Frequencies (LFS's) were introduced [19]. Nowadays, Mel Cepstral (MELCEP) coefficients [17] and sub-band cepstrum coefficients [17] have become the most widely used parameters for modeling the speech signals.

In this thesis, a new set of feature parameters proposed in [12] is used. The new set of feature parameters is obtained from the cepstral coefficients derived from multirate sub-band analysis of the speech signals. In the computation of the cepstral coefficients, the ordinary energy operator given in Equation 2.2 is used while in the computation of the new feature parameters, a new energy measure proposed in [12] based on Teager Energy Operator (TEO) is employed (Equation 2.1). This section gives a brief overview of the computation of the TEO based cepstrum coefficients or TEOCEP's. The details can be found in [12].

The discrete TEO [22] is defined as

$$\Psi\left[s(n)\right] = s^2(n) - s(n+1)s(n-1) \tag{2.1}$$

where $s(n)$ is the speech sample at time instant n. This operator makes successive samples exchange information between each other whereas the ordinary energy operator defined as

$$\xi\left[s(n)\right] = s^2(n) \tag{2.2}$$

treats the samples individually. In this thesis, TEO is used in frequency sub-bands. In other words, the Teager energies of sub-band signals corresponding to the original signal are estimated and they are used in the computation of feature parameters. TEO based cepstrum coefficients are then computed using the Teager energy values of the sub-band signals obtained via wavelet (or multiresolution) analysis of the original speech signal.

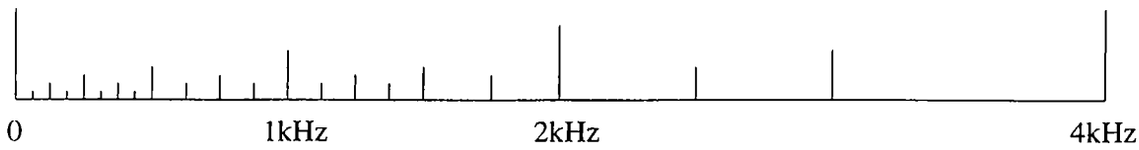

Figure 2.1. The sub-band frequency decomposition of the speech signal.

The decomposition of the speech signal into sub-band signals is described in detail in [12]. In this thesis, the speech signal, $s(n)$, is decomposed into $L = 21$ sub-signals, $\{s_l(n)\}_{l=1}^{L}$, each of which is associated with one of the bands in the frequency domain. The frequency content of the sub-signals are shown in Figure 2.1. The sub-band decomposition is almost the same as mel-scale. In other words, more emphasis is given to the low frequencies compared to the high frequencies. For each sub-signal, the average Ψ-energy e_l is computed as follows:

$$e_l = \frac{1}{T_l} \sum_{n=1}^{T_l} \left| s_l(n)^2 - s_l(n-1)s_l(n+1) \right|, \quad l = 1, 2, \ldots, L \tag{2.3}$$

where T_l is the number of samples in the l^{th} sub-band.

Log compression and Discrete Cosine Transformation (DCT) are applied to get TEO based cepstrum coefficients as follows:

$$TC(k) = \sum_{l=1}^{L} \log(e_l) \cos\left(\frac{k(l-0.5)\pi}{L}\right) \quad k = 1, 2, \ldots, 12 \tag{2.4}$$

where $TC(k)$ is the k^{th} TEO based cepstrum coefficient. The first 12 TEO based cepstrum coefficients form the first part of the feature parameters set. From the application of first-order differentials another 12 coefficients are obtained. As a result, a 24-element feature vector consisting of $TC(k)$ and differentials is extracted from each frame of the speech signal. These vectors are also called observation vectors in speech recognition terminology.

2.2 End Point Detection

Locating the endpoints of a discrete utterance is an important problem in isolated word speech recognition systems. A robust endpoint detection algorithm significantly improves the recognition rate of the overall systems.

The problem of detecting endpoints would seem to be relatively trivial, but in fact, it has been found to be very difficult in practice [2]. Usually the failure in endpoint detection is caused by: weak fricatives (/f/, /h/) or voiced fricatives that become unvoiced at the end of a word like in the Turkish word "yoz", weak plosives at either end (/p/, /t/, /k/), and nasals at the end like in the Turkish word "dam".

The wavelet analysis associated with a sub-band decomposition of the speech signal is used in the endpoint detection algorithm [11]. Consider the decomposition of speech into L sub-signals as discussed in Section 2.1. The energy parameter E_l^k is defined for k^{th} speech frame and l^{th} sub-band as follows:

$$E_l^k = \frac{1}{T_l} \sum_{n=1}^{T_l} s_l^2(n), \qquad l = 1, \ldots, L \tag{2.5}$$

where T_l is the number of samples in the l^{th} sub-band. T_l is smaller than the number of samples in the speech frame, if multirate processing is employed during sub-band decomposition. The distance measure D_k is then defined as

$$D_k = 10 \log \left[\frac{1}{L} \sum_{l=1}^{L} \frac{(E_l^k - \mu_l)^2}{\sigma_l^2} \right] \tag{2.6}$$

where μ_l and σ_l are the mean and variance of the background noise at the l^{th} sub-band, respectively. The speech free segments are used in the computation of μ_l and σ_l as follows:

$$\mu_l = \frac{1}{T} \sum_{k=1}^{T} E_l^k \tag{2.7}$$

$$\sigma_l^2 = \frac{1}{T}\sum_{k=1}^{T}(E_l^k - \mu_l)^2, \qquad l = 1, 2, \ldots, L. \tag{2.8}$$

Figure 2.2 shows the flowchart of the endpoint detection algorithm. In this flowchart, T_b , T_l, and T_h represent the beginning threshold, the lower threshold, and the higher threshold values, respectively. The lower and higher threshold values are defined in terms of the beginning threshold value as follows:

$$T_l = \frac{3}{4}T_b, \tag{2.9}$$

and

$$T_h = \frac{3}{2}T_b. \tag{2.10}$$

In Figure 2.2, N_b and N_e are the indices of the beginning and ending frame of a word in a discrete utterance, respectively, N_f is the maximum number of frames that the distance measure D_k stays below the beginning threshold T_b.

The algorithm scans the speech signal frame by frame and it marks the frame as N_b at which the threshold value T_b is first exceeded unless the distance measure D_k falls below the threshold value T_l before exceeding the threshold value T_h. After labeling a frame as N_b, the ending frame N_e is determined when the distance measure D_k falls below the threshold value T_b for longer than N_f frames.

The threshold value T_b can be predetermined by the help of the histogram of the distance measure D_k (Equation 2.6). The performance of this end-point detection algorithm is significantly influenced by the choice of N_f value. If N_f is large then the algorithm may (1) miss the silent region between two consecutive words; these words are treated as one word, or (2) treat silent regions at the end of the utterances as speech regions. On the other hand, if N_f is small then the algorithm may treat the utterance of one word as multiple words. To overcome these problems, another parameter, N_d, is introduced and N_f value is chosen as small as possible. After detecting end-points of a discrete utterance, if the number of frames between the ending frame of a word and the beginning frame of the consecutive word is smaller than N_d then the end-points are changed so that these consecutive words are treated as one word.

Therefore, the distance between two consecutive words must be larger than N_d frames.

The algorithm given in Figure 2.2 can be used to determine the end-points of a discrete utterance both at the time of recording and at any time after the recording is finished.

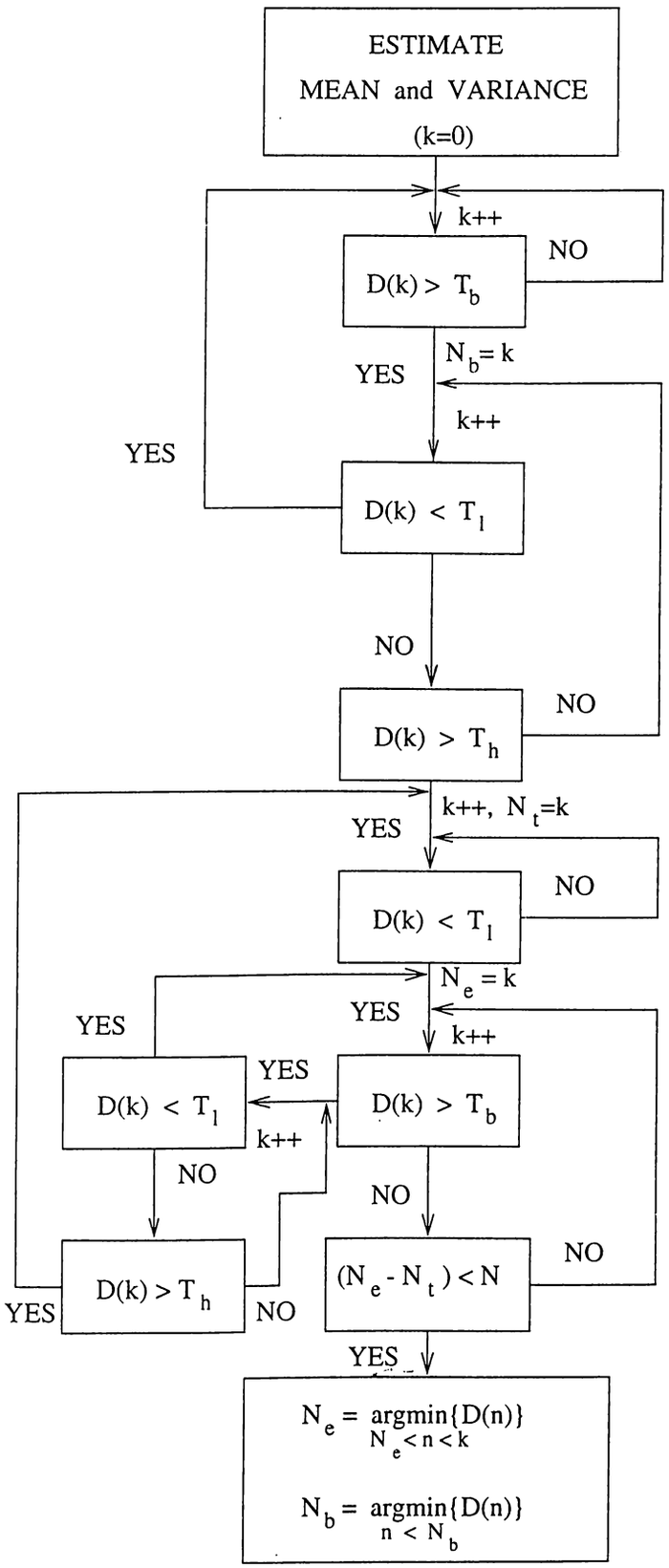

Figure 2.2. Flow chart of the end point detection algorithm.

Chapter 3

Hidden Markov Model (HMM) for Speech Recognition

The speech recognition problem can be considered as mapping the sequence of feature vectors to a word in the dictionary. Starting from the late 1960s, researchers focused on developing stochastic models for speech signals. The reason they used the probabilistic modeling was to address the problem of variability. Today, Hidden Markov Model (HMM) and Artificial Neural Networks (ANN) are the two main approaches used in speech recognition research. The HMM approach is adopted in this thesis. The use of HMM models for speech recognition applications began in 1970s [21].

An HMM model is a finite state machine that changes state at every time unit as shown in Figure 3.1. At each discrete time instant t, transition occurs from state i to j , and the observation vector $\mathbf{o}_t$ is emitted with the probability density $b_j(\mathbf{o}_t)$. Moreover the transition from state i to j is also random and it occurs with the probability a_{ij}.

The underlying assumption of an HMM model in speech recognition problem is that a speech signal can be well characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise and well-defined manner. An HMM model is considered as a generator of observation sequences (observation and feature sequences are used to refer

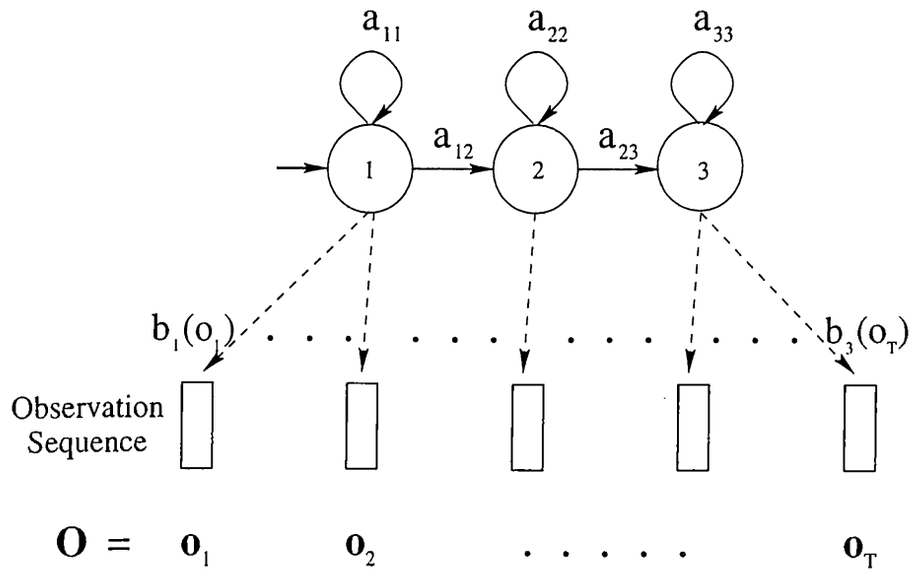

Figure 3.1. A three-state left to right HMM model with the observation vectors each being generated by one state (state 1 represents the start state).

the same thing in the rest of this thesis). In practice, only the observation sequence is known and the underlying state sequence is hidden. That is why this structure is called a Hidden Markov Model. This chapter introduces the theory of HMM models for speech recognition purpose.

