A STUDY ON LANGUAGE MODELING FOR TURKISH LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ ORKAN BAYER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

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

Assoc. Prof. Dr. Tolga Çiloğlu Co-Supervisor	Dr. Meltem Turhan Yöndem Supervisor

Examining Committee Members

Assoc. Prof. Dr. Göktürk Üçoluk	(METU, CENG)	
Dr. Meltem Turhan Yöndem	(METU, CENG)	
Assoc. Prof. Dr. Tolga Çiloğlu	(METU, EE)	
Dr. Onur Tolga Şehitoğlu	(METU, CENG)	
Assist. Prof. Dr. Bilge Say	(METU, COGS)	

ABSTRACT

A STUDY ON LANGUAGE MODELING FOR TURKISH LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Bayer, Ali Orkan

M.S., Department of Computer Engineering

Supervisor : Dr. Meltem Turhan Yöndem

Co-Supervisor : Assoc. Prof. Dr. Tolga Çiloğlu

September 2005, 54 pages

This study focuses on large vocabulary Turkish continuous speech recognition. Continuous speech recognition for Turkish cannot be performed accurately because of the agglutinative nature of the language. The agglutinative nature decreases the performance of the classical language models that are used in the area. In this thesis firstly, acoustic models using different parameters are constructed and tested. Then, three types of n-gram language models are built. These involve class-based models, stem-based models, and stem-end-based models. Two pass recognition is performed using the Hidden Markov Model Toolkit (HTK) for testing the system first with the bigram models and then with the trigram models. At the end of the study, it is found that trigram models over stems and endings give better results, since their coverage of the vocabulary is better.

Keywords: Large Vocabulary Continuous Speech Recognition, Agglutinative Languages, Language Modeling, Two-pass Recognition

ÖZ

TÜRKÇE GENİŞ DAĞARCIKLI SÜREKLİ KONUŞMA TANIMA İCİN DİL MODELLEME ÜZERİNE BİR ÇALIŞMA

Bayer, Ali Orkan

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Dr. Meltem Turhan Yöndem

Ortak Tez Yöneticisi : Doç. Dr. Tolga Çiloğlu

Eylül 2005, 54 sayfa

Bu çalışma Türkçe için geniş dağarcıklı sürekli konuşma tanıma problemine yönelmektedir. Türkçe için sürekli konuşma tanıma, dilin eklemeli yapısı nedeniyle iyi yapılamamaktadır. Dilin eklemeli doğası bu alanda kullanılan klasik dil modellerinin performansını düşürmektedir. Bu tez çalışmasında öncelikle değişik parametreler kullanılarak akustik modeller oluşturulmuştur ve test edilmiştir. Sonra üç tip n-gram dil modeli eğitilmiştir. Bu modeller sınıf tabanlı modelleri, gövde tabanlı modelleri ve gövde-ek-dizisi tabanlı modelleri kapsamaktadır. Hidden Markov Model Toolkit (HTK) kullanılarak sistemi önce bi-gram ve sonra da tri-gram modellerle test etmek için iki aşamalı tanıma yapılmıştır. Bu çalışmanın sonunda kelime dağarcığını kapsamaları nedeniyle, gövde ve ek dizileri üzerinden eğitilen modellerin daha iyi sonuç verdiği bulunmuştur.

Anahtar Kelimeler: Geniş Dağarcıklı Konuşma Tanıma, Eklemeli Diller, Dil Modelleri, İki Aşamalı Tanıma

To my grandmother, my parents, and my brother

ACKNOWLEDGMENTS

I would like to thank to my supervisor Dr. Meltem Turhan Yöndem for her support and not letting anything to bother me, for her guidance, and advices.

I would also thank to my co-supervisor Assoc. Prof. Tolga Çiloğlu for his patience, support, and guidance, and also for his encouragements with his calm and positive attitude.

I am grateful to Dinç Acar for the introductory sessions on HTK; to Onur Soysal for his help on the computer accounts.

I am very grateful to my grandmother and my parents, this thesis would not be possible without all the effort they have made for me and all the things they have given to me; and to my brother for his intercontinental support.

Finally, I would like to thank Müge Sevinç for her great support, suggestions and for anything she has done especially in the final months of the period.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Speech Recognition

Speech recognition is the problem of finding the correct transcription for a given speech signal. The research on speech recognition dates back to 1950s and started in 1952, when Davis, Biddulph, and Balashek built an isolated digit recognizer for a single speaker at Bell Laboratories [1]. There have been many studies on the subject and many approaches emerged. Currently, the approaches in the area can be divided into three main classes [1]:

1. The acoustic-phonetic approach

2. The pattern recognition approach

3. The artificial intelligence approach

The first approach is the acoustic-phonetic approach, which is the oldest approach. This methodology extracts features from the speech signal, analyzes the properties of the features and then tries to segment and label the speech signal. However, this methodology has so many limitations that it is not used today. The second approach is the pattern recognition approach; it is the one that is used in this study. The details of this approach will be given in the next section. The third approach is the artificial intelligence approach. Artificial neural networks are used to model the human auditory system. It is the youngest approach but it is not well developed yet.

1.2 Statistical Speech Recognition

In the current state of the art, statistical pattern recognition approach is the most widely used one. In terms of this approach we can think of the speech recognition problem as finding the most likely word string given some acoustic evidence. The acoustic evidence can be seen as a sequence of feature vectors, $\boldsymbol{Y} = \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T$, that are extracted from the speech signal using a feature extraction mechanism. In addition to that, the transcription involves a sequence of words, $\boldsymbol{W} = w_1, w_2, ..., w_n$. Considering that $\hat{\boldsymbol{W}}$ denotes the most likely word string, the problem can be formulated as in Equation 1.1 [2] [3].

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{Y}) \tag{1.1}$$

Equation 1.1 can be rewritten as in Equation 1.2 using Bayes' formula of probability theory.

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} \frac{P(\boldsymbol{W})P(\boldsymbol