3.1 Elements of an HMM

A complete specification of an HMM model requires specification of (1) two model parameters, N and M, (2) observation symbols, and (3) three sets of probability measures A, B, π. The definitions of these parameters are as follows:

1. The parameter, N, is the number of states in the HMM. The individual states are labeled as $\{1, 2, \ldots, N\}$, and the state at time t is denoted as q_t.

2. The parameter, M, is the number of distinct observation symbols per state. The observation symbols represent the physical output of the system being modeled. The individual observation symbols are denoted

as $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\}$. Only in HMM models for discrete observation symbols the parameter M is defined. HMM models for continuous observation sequences, as we have in this thesis, clearly do not have the parameter M, but have an observation set whose elements are continuous variables.

3. The matrix, $A = \{a_{ij}\}$, is the state transition probability distribution where a_{ij} is the transition probability from the state i to j, i.e.,

$$a_{ij} = P(q_{t+1} = j \mid q_t = i), \quad 1 \leq i, j \leq N \tag{3.1}$$

If a state j can not be reached by a state i in a single transition, we have $a_{ij} = 0$ for all i, j.

4. Let $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$ be the set of observation symbols. The matrix, $B = \{b_j(\mathbf{o}_t)\}$, is the observation symbol probability distribution in which

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t \mid q_t = j), \quad 1 \leq t \leq T \tag{3.2}$$

defines the symbol distribution in state j, $j = 1, 2, \ldots, N$. In speech recognition problem, observation symbols are feature parameter vectors.

5. The vector, $\pi = \{\pi_i\}$, is the initial state distribution, in which

$$\pi_i = P(q_1 = i), \quad 1 \leq i \leq N. \tag{3.3}$$

For convenience, we use the compact notation

$$\lambda = (A, B, \pi) \tag{3.4}$$

to indicate the complete parameter set of an HMM model. This parameter set defines a probability measure for a given observation sequence $\mathbf{O}$, i.e., $P(\mathbf{O} \mid \lambda)$. We use HMM model to indicate the parameter set λ and the associated probability measure interchangeably without ambiguity.

3.2 Probability Evaluation

One basic problem of HMM models is to calculate the probability of the observation sequence, $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots \mathbf{o}_T)$, given the HMM model $\lambda = (A, B, \pi)$. The trivial way of solving this problem is through enumerating every possible state sequence of length T (number of observations). Clearly, there are at most N^T such state sequences. For one such fixed-state sequence

$$\mathbf{q} = (q_1 q_2 \ldots q_T) \tag{3.5}$$

where q_1 and q_T are the initial and the final states respectively, the probability of the observation sequence $\mathbf{O}$ is

$$P(\mathbf{O} \mid \mathbf{q}, \lambda) = \prod_{t=1}^{T} P(\mathbf{o}_t \mid q_t, \lambda). \tag{3.6}$$

In the equation above, the statistical independence of observations is assumed. In other words we have

$$P(\mathbf{O} \mid \mathbf{q}, \lambda) = b_{q_1}(\mathbf{o}_1).b_{q_2}(\mathbf{o}_2) \ldots b_{q_T}(\mathbf{o}_T). \tag{3.7}$$

Moreover, the probability of the state sequence $\mathbf{q}$ can be given as follows:

$$P(\mathbf{q} \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}. \tag{3.8}$$

Then the probability that $\mathbf{O}$ and $\mathbf{q}$ occur simultaneously is simply the product of the two terms above, that is,

$$P(\mathbf{O}, \mathbf{q} \mid \lambda) = P(\mathbf{O} \mid \mathbf{q}, \lambda) P(\mathbf{q} \mid \lambda). \tag{3.9}$$

The probability of the observation sequence $\mathbf{O}$ given the model is obtained by summing the Equation 3.9 over all possible state sequences q, which is

$$\begin{aligned} P(\mathbf{O} \mid \lambda) &= \sum_{q_1,q_2,\ldots,q_T} P(\mathbf{O} \mid \mathbf{q}, \lambda) P(\mathbf{q} \mid \lambda) \\ &= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \ldots a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T). \quad (3.10) \end{aligned}$$

The interpretation of the Equation 3.10 is the following: At time $t = 1$ we are in state q_1 with probability π_{q_1}, and generate the symbol $\mathbf{o}_1$ with probability $b_{q_1}(\mathbf{o}_1)$. The clock changes from time t to time $t + 1$ and we make transition from state q_1 to state q_2 with probability $a_{q_1 q_2}$ and generate the symbol $\mathbf{o}_2$ with probability $b_{q_2}(\mathbf{o}_2)$. This computation continues until we make the last transition, at time T, from state q_{T-1} to q_T and generate the output symbol $\mathbf{o}_T$.

It is clear that the calculation of $P(\mathbf{O} \mid \lambda)$ by using its direct definition (Equation 3.10) involves on the order of $2TN^T$ calculations. This computational complexity is infeasible even for small values of N and T. For instance, for $N = 5$ and $T = 100$, there are $2 \cdot 100 \cdot 5^{100} \sim 10^{72}$ computations. Therefore, more efficient algorithms are needed.

The *backward procedure* and *forward procedure* are recursive methods of performing this calculation [1]. The crucial point of these algorithms is that each of them allows the calculation of the probability of a given state at a given time.

3.2.1 The Forward Procedure

Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2 \ldots \mathbf{o}_t, q_t = i \mid \lambda) \tag{3.11}$$

that is, as the probability of the partial observation sequence, $\mathbf{o}_1\mathbf{o}_2 \ldots \mathbf{o}_t$, and state i at time t, given the model λ. We can solve Equation 3.11 for $\alpha_t(i)$ inductively as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N. \tag{3.12}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(\mathbf{o}_{t+1}), \qquad \begin{matrix} 1 \leq t \leq T-1, \\ 1 \leq j \leq N \end{matrix}. \tag{3.13}$$

3. Termination

$$P(\mathbf{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{3.14}$$

As can be seen from this iterative solution, the computation of $P(\mathbf{O} \mid \lambda)$ requires on the order of N^2T calculations, rather than $2TN^T$ as required by the direct calculation in Equation 3.10.

3.2.2 The Backward Procedure

In a similar manner, the backward variable $\beta_t(i)$ can be defined as

$$\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2} \ldots \mathbf{o}_T | q_t = i, \lambda) \tag{3.15}$$

that is, as the probability of the partial observation sequence from $t+1$ to the end, given state i at time t and the model λ. Again we can solve for $\beta_t(i)$ inductively, as follows:

1. Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N. \tag{3.16}$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \qquad \begin{array}{l} t = T-1, T-2, \ldots, 1, \\ 1 \leq i \leq N. \end{array} \tag{3.17}$$

Again the computation of $\beta_t(i)$ requires on the order of N^2T calculations like the forward procedure.

Both the backward procedure and forward procedure are used in "optimal" state sequence computation and parameter estimation algorithm in Section 3.3 and Section 3.4, respectively.

3.3 The "Optimal" State Sequence

An important problem in HMM model formulation is the estimation of the optimal state sequences. There are several ways to find the optimal state sequence associated with the given observation sequence. Various optimality criteria can be defined.

Our optimality criterion is based on finding the state sequence which maximizes $P(\mathbf{q} \mid \mathbf{O}, \lambda)$. It is equivalent to maximizing $P(\mathbf{q}, \mathbf{O} \mid \lambda)$ due to Bayes' rule [9].

The solution is given by the Viterbi algorithm which is essentially a dynamic programming method [23]. Consider the Viterbi variable $\delta_t(i)$ defined as follows:

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t | \lambda) \tag{3.18}$$

where $\mathbf{q} = (q_1 q_2 ... q_T)$ is the best state sequence for the given observation sequence $\mathbf{O} = (\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_T)$. In other words, $\delta_t(i)$ is the highest probability along

a single path, at time t, which accounts for the first t observations and ends in state i. The recursive version of $\delta_t(i)$ can be written as

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i)a_{ij}\right] b_j(\mathbf{o}_{t+1}). \tag{3.19}$$

The recursive procedure to find the single best state sequence is as follows:

1. Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N \tag{3.20}$$

$$\psi_1(i) = 0 \tag{3.21}$$

where $\psi_t(j)$ is used to keep track of the arguments that maximize the Equation 3.23 for each t and j.

2. Recursion

$$\delta_t(j) = \max_{1\leq i\leq N} [\delta_{t-1}(i)a_{ij}]\, b_j(\mathbf{o}_t), \qquad \begin{matrix} 2 \leq t \leq T, \\ 1 \leq j \leq N \end{matrix} \tag{3.22}$$

$$\psi_t(j) = \arg\max_{1\leq i\leq N} [\delta_{t-1}(i)a_{ij}], \qquad \begin{matrix} 2 \leq t \leq T, \\ 1 \leq j \leq N. \end{matrix} \tag{3.23}$$

3. Termination

$$P^* = \max_{1\leq i\leq N} [\delta_T(i)] \tag{3.24}$$

$$q_T^* = \arg\max_{1\leq i\leq N} [\delta_T(i)]. \tag{3.25}$$

where P^* and the q_t^* are the maximum likelihood of observing $\mathbf{O}$ in the model λ and the state at time t in the final state sequence which results in the probability P^*, respectively.

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1. \tag{3.26}$$

As can be seen from the procedure above, Viterbi algorithm is similar to the forward procedure (Equations 3.12–3.14) except for the backtracking step.

The difference is that, the summing step in the forward procedure (Equation 3.13) is changed with the maximization criterion (Equation 3.22) in the Viterbi algorithm.

The model parameters in Equations 3.21−3.26 are very small values. The multiplications in the algorithm result in further smaller values. After a few iteration of the Viterbi algorithm, the result of these computations become 0 because of the limited representation (32-64 bit) for the floating numbers in computers. The Viterbi algorithm can be implemented without the need for any multiplication by taking the logarithms of the model parameters as follows:

0. Preprocessing

$$\tilde{\pi}_i = log(\pi_i), \quad 1 \leq i \leq N \tag{3.27}$$

$$\tilde{b}_i(\mathbf{o}_t) = \log(b_i(\mathbf{o}_t)), \quad 1 \leq i \leq N,\ 1 \leq t \leq T \tag{3.28}$$

$$\tilde{a}_{ij} = \log(a_{ij}), \quad 1 \leq i, j \leq N \tag{3.29}$$

1. Initialization

$$\begin{aligned} \tilde{\delta}_1(i) &= \log(\delta_1(i)) \\ &= \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1), \quad 1 \leq i \leq N \end{aligned} \tag{3.30}$$

$$\psi_1(i) = 0, \quad 1 \leq i \leq N \tag{3.31}$$

2. Recursion

$$\begin{aligned} \tilde{\delta}_t(j) &= \log(\delta_t(j)) \\ &= \max_{1 \leq i \leq N} \left[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\right] + \tilde{b}_j(\mathbf{o}_t) \qquad \begin{matrix} 2 \leq t \leq T, \\ 1 \leq j \leq N. \end{matrix} \end{aligned} \tag{3.32}$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} \left[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\right], \qquad \begin{matrix} 2 \leq t \leq T, \\ 1 \leq j \leq N. \end{matrix} \tag{3.33}$$

3. Termination

$$\tilde{P}^* = \max_{1 \leq i \leq N} \left[\tilde{\delta}_T(i)\right] \tag{3.34}$$

$$q_T^* = \arg \max_{1 \leq i \leq N} \left[\tilde{\delta}_T(i)\right]. \tag{3.35}$$

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1. \tag{3.36}$$

This alternative implementation of the Viterbi algorithm requires on the order of N^2T additions. The cost of preprocessing is negligible as it is performed once and saved.

3.4 Parameter Estimation

Parameter estimation is computationally the most difficult problem in HMM models. The model parameters A, B, and λ are estimated to satisfy an optimization criterion. Our optimization criterion is based on maximizing $P(\mathbf{O} \mid \lambda)$ where $\mathbf{O}$ represents the training observations. In order to do that, the Baum-Welch method also known as expectation-maximization (EM) method is used [14].

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. That is,

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{M} c_{jk} \mathcal{N}(\mathbf{o}_t, \mu_{jk}, \mathbf{U}_{jk}), \quad 1 \leq j \leq N \tag{3.37}$$

where $\mathbf{o}_t$ is the observation vector being modeled, M is the number of mixture values used for each state (in this thesis three mixture values ($M = 3$) are used for each state), $\mathcal{N}$ represents a Gaussian probability density function (pdf), and c_{jk} is the mixture coefficient for the k^{th} mixture in state j such that

$$\sum_{k=1}^{M} c_{jk} = 1, c_{jk} \geq 0 \qquad \begin{matrix} 1 \leq j \leq N, \\ 1 \leq k \leq M \end{matrix} \tag{3.38}$$

The Gaussian pdf $\mathcal{N}$ has a mean vector μ_{jk} and covariance matrix $\mathbf{U}_{jk}$ for the k^{th} mixture component in state j, that is,

$$\mathcal{N}(\mathbf{o}_t, \mu_{jk}, \mathbf{U}_{jk}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{U}_{jk}|}} e^{-\frac{1}{2}(\mathbf{o}_t - \mu_{jk})' \mathbf{U}_{jk}^{-1} (\mathbf{o}_t - \mu_{jk})} \tag{3.39}$$

where n is the dimensionality of the observation vector $\mathbf{o}_t$. In our case n is 24, as 24 feature parameters are extracted from each frame of the speech signal as discussed in Section 2.1. 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 estimates of μ_{jk} and $\mathbf{U}_{jk}$ would be simple averages as follows:

$$\widehat{\mu}_{jk} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{o}_t \tag{3.40}$$

and

$$\widehat{\mathbf{U}}_{jk} = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{o}_t - \mu_{jk})(\mathbf{o}_t - \mu_{jk})'. \tag{3.41}$$

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. Moreover, 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. However, Equations 3.40 and 3.41 can be used if some approximate assignments of observation vectors to the states could be done. Section 5.1.1 introduces the techniques used in the initial assignment of the observation vectors to the states of an HMM model.

We now define the variable $\xi_t(i, j)$ [14] to help us in 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 $\mathbf{O}$, that is

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|\mathbf{O}, \lambda). \tag{3.42}$$

The variable $\xi_t(i,j)$ can be re-written by using the definitions of the forward and backward variables (Sections 3.2.1 and 3.2.2) as follows:

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t = i, q_{t+1} = j, \mathbf{O} \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{i+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}. \end{aligned} \tag{3.43}$$

A posteriori variable $\gamma_t(i)$ [14] is defined for making the parameter estimation algorithm tractable as follows:

$$\gamma_t(i) = P(q_t = i \mid \mathbf{O}, \lambda) \tag{3.44}$$

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

$$\begin{aligned} \gamma_t(i) &= P(q_t = i \mid \mathbf{O}, \lambda) \\ &= \frac{P(\mathbf{O}, q_t = i \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{P(\mathbf{O}, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, q_t = i \mid \lambda)}. \end{aligned} \tag{3.45}$$

The equation above can be re-written by the help of forward and backward variables as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}. \tag{3.46}$$

We can relate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over j, that is,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{3.47}$$

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 as the expected number of transitions from state i to j. That is,

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i \tag{3.48}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from state } i \text{ to } j \tag{3.49}$$

If the current model is defined as $\lambda = (A, B, \pi)$ then a set of reasonable re-estimation formulas for the parameters of the model can be defined as in Equations 3.50–3.52.

$$\overline{\pi}_j = \text{expected number of times in state } i \text{ at time } t = 1 \tag{3.50}$$

$$\overline{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.51}$$

$$\overline{b}_j(\mathbf{o}_t) = \sum_{k=1}^{M} \overline{c}_{jk} \mathcal{N}(\mathbf{o}_t, \overline{\mu}_{jk}, \overline{\mathbf{U}}_{jk}), \quad 1 \le j \le N \tag{3.52}$$

In Equation 3.52, the formulas for the re-estimation of the coefficients $\overline{c}_{jk}, \overline{\mu}_{jk}$, and $\overline{\mathbf{U}}_{jk}$ are given as follows:

$$\overline{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{3.53}$$

$$\overline{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3.54}$$

$$\overline{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(\mathbf{o}_t - \mu_{jk})(\mathbf{o}_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3.55}$$

where $\gamma_t(j,k)$ is the probability of being in state j at time t with the k^{th} mixture component accounting for $\mathbf{o}_t$. That is,

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right] \left[\frac{c_{jk}\mathcal{N}(\mathbf{o}_t, \mu_{jk}, \mathbf{U}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{o}_t, \mu_{jm}, \mathbf{U}_{jm})}\right] \tag{3.56}$$

At the end of these computations, a re-estimated model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is obtained and either

- $\overline{\lambda} = \lambda$, that is $P(\mathbf{O} \mid \overline{\lambda}) = P(\mathbf{O} \mid \lambda)$, or,
- model $\overline{\lambda}$ is more likely than model λ, that is $P(\mathbf{O} \mid \overline{\lambda}) > P(\mathbf{O} \mid \lambda)$ [1].

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

In this thesis, triphones (Section 4.2) are used as the smallest unit for speech recognition. Each triphone is represented by a three-state left to right HMM model as illustrated in Figure 4.1. The algorithms described in this chapter are used on the HMM models of the triphones both in the training and recognition stage.

Chapter 4

The Language Model

Designing the word model and the dictionary model is a challenging problem in speech recognition systems. They must be made in a way that allows for fast and accurate search algorithms. Moreover, it is a good idea to construct these models according to the needs of the language.

In this thesis, a Turkish corpus is used for gathering statistical information such as the most frequently used words/triphones, minimum set of triphones which covers most of the words in the corpus, etc. This corpus is constructed by collecting newspaper text on different subjects from the Internet, and it contains about 4.2 million words of which about 300000 are unique.

Section 4.1 gives a general overview of the Turkish language from the point of view of speech recognition problem. Next, the word model is introduced in Section 4.2. Finally, Section 4.3 introduces the dictionary model adopted in this thesis.

4.1 Turkish Language

This section examines the Turkish language from the point of view of speech recognition problem.

Turkish is an agglutinative language. In agglutinative languages, bound morphemes are concatenated to a free morpheme. Each morpheme usually conveys one piece of morphological information such as tense, case, agreement, etc. The morphological structure of Turkish words has adverse impact both on recognition rate and system response time.

Let us examine the following two cases to understand the adverse impact on the recognition rate.

case 1: same stem, different suffixes
ev + (i)m
ev + (i)n

In the first case, we have the same Turkish stem "ev" but different suffixes. The inflectional suffixes **m** and **n** are frequently used suffixes in Turkish but, unfortunately, their pronunciations are similar. Although the words "evim" and "evin" are different their pronunciations are very similar.

case 2: different stems, same sequence of suffixes
ev + lerinizdekilerden
iş + lerinizdekilerden

In this case, we have different Turkish stems namely "ev" and "iş" but the same sequence of suffixes. The pronunciation of the suffix part are the same in both words. Since the suffix part is about 90% of each word, the pronunciation of the words "evlerinizdekilerden" and "işlerinizdekilerden" are very similar. Our solution to this problem is discussed in Section 5.2.2.

In Turkish, one can generate a large number of words which have the same stem but different suffixes. Therefore, even for a single stem the dictionary can contain quite large number of words with different suffixes. Increases in the size of the dictionary result in increases in the system response time.

4.2 Word Model

Many HMM-based small vocabulary or even medium vocabulary speech recognition systems assign fixed model topology to each of the words. That is, each HMM model in the system has the same number of states. In such systems, the smallest unit for recognition is the words themselves. It is impossible to use any part of the trained model of a word for the training or the recognition of different words. Such systems recognize only the words that are trained. Fixed-topology word models are not reasonable for large vocabulary speech recognition systems.

Many modern large vocabulary speech recognition systems use phonemes as the smallest unit for speech recognition. Usually one-state HMM models are used to model the phonemes of the language. In such strategies, the model of a word is obtained by cascading the models of the phonemes which make up the word.

Turkish phonemes sound differently according to the phonetic context in which they appear. Neighboring phonemes affect the pronunciation of a phoneme significantly. Therefore modeling a phoneme with its phonetic context should give better results than modeling it individually. A good way of achieving this is to model each phoneme in the context of its left and right neighboring phoneme. This is known as the triphone model. The word "bir", for instance, is represented by the following triphones

$$/sil\text{-b+i}/\ /\text{b-i+r}/\ /\text{i-r+}sil/$$

where *sil* stands for silent regions. Each phoneme in the word "bir", namely /b/, /i/, and /r/, is represented by a triphone. For example, the triphone /b-i+r/ is used to model the phoneme /i/ which has the phoneme /b/ and phoneme /r/ as the left and right neighboring phoneme, respectively. Using triphone models lead to a good phonetic discrimination [10].

Each triphone is represented by an HMM model $\lambda = (A, B, \pi)$. This model has three emitting states and a simple left-right topology [1] as illustrated in

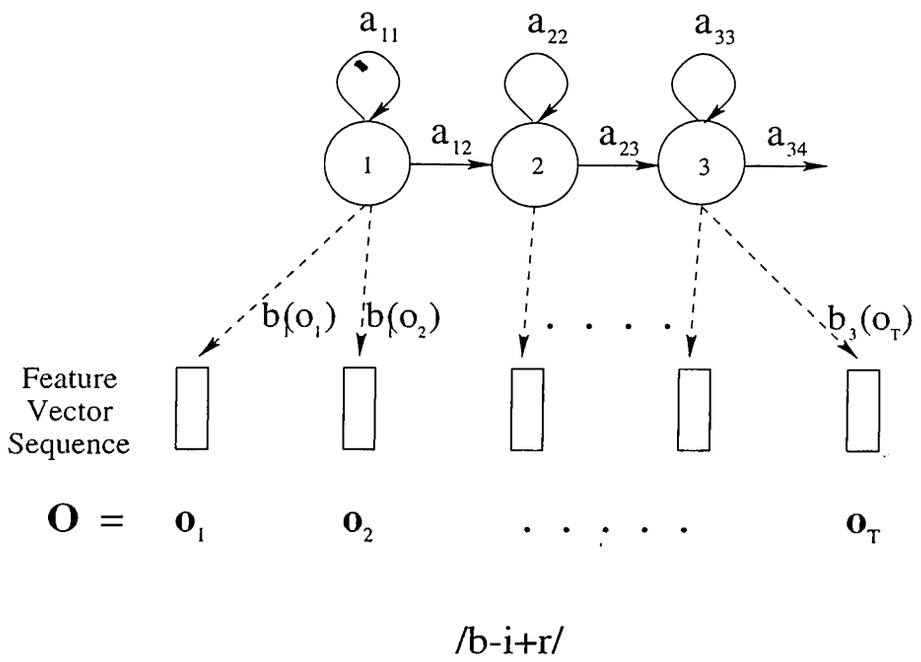

Figure 4.1. Example HMM model for the triphone /b-i+r/.

Figure 4.1. As can be seen from the figure, the HMM model has the property that, as time increases, the state index either increases by one or stays the same. This fundamental property of the topology is expressed mathematically as follows:

$$a_{ij} = 0, \quad \text{where} \quad \begin{cases} j < i, & i, j \leq N \\ i > j + 1, & i, j \leq N \end{cases} \tag{4.1}$$

The model states proceed from left to right. This topology is convenient to model the speech signal whose properties change over time in a successive manner. With this topology we have two special cases:

- Triphones which have */sil/* at the leftmost position. They represent the beginning of words. The leftmost state in the HMM model of such triphones is interpreted as the start state in the HMM model of the words which begin with these triphones.
- Triphones which have */sil/* at the rightmost position. They represent the ending of words. The last state in the HMM model of such a triphone

has another state transition constraint, $a_{NN} = 1$, in addition to the one in Equation 4.1 as the state sequence must end in state N.

For each state, a three-mixture ($M = 3$) Gaussian density is used as described in Section 3.4. The memory requirement of an HMM model for a triphone is the sum of memory space needed for (1) mixture values: $N \cdot (M \cdot (n \cdot S\_D)) = 3 \cdot (3 \cdot (24 \cdot 8)) = 1728$ where N and M is the number of mixture values used for a state and the number of states in the model, respectively, n is the dimensionality of feature vectors, and S_D is the memory space needed to store a *double* value in bytes, and (2) transition probabilities $N \cdot (N\_T \cdot S\_D) = 3 \cdot (2 \cdot 8) = 48$ where N_T is the number of transitions from a state. That is, 1776 bytes are needed to store an HMM model for a triphone.

The model of a word is obtained by cascading the models of the triphones which make up the word. Figure 4.2 illustrates the process of constructing the HMM model for the word "bir".

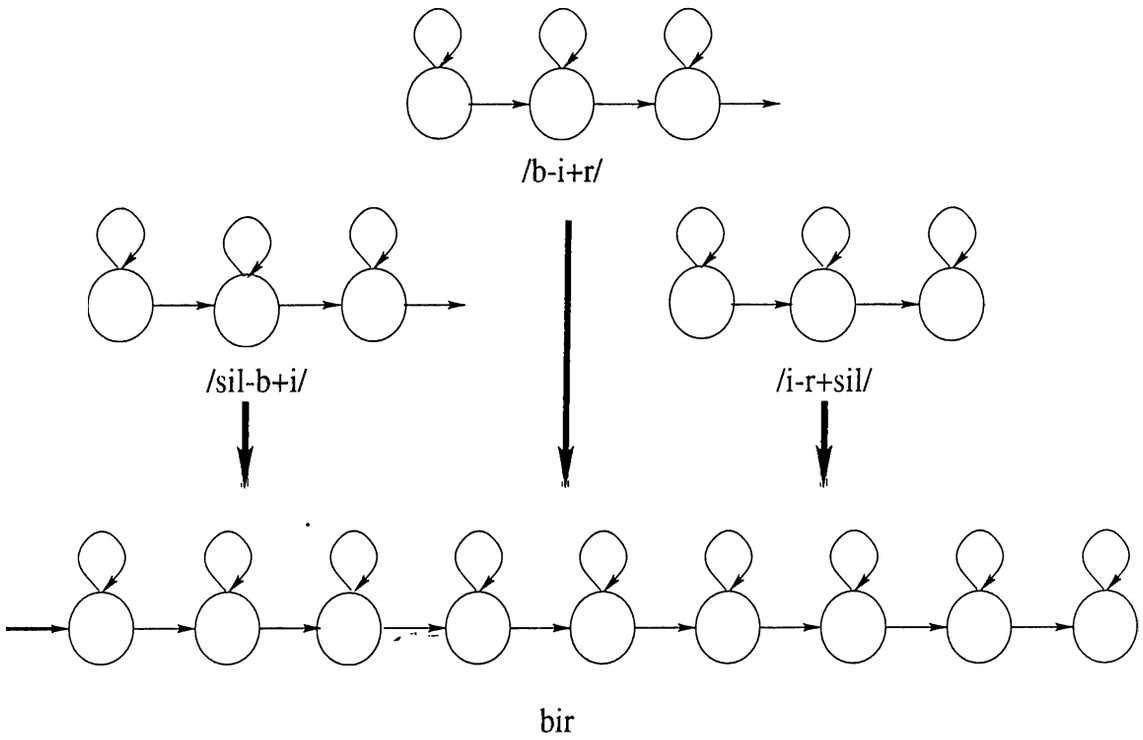

Figure 4.2. Example HMM model for the Turkish word "bir".

We have a fixed model topology assigned to each triphone (Figure 4.1). On the other hand, the size of the word model depends on the number of triphones

in the word. Each word has its own HMM model which may have different number of states.

In the training stage (Section 5.1), first, the HMM model for a training word is constructed and trained. Next, the HMM models for the triphones which make up the word are extracted and stored individually. In the recognition stage (Section 5.2), we have the reverse operation, the HMM models for the triphones of a word are concatenated to construct the HMM model λ of the word. This model is then used with the given observation sequence $\mathbf{O}$ to find $P(\mathbf{O} \mid \lambda)$.

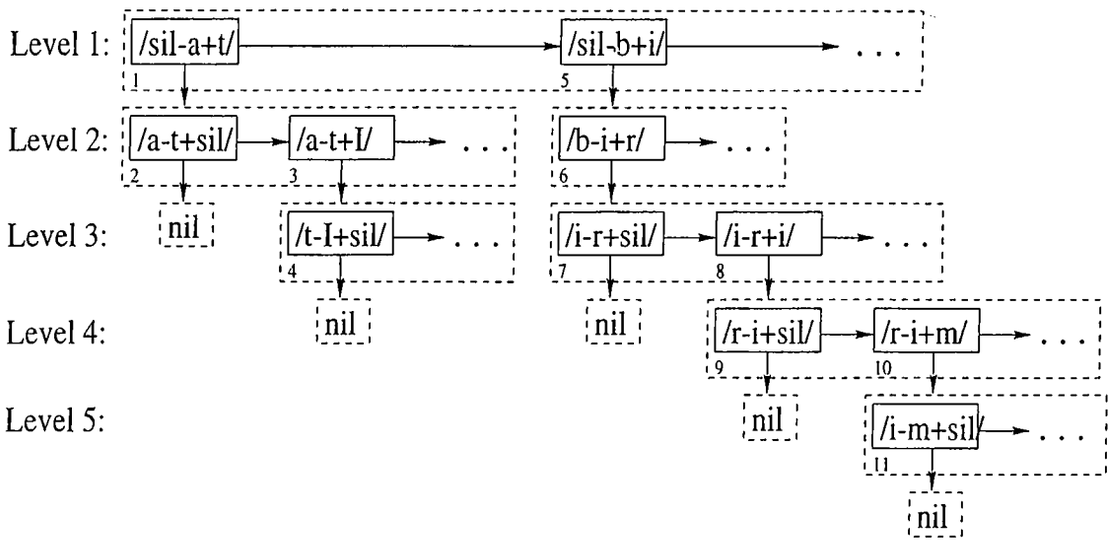

Figure 4.3. Example dictionary which contains the Turkish words "at", "atı", "bir", "biri", and "birim". Nodeboxes and nodes are represented by dashed-rectangles and solid-rectangles, respectively.

4.3 The Dictionary Model

The dictionary of a speech recognition system contains the words that can be recognized by the system. The design and implementation of the dictionary for Turkish is one of the challenging problems that was faced. The design of the dictionary must adhere to certain requirements:

- The dictionary must allow for fast and accurate search algorithms.

- Turkish morphological structure must be taken into consideration in the design of the dictionary.
- The memory requirements must be as small as possible.

To meet the requirements above the dictionary model employed in this thesis is based on trie structure [24] as illustrated in Figure 4.3. A new terminology, *nodebox*, is introduced here to make the algorithms defined on the trie tractable. A *nodebox* in the trie either contains *nil* corresponding to the termination nodebox, or it contains a linked-list of *nodes* each of which contains a triphone and a pointer to a subtrie. It should be noted that a subtrie is, in fact, a nodebox with all the descendants. The trie is constructed with standard algorithms [24].

Figure 4.3 gives an example dictionary which has the Turkish words "at", "atı", "bir", "biri", and "birim". For instance, if we follow the nodes 1 and 2, we have the Turkish word "at". It is understood that there is a valid Turkish word because each word must have silent regions both at the beginning and at the end. If we follow the nodes 1,3, and 4, we have the word "atı" which is the accusative form of "at".

Table 4.1. The number of nodes at each level of the trie.

Level	Number of Nodes	Level	Number of Nodes
1	160	11	1901
2	422	12	1350
3	828	13	865
4	1459	14	507
5	2484	15	295
6	3023	16	138
7	3501	17	77
8	2917	18	24
9	2711	19	2
10	2371		

All the triphones in the trie must be trained in the training stage. It is not possible to recognize a word containing a triphone which is not trained.

Though in the future this may be improved by allowing a self-correcting search algorithm such as the error-tolerant finite state recognition algorithm [25].

The words which are covered by the trained triphone models are chosen from the Turkish corpus and inserted into the dictionary. In this thesis, the depth of the trie is 19. Table 4.1 shows the number of nodes at each level of the trie. This table is included here in order to give a general idea about the search space.

This dictionary structure saves space because of the characteristic of trie data structure and Turkish morphological structure. Moreover it allows for fast and accurate search algorithms as discussed in Section 5.2.2.

Chapter 5

The Training and Recognition Stages

A spoken word can be represented by a sequence of observation vectors $\mathbf{O}$, defined as $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_1, \ldots, \mathbf{o}_T$ where $\mathbf{o}_t$ is the t^{th} observation vector and T is the number of observation vectors for a single word utterance. Then the problem of isolated word recognition can be defined as

$$w^* = \arg\max_i P(w_i \mid \mathbf{O}) \tag{5.1}$$

where w_i is the i^{th} word in the dictionary and w^* is the desired word. By using Bayes' rule, $P(w_i \mid \mathbf{O})$ can be expressed as follows:

$$P(w_i \mid \mathbf{O}) = \frac{P(\mathbf{O} \mid w_i)P(w_i)}{P(\mathbf{O})} \tag{5.2}$$

Prior probabilities $P(w_i)$ are taken to be equal to each other for all w_i in this thesis. Therefore the most probable spoken word depends only on the likelihood $P(\mathbf{O} \mid w_i)$. It is not feasible, for a given observation sequence $\mathbf{O}$, to directly estimate the joint probability $P(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T \mid w_i)$ for each word w_i as discussed in Section 3.4. HMM models introduced in Chapter 3 are used in this thesis so that computing $P(\mathbf{O} \mid w_i)$ is replaced by estimating the HMM

model parameters of the word w_i.

Triphones are used as the smallest unit for training and recognition of the words. The training of these models and their usages in recognition stage will be introduced in Section 5.1 and Section 5.2, respectively.

The general architecture of the speech recognition system developed in this thesis is given in Figure 5.1. As can be seen from the figure, after feature extraction step, the trained HMM models of the triphones are used to construct the HMM model of a word w supplied by the dictionary. The Viterbi algorithm is then applied to the HMM model with the feature vectors to get the probability $P(w \mid \mathbf{O})$. Later, in Section 5.2.2, this architecture is powered with a search strategy for a word in the dictionary.

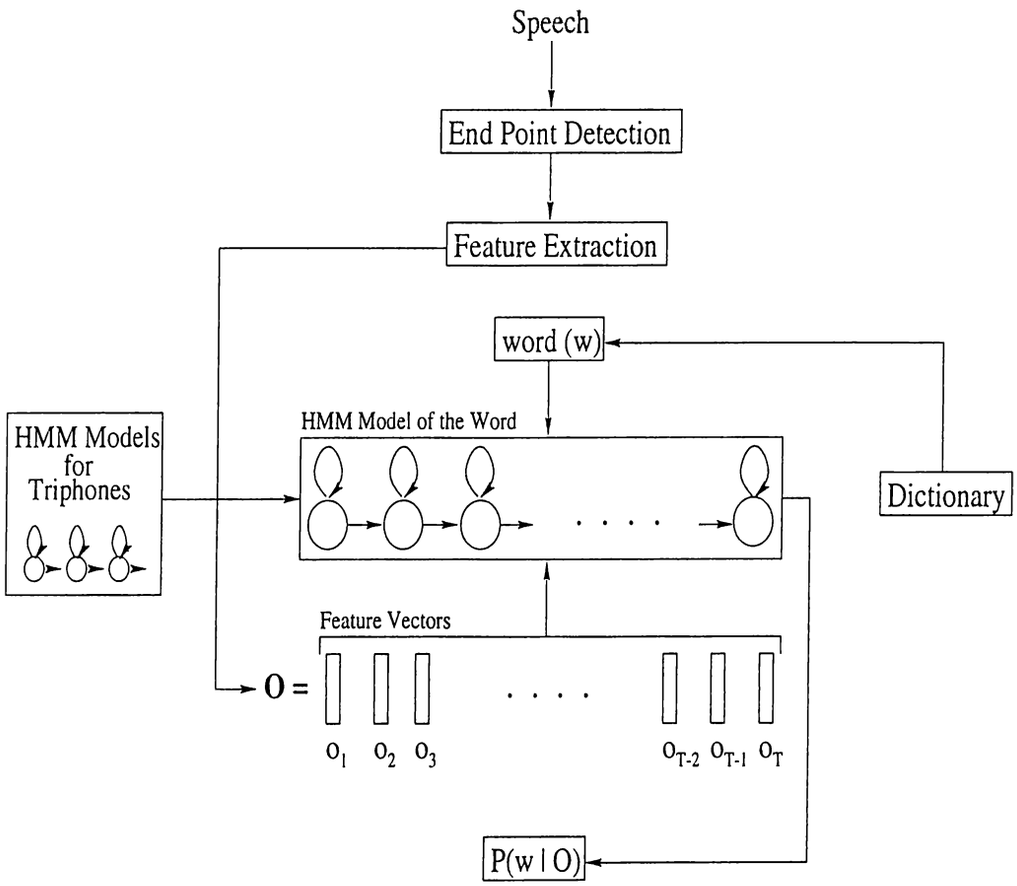

Figure 5.1. General architecture of the system.

5.1 The Training Process

Several training algorithms for HMM models were introduced in [20], [3], [6], and [8]. The training algorithm used in this thesis has 4 steps for a given word:

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

In this way, the words that are not trained, can be recognized. For instance, once the word "okul" is trained, we have four HMM models for the triphones /*sil*-o+k/, /o-k+u/, /k-u+l/, and /u-l+*sil*/. In the recognition stage, we can use the models for the triphones /k-u+l/ and /u-l+*sil*/ to construct the HMM model of the word "kul" (triphones for "kul" is /*sil*-k+u/, /k-u+l/, and /u-l+*sil*/) assuming that the triphone model for /*sil*-k+u/ is also trained. Therefore, the word "kul" does not need to be trained.

In the algorithm above and throughout this section, it is assumed that an HMM model is trained by using single utterance of a word. However, in order to get good HMM models three utterances of a word are used. The use of multiple observation sequences adds no additional complexity to the algorithm above. Step 3 is simply repeated for each distinct training sequence.

The Turkish corpus is used to determine the words that will be trained. One of the strategies to choose the words for training is to select the minimum number of words whose triphones cover most of the words in the corpus. This strategy favors the long words in the corpus. Simulation results show that using triphone models obtained by training long words has negative effects on recognition rate because it is usually difficult to pronounce too long words. Therefore a constraint on the length of the words is also used in the selection of the words. This strategy allows us to find about 1200 words (of maximum ten letters long) whose triphones cover 90% percent of the words in the corpus. That is, if we train about 1200 words, we can theoretically include 90% percent of the words in the corpus.

In order to increase the recognition rates of the most frequently used words and have a manageable dictionary size, a second strategy is introduced to select the words. This strategy simply chooses the most frequently used words in the corpus for training.

For the rest of this section, assume that a training word has T frames in the speech signal, N_TP triphones from which N_V of them contains a vowel or semi vowel in the middle, N states, and the HMM model $\lambda(A, B, \pi)$.

5.1.1 Initial Guess of the HMM Model Parameters

Every training algorithm must start with an initial guess. This section introduces the strategy to guess initial parameters for the HMM model $\lambda = (A, B, \pi)$. The training word may have triphones which may already have been trained. If that is the case, these trained models are used as the initial guess, otherwise, the model parameter A, B, and π are initialy guessed as follows:

Initial Guess of the state transition probability distribution, A:
The initial guess of the state transition probability distribution is given in Figure 5.2.

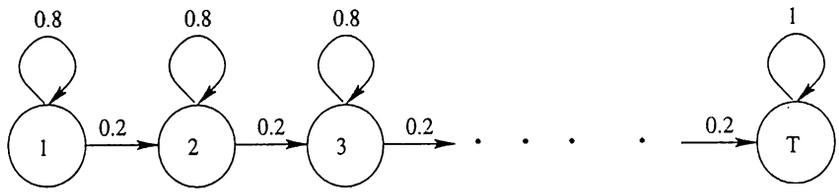

Figure 5.2. Initial estimate of the state transition probability distribution, A.

The triphones which have */sil/* at the rightmost phoneme position have only one state transition for the rightmost state. This transition goes from the rightmost state to itself because there is no other state on the right. Since the sum of the outgoing transition probabilities for a state must be 1, the probability of taking this transition is 1.

Initial Guess of the observation symbol probability distribution, B: As described in Section 3.4, in order to apply parameter estimation formulas (Equations 3.50−3.56), 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. In fact, more feature vectors are assigned to the states corresponding to the vowels or semi-vowels than those corresponding to the plosive consonants or the weak fricatives. Triphones that have vowels or semi vowels in the middle get $2 * \frac{T}{N\_TP+N\_V}$ feature vectors whereas the others get $\frac{T}{N\_TP+N\_V}$ feature vectors.

After assigning the feature vectors on triphones, we distribute feature vectors for a triphone 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 (Section 5.1.2), the Viterbi algorithm is used to modify this static distribution of the feature vectors on the states . Figure 5.3 illustrates the distribution of feature vectors on the states.

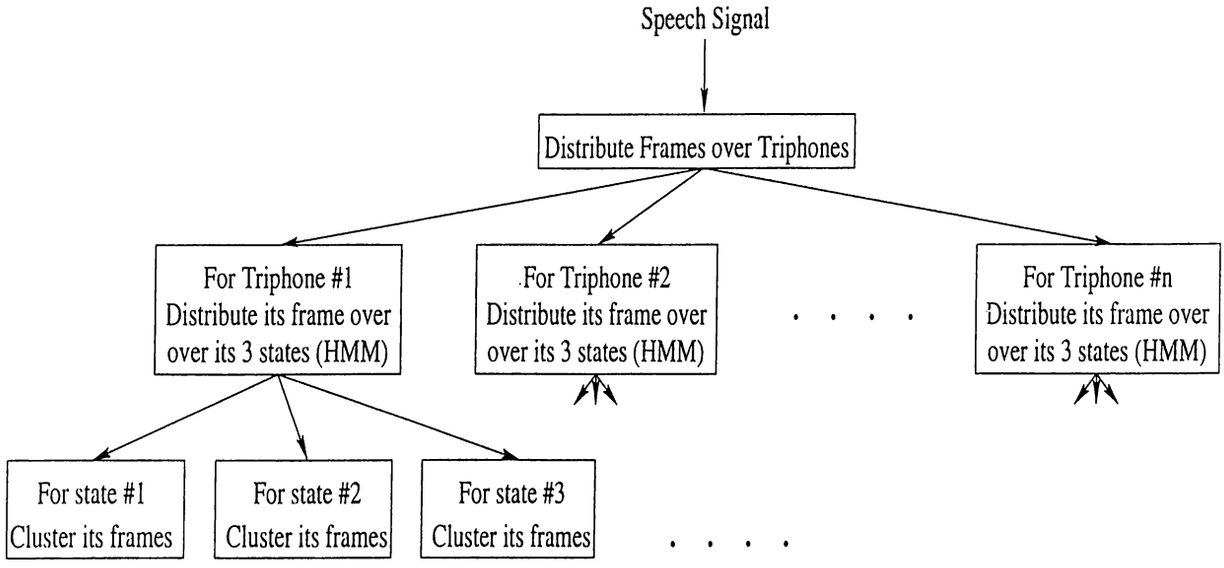

Figure 5.3. Distribution of feature vectors on to the states.

As discussed in Section 3.4, continuous observation densities with three mixture values ($M = 3$) for each state are used in this thesis. K-means clustering algorithm [15] is used to cluster the feature vectors within each state j into

a set of M clusters (using a Euclidean distance measure), where each cluster represents one of the M mixtures of the $b_j(\mathbf{o}_t)$. These mixture values are used with the given observation sequence $\mathbf{O}$ and the Equation 3.37 to compute the observation symbol probability distribution, B, for each state.

Initial state distribution, π:
The HMM model for a word has only one starting state. Therefore, we have $\pi_1 = 1$ and $\pi_i = 0$ for all i where $i = 2, 3, \ldots, N$.

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

5.1.2 Improving the HMM Model

Improvement of the model $\lambda = (A, B, \pi)$ means that the parameters of the model have to be re-estimated to get an improved model $\overline{\lambda} = (\overline{A}, \overline{B}, \pi)$.

As Figure 5.4 illustrates, there are three main steps in the improvement of an HMM.

First, we find the optimum state sequence for the given model $\lambda = (A, B, \pi)$ and given observation sequence $\mathbf{O}$ 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 $P(\mathbf{O} \mid \lambda)$ is maximized.

Second, for each state, K-means clustering algorithm are 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 3.50−3.55 are used to reestimate the parameters of the model $\lambda = (A, B, \pi)$ to get the new improved model $\overline{\lambda} = (\overline{A}, \overline{B}, \pi)$. Note that the parameter π did not change because the new model should have also one start state.

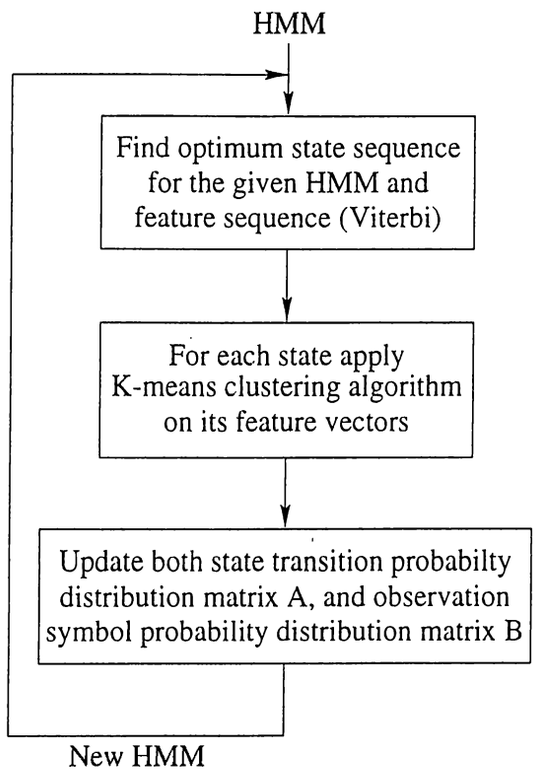

Figure 5.4. The improvement algorithm for an HMM model.

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

The 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 state 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.

5.2 The Recognition Process

The recognition algorithms for large vocabularies are complex when compared to those for small or medium size vocabularies. The complexity is caused by the search strategies employed on the vocabularies. In systems with small or even

medium size vocabularies the HMM model of each word in the vocabulary can be constructed and tested with the feature sequence fed to the system. On the other hand, this strategy is not feasible for large vocabulary speech recognition systems in terms of both system response time and recognition rate. Therefore large vocabulary speech recognition systems should have more sophisticated search strategies. This section will introduce the techniques and algorithms used in the recognition stage.

5.2.1 Codebook Information

In order to apply the Viterbi algorithm (Section 3.3) 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 $O(T \cdot N \cdot M)$ 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. From the point of system response time, the bottleneck is at the computation of the matrix B. This section introduces the *codebook* concept which is used to improve the speed of computing the matrix B.

In the training stage, after the creation of each triphone model, a space of mixture values is created by using the three mixture values of each state in every model. The space is 24-dimensional because we have 24-dimensional feature vectors (Section 2.1). Later, this space is divided into C classes by the help of the K-means clustering algorithm. In this clustering, the similarity criterion is Euclidean distance in 24-dimensional space. These clusters are usually refered to as codebook.

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. $\omega_j(\kappa_k)$ 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 in the codebook) and it is defined as

$$\omega_j(\kappa_k) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\kappa_k, \mu_{jm}, \mathbf{U}_{jm}), \qquad \begin{matrix} 1 \leq j \leq N, \\ 1 \leq k \leq C \end{matrix} \tag{5.3}$$

where c_{jm} is the mixture coefficient for the m^{th} mixture in state j. $\mathcal{N}$ represents the Gaussian pdf with mean vector μ_{jm} and covariance matrix $\mathbf{U}_{jm}$ given in Equation 3.39. The computation of $\omega_j(\kappa_k)$ is performed at the training stage and it is inserted into the model of each triphone. Suppose that, $d(\mathbf{o}_t)$ stores the cluster number in the codebook to which the observation vector $\mathbf{o}_t$ is assigned, that is $1 \leq d(\mathbf{o}_t) \leq C$. Then, the computation of the observation symbol probability distribution, B, is just a table look-up process as expressed in the equation below,

$$b_j(\mathbf{o}_t) = \omega_j(d(\mathbf{o}_t)) \qquad \begin{matrix} 1 \leq j \leq N, \\ 1 \leq t \leq T. \end{matrix} \tag{5.4}$$

Using codebook information improves the system response time. On the other hand, it may decrease the recognition rate. Therefore, it should be used with care. Section 5.2.2 illustrates the way of using codebook information in this thesis.

5.2.2 The Search Strategy

A large vocabulary, isolated word, speech recognition system must put extra effort into the designing of the search strategy for a word in the dictionary. As Section 4.3 has introduced, the dictionary model employed in this thesis is based on trie structure. There are two main approaches to search the dictionary for a word: *depth-first* and *breadth-first* search.

5.2.2.1 The Depth-First Search Strategy

The depth-first traversal strategy is given in Figure 5.5. Depth first traversal of the trie is analogous to the preorder traversal of an ordered tree. Suppose that the traversal has just visited a node v in a nodebox V and let

```
DFSearch(trie, speech)
begin
PushStack(AllNodesOf(trie))
while (StackIsNotEmpty())
    node = PopStack()
    hmm = ConstructHMM(node)
    logprob = Viterbi(hmm, speech)
    PruneSearchSpaceDF(node, logprob)
end
```

Figure 5.5. Depth-first search algorithm for a word in the dictionary.

$w_1, w_2, \ldots, w_k$ be the other nodes in the nodebox V. Then it visits w_1 next and keeps $w_2, w_3, \ldots, w_k$ waiting. After visiting w_1, we traverse the subtrie pointed by w_1 in the same manner before returning to traverse $w_2, w_3, \ldots, w_k$. At each node, the HMM model is created by cascading the HMM models of each node in the path from the level 1 of the trie to the current node. Then, the Viterbi algorithm is applied on the current HMM model with the feature sequence. The output (log-probability) of the Viterbi algorithm is used to decide at each node whether to continue on the path or to prune the rest of the trie pointed by the node and then start backtracking. If the end of a word is reached the word is inserted to the *estimation list* according to the output of the Viterbi algorithm. The estimation list contains the words that were possibly uttered. The words in the estimation list are kept in descending order according to their log-probabilities.

In Figure 5.5, the *stack* structure is used to keep truck of the nodes to be traversed. The function $AllNodesOf(trie)$ returns all the nodes in the linked-list of the nodebox *trie* from the left to the right. The function *PushStack(nodes[])* and *PopStack*() is used to push a list of nodes to the stack and pop a node from the stack, respectively. The function $ConstructHMM(node)$ returns the HMM model by cascading the HMM models of each node in the path from the level 1 of the trie to the node *node*. The function $Viterbi(hmm,speech)$ is the implementation of the Viterbi algorithm as described in Section 3.3 and it returns a log-probability value *logprob*.

The function $PruneSearchSpaceDF(node, logprob)$ is used to prune the

search space at run time. Suppose, for the sake of simplicity, this function is also responsible for adjusting the stack according to the decision made. It decides to prune the subtrie pointed by the node *node* according to the log-probability *logprob*. If this function does not perform any pruning then the HMM model of each word in the dictionary is created and tested with the feature sequence. This strategy was initialy used to test the system. The experiments (Table 5.1) show that the system has the problems caused by the morphological structure of the Turkish as explained in Section 4.1.

The pruning strategy used in the function *PruneSearchSpaceDF(node, log-prob)* is crucial in the algorithm above. The naive way of pruning the search space is to use static threshold values. These threshold values should be in terms of log-probabilities because the outputs of the Viterbi algorithm are log-probabilities. Moreover, it is a good idea to use different threshold values for each level of the trie. The threshold values for the top levels of the trie should be relatively flexible when compared to those deeper in the trie because an error in the pruning at the top levels can never be recovered and any potential computational savings are lost.

Experiments show that using static threshold values which may change according to the level of the trie does not give good results. Application of the Viterbi algorithm on different utterances of the same word may result in very different log-probabilities. When the Viterbi algorithm is applied to different HMM models with the same feature sequence, the relative position of the log-probabilities obtained from each HMM model among the others are more important than their real values. Therefore, it is reasonable to choose the promising paths at a level by examining the log-probabilities obtained from the nodes at this level. Applying this strategy with the depth-first search algorithm is difficult and costly. On the other hand, it can be easily integrated with the breadth-first search strategy.

5.2.2.2 The Breadth-First Search Strategy

Breadth-first traversal of a trie is analogous to level-by-level traversal of an ordered tree. If the traversal has just visited a nodebox V, then it next visits

```
BFSearch(trie,speech, T)
begin
level = 1
Enqueue(AllNodesOf(trie))
while (QueueIsNotEmpty()
    nodes[] = DequeLthLevelNodes(level)
    for each node node in nodes[]
        hmm = ConstructHMM(node)
        logprob[node] = Viterbi(hmm, speech)
    PruneSearchSpaceBF(logprob[], T)
    level = level + 1
end
```

Figure 5.6. Breadth-first search algorithm for a word in the dictionary.

all the nodeboxes at the same level with V. All the nodeboxes pointed by the nodes at the current level are inserted to a waiting list to be traversed after all the nodeboxes at the same level with V have been visited. Then, as in the depth-first search strategy in Figure 5.5, the HMM model is constructed at each node. Next, the Viterbi algorithm is applied on the current HMM model with the feature sequence. At this step, the log-probabilities of each node at the same level are collected. After finishing all the nodes at the current level the most promising nodes according to these log-probabilities are selected to continue to search and the rest is pruned. If the end of a word is reached, the word is inserted to the estimation list according to the output of the Viterbi algorithm. Figure 5.6 gives the algorithm of breadth-first search strategy.

In Figure 5.6, *queue* structure is used instead of stack structure as in Figure 5.5. The function *Enqueue*(*nodes*[]) is used to insert a list of nodes to the end of the queue whereas the function *DequeueLthLevelNodes*(*level*) returns the nodes which are at level *level* in the queue and removes them from the queue.

As can be seen from the algorithm above, the pruning of the search space is performed at each level of the trie. After collecting all the log-probabilities from each node at the same level, the function *PruneSearchSpaceBF(logprob[])* chooses the most promising ones. It sorts these probabilities and selects the

top $\mathcal{T}$ nodes (paths) to continue to search and the others are pruned. Suppose, for the sake of simplicity, this function also adjusts the queue according to the decision given. Different $\mathcal{T}$ values can be employed for each level of the trie, that is, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_d$ can be used at the level $1, 2, \ldots, d$ of the trie, respectively, where d is the depth of the trie. As discussed above, the pruning at the top levels should be flexible. In order to gain computational savings and yet have a higher recognition rate, we may have

$$\mathcal{T}_1 \geq \mathcal{T}_2 \geq \cdots \geq \mathcal{T}_d > 0.$$

That is, when the search goes deeper and deeper the number of paths to be followed at each level are decreased or at least not changed. For the sake of simplicity, in Figure 5.6 and in the rest of this thesis, the parameter $\mathcal{T}$ is used to select the top $\mathcal{T}$ most likelihood choices at each level in the trie, that is, $\mathcal{T} = \mathcal{T}_1 = \mathcal{T}_2 = \cdots = \mathcal{T}_d$.

Another advantage of this pruning strategy is that the system response time for one word can be predicted because the maximum number of paths to be followed is known at each level.

There is an assumption hidden in this search strategy. Considering the recognition of a word, at each level of the trie the Viterbi algorithm is applied to the partial HMM models. The HMM model which represents the first part of the given word should give a better result than those of other HMM models. At least, its result should be in the top portion of the list when the results of other HMM models are sorted. For instance, suppose that the speech signal for the Turkish word "bir" is given and our trie has a number of nodes at level 1 in which one of them contains the triphone /*sil*-b+i/. Then, when we apply the feature vectors corresponding to the speech signal to the HMM models of the nodes at level 1, the score of the HMM model for the triphone /*sil*-b+i/ should be in the top $\mathcal{T}$ when the results are sorted. Otherwise, the search strategy in Figure 5.6 prunes the search space which contains the correct word. This problem is valid at each level of the trie. The naive way of solving this problem is to follow more paths at each level. Unfortunately, this solution has a negative effect on the system response time because the number of words to be tested is increased.

The experiments show that applying the feature sequence for a word to partial HMM models does not give the desired results as described above (Table 5.2). Consequently, instead of using the entire feature sequence for a word with partial HMM models, using partial feature sequence with partial HMM models would be a better idea. That is, the appropriate region of the speech signal should be used at each level.

5.2.2.3 The Breadth-First Search Strategy Using Appropriate Region of the Speech at Each Level

The levels of the trie have special importance to the system because the HMM models created at the same level have the same number of triphones. Therefore, for instance at level l, it is a good idea to use the region of the speech signal which contains the pronunciation of l triphones with the partial HMM models at this level. The search strategy introduced in this section uses the appropriate region of the speech signal at each level of the trie. The difficulty with this approach is determining the appropriate region of the speech signal.

Dividing the speech signal into segments according to the phoneme boundaries is one of the main research areas in speech processing and it is called *speech segmentation*. In fact, one can not only predict the length of the segments but also predict the possible phonemes in the segments by using speech segmentation techniques in clean speech. Most of these techniques are computationaly costly [13].

In order to predict the appropriate region of a speech signal for each level in the trie a heuristic is adopted in this thesis. In this heuristic, the speech signal is divided into equal length regions. Each region has s frames and s is basically the estimated number of frames for a triphone in Turkish. Then, for instance at level l, the consecutive l regions of the speech from the beginning are used.

In the calculation of s, a signal database containing the utterances of 3000 Turkish words is used. For each utterance of a word, first, the average number of frames for a triphone in the word is computed by taking the ratio of the

```
BFSearchPartialSpeech(trie,speech, T)
begin
level = 1
Enqueue(AllNodesOf(trie))
while (QueueIsNotEmpty()
    nodes[] = DequeLthLevelNodes(level)
    for each node node in nodes[]
        hmm = ConstructHMM(node)
        partialspeech = AppropriateRegion(speech, level)
        logprob[node] = Viterbi(hmm, partialspeech)
    PruneSearchSpaceBF(logprob[], T)
    level = level + 1
end
```

Figure 5.7. Breadth-first search algorithm which uses an appropriate region of the speech at each level of the trie.

number of triphones in the word to the number of frames in the speech signal. Then, s is calculated as the average of these averages.

This new search strategy is illustrated in Figure 5.7 which is the same as to Figure 5.6 except that the function $AppropriateRegion(speech, level)$ is used to choose the appropriate region of the speech signal at each level. In choosing the appropriate region of the speech, the following strategies are used in special cases:

- If the end of a word is reached at some point in the search, the entire feature sequence is used with the HMM model of the word.
- If, for instance, at level l, there are less number of regions in the speech signal then l, the entire feature sequence is used with the partial/complete HMM models.
- If the number of frames in the speech signal is not evenly divisible by s, the last region of the speech signal contains also the remaining part.

```
BFSearchPartialSpeechWithCBook(trie,speech, T, cbook, L)
begin
Enqueue(AllNodesOf(trie))
level = 1
while (QueueIsNotEmpty())
    nodes[] = DequeLthLevelNodes(level)
    for each node node in nodes[]
        hmm = ConstructHMM()
        partialspeech = AppropriateRegion(speech, level)
        If (level > L)
            logprob[node] = Viterbi(hmm, partialspeech, cbook)
        else
            logprob[node] = Viterbi(hmm, partialspeech)
    PruneSearchSpaceBF(logprob[], T)
    level = level + 1
end
```

Figure 5.8. The search algorithm using codebook information after a user defined level $\mathcal{L}$.

5.2.2.4 Using Codebook Information

Sections 5.2.2.1–5.2.2.3 concentrate on the pruning of the search space for a word. In this section, the codebook information [14] is used to improve the search speed in the search space left after pruning. As described in Section 5.2.1, using codebook information without care has negative effects on the recognition rate. Therefore, in this thesis, the codebook information is used after a user-defined level in the trie. It is a good idea not to use codebook information at the top levels of the trie because an error at that level can not be recovered. The search strategy which uses codebook information is given in Figure 5.8. As a special case, if the end of a word is reached the score of the HMM model for the word is computed without using the codebook information in order to improve the recognition rate.

In Figure 5.8, *cbook* and $\mathcal{L}$ are the codebook information and the level after which it is used, respectively. The function $Viterbi(hmm, speech, cbook)$ uses the codebook information in the computation of the observation symbol probability distribution, B (Section 5.2.1).

The pronunciation of words having different stems but same suffixes may be very similar to each other as discussed in Section 4.1 in case 2. The search strategy adopted in this thesis reduces the negative effects of this problem resulting in the improvement of the recognition rate.

5.2.3 Experimental Results

The test of the system is performed with a dictionary containing about 10000 words. In the training stage, 500 words given in Appendix 6 are trained by the author. This results in 1164 triphone models. The dictionary is constructed from these 1164 triphones. The test is performed with 500 words given in Appendix 6.

The *Total Time* and *Average Time* columns in Tables 5.1–5.4 are the total time in seconds spent for recognizing the 500 test words and average time in seconds spent per word, respectively, and the *Recognition Rate* column shows the recognition rate of the system.

Table 5.1. The result of constructing HMM models of each word and then testing it with the feature sequence fed to the system.

Recognition Rate	Total Time (sec)	Average Time (sec)
78.1%	48295.12	96.59

As discussed in the begining of this chapter, the naive solution to the search problem for a word in a dictionary is to construct HMM models of every word and then test them with the given feature sequence. The result of this search strategy is given in Table 5.1. As can be seen from this table, the recognition rate of the system is relatively low and the system response time is quite large.

Table 5.2 shows the results obtained by applying the search strategy introduced in Section 5.2.2.2 with different $\mathcal{T}$ values. We include this table to show the improvements obtained by using appropriate region of the speech signal at each level in Table 5.3. As can be seen from the Table 5.2, for each value of $\mathcal{T}$ the system response time is better than the system response time in Table 5.1

as the search space is pruned at each level in the trie. Increases in the value of $\mathcal{T}$ results in increases in system response time. The increase in the recognition rate with the increase in $\mathcal{T}$ shows that applying the entire feature sequence of a word to partial HMM models does not give the required result as discussed in Section 5.2.2.

Applying the search strategy introduced in Section 5.2.2.3 gives the results in Table 5.3. In this table, when $\mathcal{T}$ increases the system response time increases because the number of words to be tested with the feature sequence increases. The increase in $\mathcal{T}$ improves the recognition rate until a limiting point. In Table 5.3, this limiting point is reached for $\mathcal{T} = 20$. For small values of $\mathcal{T}$ (5,10, and 15), the recognition rate is lower when compared to the recognition rate for $\mathcal{T} = 20$ because for these values the pruning is not flexible for these values and the error made at the top level of the trie can not be recovered in the rest of the trie. After $\mathcal{T}$ exceeds 20, the recognition rate decreases as the number of words searched increases so does the number of confusable words.

When the system response time and the recognition rates in Table 5.3 compared with the ones in Table 5.2, the positive effects of using appropriate region of the speech at each level can be seen clearly. For instance, for the value $\mathcal{T} = 20$, we spend 3053.08 seconds to obtain the recognition rate of 92.2% in Table 5.3 where as in Table 5.2, we spend 3669.32 seconds for the recognition rate of 20.6%.

The effects of using codebook information as discussed in Section 5.2.2.4 can be examined in Table 5.4. In this figure, $\mathcal{L}$ = -1 represents the case in which codebook information is not used.

The Table 5.4 shows clearly that the codebook information must be used with care. Using codebook information especially at the top level of trie results in non-recoverable errors.

The speech recognition system developed in this thesis has been also trained and tested by a male and a female. During the tests performed over the system, the search strategy introduced in Section 5.2.2.4 is used with the parameters $\mathcal{T}$= 20 and $\mathcal{L}$= -1. As a result, recognition rates of 93.0% for the male and

Table 5.2. The results obtained by the execution the function *BFSearch(trie, speech, $\mathcal{T}$)* for $\mathcal{T}$ = 5, 10, 15, 20, 25, and 30.

$\mathcal{T}$	Recognition Rate	Total Time (sec)	Average Time (sec)
5	9.0%	1685.00	3.37
10	15.0%	2323.73	4.65
15	18.2%	2984.77	5.97
20	20.6%	3669.32	7.34
25	23.2%	4369.12	8.74
30	25.0%	5100.43	10.20

Table 5.3. The results obtained by the execution of the function *BFSearchPartialSpeech(trie, speech, $\mathcal{T}$)* for $\mathcal{T}$ = 5 , 10, 15, 20, 25, and 30.

$\mathcal{T}$	Recognition Rate	Total Time (sec)	Average Time (sec)
5	73.2%	1000.76	2.0
10	87.2%	1689.39	3.38
15	91.0%	2384.84	4.77
20	92.2%	3053.08	6.11
25	91.2%	3750.94	7.50
30	90.8%	4451.70	8.90

Table 5.4. The results obtained by the execution of the function *BFSearchPartialSpeechWithCBook(trie, speech, $\mathcal{T}$, cbook, $\mathcal{L}$)* for $\mathcal{T}$ = 20 and $\mathcal{L}$ = -1, 2, 3, 4, 5, 6, 7, and 8 (l = -1 corresponds to the use in which codebook information is not used).

$\mathcal{L}$	Recognition Rate	Total Time (sec)	Average Time (sec)
2	28.4%	1539.52	3.08
3	45.4%	1616.18	3.23
4	71.4%	1774.69	3.55
5	79.8%	1918.85	3.84
6	88.0%	2105.44	4.21
7	90.6%	2287.08	4.57
8	91.0%	2461.27	4.92
-1	92.2%	3053.08	6.11

90.0% for the female were obtained. These are comparable and acceptable recognition rates.

5.2.4 Conclusions

Experiments show that both in the training stage and the recognition stage the words should be uttered as clearly as possible. Especially, the utterances of the suffixes are very important; they must be done with care.

The codebook information must be used with care and it is not reasonable to use codebook information at the top levels of the trie as can be understood from the Table 5.4. When Tables 5.3 and 5.4 compared, it seems to be more reasonable to have the parameter $\mathcal{L}= -1$ and play with the parameter $\mathcal{T}$ in the final version of the search strategy (Figure 5.8) to fine tune the tradeoff between the system response time and recognition rate.

With the naive search algorithm which constructs the HMM models of each word in the dictionary and then tests them according to the feature sequence fed to the system, 48295.12 seconds are spent to obtain the recognition rate of 78.1%. With the new search strategy introduced in this thesis, 3053.08 seconds are spent for the recognition rate of 92.2% (with $\mathcal{L}= -1$ and $\mathcal{T}= 20$ in the algorithm in Figure 5.8), and the reasons for the errors made in the recognition stage are given below:

- 10% of the errors are caused by the wrong pruning of the search space,
- 70% of the errors are made because of the words whose pronunciations are very similar like "günüz" - "günümüz", "çalışmayan" - "çalışamayan", and "olanı" - "olanın" (to overcome this problem more training data should be used especially for confusing words), and
- it seems that 20% of the errors are caused by the initial guess strategy for the HMM models as described in Section 5.1.1. Experiments show that with this strategy some words can have a bigger influence on the shared triphones. The training order of the words may change the recognition rate.

Therefore, the new search strategy improves significantly both the system response time and recognition rate.

Chapter 6

Conclusions

In this thesis, a speaker dependent, large vocabulary (about 10000 words), isolated word speech recognition system is developed for Turkish language.

The TEOCEP feature parameters are used to characterize the speech signal. These parameters are based on multirate sub-band analysis of the speech signal. The average Teager energy value is computed for each sub-signal. TEOCEP feature parameters are obtained by applying logarithmic compression and cosine transformation to these energy values.

Triphones are used as the smallest unit for recognition. Each word in the dictionary is modeled with triphones which are modeled by a three-state HMM model. The output distributions in the HMM models are represented by Gaussian Mixture Densities with three mixture values. The topology of HMM models for the triphones are fixed. On the other hand, the HMM models for the words may have different number of states according to the number of triphones in it. In the training stage, the word model is trained as a whole and then each HMM model of the triphones are stored individually. In the recognition stage, trained HMM models for triphones are used to construct the HMM models of the words in the dictionary. In this way the words that are not trained can be recognized.

Turkish is an agglutinative language and it uses purely concatenative morphological combination with only suffixes attaching to a free morpheme. That

is, even for one stem, the dictionary must contain large number of entries with the same stem but different suffixes. A new dictionary model for Turkish language is introduced. This model is based on a trie structure and each node of the trie contains a triphone. This dictionary model provides space efficiency for large vocabularies. Moreover, this modeling of the dictionary allows for fast and accurate search algorithms.

A new search strategy for Turkish is introduced on the dictionary for a given word. This search strategy performs breadth-first traversal of the trie. At each level l, the HMM models of the nodes in the level l are constructed by cascading the HMM models of the triphones stored in the nodes in the path from the level 1 to the current node. Viterbi algorithm is applied to these partial HMM models with the appropriate region of the speech signal for the level l. The pruning algorithm then chooses the most promising nodes to continue searching the others are pruned. In this thesis, the proposed search strategy was described in detail and the experiments show that the recognition rate of the system is more than 90% with an average response time 6.11 seconds for a word.

The strategies and algorithms adopted in this thesis can be extended to be used with Turkish language models which may use grammar rules to improve both the system response time and recognition rate.

Bibliography

[1] Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Appllications in Speech Recognition", *Proceedings of the IEEE*, vol. 77, no. 2, February 1989.

[2] Kai-Fu Lee, Hsiao-Wuen Hon, and Raj Reddy, "An Overview of the SPHNIX Speech Recognition System", *IEEE Transactions on Acoustic, Speech, and Signal Processing* , vol. 38, no. 1, January 1990.

[3] Arthur Nadas, David Nahamoo, and Michael A. Picheny, "On a Model-Robust Training Method for Speech Recognition", *IEEE Transactions on Acoustic, Speech, and Signal Processing* , vol. 36, no. 9, April 1988.

[4] William G. Bliss, and Louis L. Scharf, "Algorithms and Architectures for Dynamic Programming on Markov Chains" *IEEE Transactions on Acoustic, Speech, and Signal Processing* , vol. 37, no. 6, June 1989.

[5] Joseph W. Picone, "Signal Modeling Techniques in Speech Recognition", *Proceedings of the IEEE*, vol. 81, no. 9, September 1993.

[6] Sadaoki Furui, "A Training Procedure for Isolated Word Recognition Systems", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. ASSP-28, no. 2, April 1980.

[7] Frederick Jelinek, Lalit R. Bahl, and Robert L. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech" *IEEE Transactions on Information Theory*, vol. IT-21, no. 3, May 1975.

[8] David Pepper, T. P. Barnwell, and M. A. Clements, "Using a Ring Parallel Processor for Hidden Markov Model Training", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. 36, no. 2, February 1990.

[9] Louis A. Liporace, "Maximum Likelihood Estimation for Multivariate Observations of Markov Sources", *IEEE Transactions on Information Theory*, vol. IT-28, no. 5, September 1982.

[10] Steve Young "A Review of Large-Vocabulary Continuous-Speech Recognition", *IEEE Signal Processing Magazine*, September 1996.

[11] Engin Erzin, "New Methods for Robust Speech Recognition", *PhD thesis*, Bilkent University, 1995.

[12] Firas Jabloun, "Large Vocabulary Speech Recognition in Noisy Environments", *MS thesis*, Bilkent University, 1998.

[13] Arçın Bozkurt, "Automatic Speech Segmentation Based on Subband Decomposition", *MS thesis*, Bilkent University, 1999.

[14] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[15] Biing-Hwang Juang and L. R. Rabiner, "The Segmental K-Means Algorithm for Estimating Parameters of Hidden Markov Models", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. 38, no. 9, September 1990.

[16] Steve Young, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland, *The HTK Book*, Entropic Cambridge Research Laboratory, March 1997.

[17] John R. Deller, John G. Proakis, and John H. L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan, 1993.

[18] J. B. Allen, "How Do Humans Process and Recognize Speech", *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 567-577, October 1994.

[19] F. Itakura, "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals", *Journal of Acoustic Soc. Am.*, pp. 535a, 1975.

[20] Patrick Kenny, Matthew Lennig, and Paul Mermelstein, "A Linear Predictive HMM for Vector-Valued Observations with Applications to Speech

Recognition", *IEEE Transactions on Acoustic, Speech, and Signal Processing* , vol. 38, no. 2, February 1990.

[21] F. Jelinek, "Continuous Speech Recognition by Statistical Methods", *Proceedings of IEEE*, pp. 532-536, April 1976.

[22] H. M. Teager, "Some Observations on Oral Air Flow During Phonation", *IEEE Transactions on Speech and Audio Processing*, October 1980.

[23] A. J. Viterbi, "Error Bounds for convolutional Codes and an Asymptotically Optimal Decoding Algorithm", *IEEE Transactions on Information Theory*, vol. 13, pp. 260-269, April 1967.

[24] Robert L. Kruse, *Data Structures and Program Design*, Prentice-Hall, Englewood Cliffs, NJ 1984.

[25] Kemal Oflazer, "Error-Tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction", *Computational Linguistics*, vol. 22, no. 1, 1996.

Appendix A
The Training Word List

In the following list, the words are given in descending order according to the usage frequencies in the Turkish corpus.

ve	iş	bundan	bakan	vardı
bir	hiç	verdiği	erken	barış
bu	yine	vardır	siz	yapılması
için	şekilde	açık	halk	şeklinde
ile	ikinci	farklı	bize	yatırım
olarak	eğitim	yaklaşık	içerisinde	izin
çok	şimdi	bulunduğu	edildi	gerçekten
daha	çünkü	işte	çalışan	tür
en	dünya	olmadığını	öne	sırada
olan	nedeniyle	ele	belki	ona
sonra	eski	mustafa	elde	sürekli
kadar	alan	değerli	bunlar	bildirildi
gibi	hiçbir	halinde	su	karşısında
ise	insan	biraz	açıklamada	hareket
büyük	nasıl	aldı	ileri	tekrar
her	siyasi	cumhuriyet	kişinin	an
genel	konuştu	mücadele	herkes	irtica
ama	artık	sağlık	gündeme	dönemde
o	tam	gerektiğini	ediyorum	kötü
iki	hakkında	yıllık	olsa	sayısı

türkiye	kurulu	bildirdi	yapmak	derece
ne	yerine	olur	türlü	yoğun
olduğunu	milli	sonunda	başarılı	beşiktaş
türk	lira	çıkan	haline	hakları
var	doğru	bağlı	millet	herhangi
yeni	şey	görev	çocuk	birliği
son	ayrıca	dış	kalan	konu
başkanı	bizim	ayrı	ayında	rum
yüzde	yüksek	ettiği	anadolu	bulundu
göre	ana	mesut	gelecek	genelkurmay
dedi	milletvekili	belediye	yılmazın	bugüne
zaman	olması	çıktı	madde	istanbulda
olduğu	karar	yaptı	hangi	alıyor
yıl	yılında	yönelik	baykal	öte
önce	özellikle	bana	yılı	olumlu
bin	uzun	güzel	nedenle	yanında
içinde	kişi	sendika	demokrasi	mevcut
sayın	yani	kararı	birkaç	gece
başkan	yeniden	çeşitli	hale	olay
ilgili	vergi	dünyanın	beş	boyunca
tarafından	dolar	amacıyla	şunları	araştırma
diye	biri	kurban	geliyor	ismail
söyledi	olacak	arasındaki	iddia	belirtildi
arasında	ay	veren	hukuk	olabilir
iyi	ali	lideri	başına	zorunda
ki	güvenlik	onu	merkez	doğu
milyon	cumhurbaşkanı	hatta	ecevit	olmadığı
türkiyenin	belirterek	ortak	dile	düzenlenen
bakanı	yapan	yandan	toplantısında	inci
önemli	verdi	kanun	teşekkür	size
yer	ardından	müdürü	durumda	almak
karşı	onun	bayram	hep	haber
aynı	neden	adlı	efendim	hasan
bazı	ülke	hava	aslında	çözüm

ankara	belirten	el	bunların	başlayan
diğer	yol	ağır	kültür	kendini
oldu	süre	değildir	meydana	yanı
nisan	ilişkin	kaydetti	olmayan	göz
gün	günü	iç	ırak	lazım
özel	az	gerçek	altına	bankası
devam	küçük	belli	başında	söyleyen
yılmaz	sosyal	demokratik	savunma	kuzey
ben	dışişleri	diyor	zamanda	mutlaka
sadece	altında	temel	süren	dünyada
kendi	eğer	gerekli	isteyen	sarı
üzerine	ekonomik	alınan	gerekiyor	bizi
etti	benim	dört	üye	olup
hem	basın	ceza	sahibi	başta
eden	ediyor	üyesi	yanlış	sona
gelen	arada	ülkenin	beni	dolayı
istanbul	hemen	sonucu	resmi	tarihinde
bunun	teknik	anap	günlerde	turizm
yaptığı	konuda	il	savaş	alkışlar
bugün	kısa	aldığı	ciddi	kimse
konusunda	edilen	ünlü	böylece	kaldı
birlikte	sırasında	zaten	dolayısıyla	sıralarından
şu	yönetim	gelir	cem	sistemi
böyle	önceki	deniz	orta	oluşan
üzere	bakanlığı	oluyor	açıkladı	ön
tüm	adına	adı	soru	kemal
biz	kadın	sıra	istedi	sanayi
ifade	başladı	anayasa	ziyaret	komisyonu
tek	sayfa	devletin	film	galatasaray
veya	uygun	zor	ticaret	güçlü
ortaya	anda	yabancı	önümüzdeki	insanların
bile	destek	yana	sık	gerekir
milyar	dışında	sabah	beri	onlar
üç	etmek	gerek	süleyman	istiyorum

kamu	mümkün	bulunuyor	kendisine	işi
şöyle	geri	verilen	yılda	çarşamba
türkiyede	geldi	gereken	oldukça	alman
başka	yakın	sayılı	izmir	erkek
başbakan	yardımcısı	çiller	merkezi	futbol
fazla	bilgi	ederek	geniş	altın
geçen	genç	açısından	hükümeti	adam
üzerinde	çalışma	hizmet	önünde	türkiyeyi
bunu	birçok	birinci	insanlar	yıldır
seçim	konusu	takım	edecek	dakika
dün	yasa	öyle	henüz	yıllarda
bulunan	askeri	olsun	bölge	çal

Appendix B
The Testing Word List

genelkurmaydaki	yapılmasıyla	alkışlayanların
beşiktaştaki	yapanlara	alkışlayanlara
alanındakine	nedenlerdendir	alkışlayarak
konuşturmadan	nedenlerini	anayasadaki
ankaradakiler	nedenlerine	anayasadan
alınanlardan	nedenlerden	anayasanın
cumhurbaşkanını	ülkesindeki	anayasayı
ben	ülkesinden	anayasaya
verdiklerinden	ülkesini	başaramadığı
bugünlerinden	ülkerden	başaramadığını
ikiyle	ülkesiz	başaramasa
beşik	yolunu	başaramayan
barış	yoluna	başaramayanlar
ayını	yoldakilerin	başaramaz
ünlü	yolundaki	başardı
sorumlunun	süresiyle	başardığını
sonrasının	süresinin	başardığına
konusundaki	süresince	başardım
bankasındaki	süresini	başkanda
söyleyenlerdir	süresine	başkandan
çarşambasında	ilişkinizin	başkanlığı
dünyadaki	ilişkinize	başkanlığın
dünyaları	ilişkimde	başkanlığını

oluşana	gününde	başkanlığının
işindesiniz	günümüz	başkanlığına
almanlardan	günlü	başkanlığında
onların	günkü	başkanlığındaki
bizimdir	günde	başkanlığından
durumdakileri	küçükse	başkanlıkla
ilgilinin	benim	başkanlıkları
çalmasından	benimdir	başkanlıklarını
komisyonlarındaki	benimdi	başkanlıklarının
olanı	benimse	başkanlıklarında
nasıl	kısarak	başkanlıklarından
kısık	kısanın	başkanlıklara
açısı	kısadır	başkanlıkta
alışı	kısa	başkanlıktaki
sonrasındaki	kısarken	başkanlıktan
kişi	kısaya	başkanla
işi	yönetimdir	başkanlar
gibilikten	yönetimden	başkanları
yenisinin	öncekilerden	cumhuriyetinden
erdi	öncekilerde	başkanlarını
eskilerinden	öncekinde	cumhuriyetinize
şeyi	öncekilerin	cumhuriyetini
özellikle	sayfadaki	cumhuriyetinin
büyükse	sayfadan	cumhuriyetiyle
dışişlerinden	uygunlarından	başkanlara
konudaki	uygunlar	başkanlardır
halindeler	uygundur	başkanlardan
söylenenlere	uygunun	cumhuriyetin
genelkurmayının	uygunca	cumhuriyetinde
görevlendirdiği	destekliklerinin	cumhuriyetindeki
içindekileri	destekleyince	özenim
sonrakilerde	destekleyerek	özenin
kadardaki	destekleriyle	özenle
herkesten	gerisi	bölgelerdeki

hiçbirini	gerili	ülkem
hiçbirisi	gerisindekiler	ülkemde
hiçbirine	gerisindedir	ülkemdeki
olduğunun	yakınındaki	ülkerde
dolduğunu	yakındakine	ciddi
yatırmadan	açıklayabiliyor	ceza
yenisine	açıklamasındaki	cezam
yatırılmasıyla	açıklamalarında	yatı
yatırılmasını	ettiklerinden	dönemlik
görevlerde	etmesindendir	elde
zamanlama	ekonomisinden	kalın
zamanına	ekonomilerini	ticaret
yılmayın	ekonomilerinin	ticareti
yılmadan	ekonomisinin	zoru
öncekini	ekonomisinde	çıkarım
öncekine	çalması	çıkarın
önceki	basınında	bölgeler
içini	basındaki	bölgelerde
içine	dakikada	bölgelerden
söylediklerinizi	dakikası	bölgelerdir
iyiniyeti	dakikasını	bölgelere
iyiniyetinizi	dakikasına	bölgeleri
iyiniyetine	yıldıramazlar	bölgelerin
dönemli	adambaşı	bölgelerinde
ankaradaki	adamaya	bölgelerindeki
doldurulduğunu	adanaya	bölgelerinden
nisandaki	altını	bölgelerine
gününüz	altınını	bölgelerini
günleri	futbolu	bölgelerinin
günlere	erkekliklerinden	bölgelerinindir
özelden	çarşambaya	bölgelerinize
üzerinedir	gerektiği	bölgeleriyle
söylettiği	insanlığını	bölgeli
birlikten	galatasarayın	bölgeni

milyarı
milyara
şöyledir
şöylece
geçenlerden
geçenlerde
geçenlerin
bunuda
seçimdeki
yapılmasıyla
yapanlara
nedenlerdendir
nedenlerini
nedenlerine
nedenlerden
ülkesindeki
ülkesinden
ülkesini
ülkerden
ülkesiz
yolunu
yoluna
yoldakilerin
yolundaki
süresiyle
süresinin
süresince
süresini
süresine
ilişkinizin
ilişkinize
ilişkimde
gününde
günümüz

galatasaraya
sanayinde
oluşturma
sistemdeki
sıralarken
kaldırılmasındaki
dakikasından
dakikasında
dakikaya
alkışlayanların
alkışlayanlara
alkışlayarak
anayasadaki
anayasadan
anayasanın
anayasayı
anayasaya
başaramadığı
başaramadığını
başaramasa
başaramayan
başaramayanlar
başaramaz
başardı
başardığını
başardığına
başardım
başkanda
başkandan
başkanlığı
başkanlığın
başkanlığını
başkanlığının
başkanlığına

bölgenin
ciddiler
ciddilerinden
ciddiliğine
ciddiye
ciddiyet
ciddiyeti
ciddiyetinde
ciddiyetine
ciddiyetini
ciddiyetinin
ciddiyetiyle
ticaretin
ticaretinde
ticaretindeki
ticaretinden
ticaretine
ticaretini
başaramasa
ticaretiyle
ticaretti
zoruna
zorunda
zorundu
zorunu
çıkarını
çıkarının
çıkarına
çıkarınadır
çıkarında
çıkarından
çalışması
çalışmaz
çalışmayan

günlü
günkü
günde
küçükse
benim
benimdir
benimdi
benimse
kısarak
kısanın
kısadır
kısa
kısarken
kısaya
yönetimdir
yönetimden
öncekilerden
öncekilerde
öncekinde
öncekilerin
sayfadaki
sayfadan
uygunlarından
uygunlar
uygundur
uygunun
uygunca
destekliklerinin
destekleyince
destekleyerek
destekleriyle
gerisi
gerili
gerisindekiler

başkanlığında
başkanlığındaki
başkanlığından
başkanlıkla
başkanlıkları
başkanlıklarını
başkanlıklarının
başkanlıklarında
başkanlıklarından
başkanlıklara
başkanlıkta
başkanlıktaki
başkanlıktan
başkanla
başkanlar
başkanları
cumhuriyetinden
başkanlarını
cumhuriyetinize
cumhuriyetini
cumhuriyetinin
cumhuriyetiyle
başkanlara
başkanlardır
başkanlardan
cumhuriyetin
cumhuriyetinde
cumhuriyetindeki
özenim
özenin
özenle
bölgelerdeki
ülkem
ülkemde

çalıyor
çalıyorum
çalınan
çalabilir
çalacak
çalacaklar
çalamadım
çalmaz
çalanı
çalmayan
çalana
çalanlar
çalanları
çalanların
çalanlara
çalanlarda
çalar
çaları
çalarım
çalmaya
çalardı
çalmayı
çaldı
çaldığı
çaldığını
çaldığına
çaldığında
çalmasa
çaldıramaz
çaldırmadı
çaldırmayan
çalma
çalmadı
çalmadaki

gerisindedir
yakınındaki
yakındakine
açıklayabiliyor
açıklamasındaki
açıklamalarında
ettiklerinden
başaramasa
başkanlığına
ülkemdeki
ülkerde
ciddi
ceza
cezam
yatı
dönemlik
etmesindendir
ticaretinin
çalmadan
çalmak
çalmakla
çalmalık
çalmalarını
çalmalarından
çalmanın
elde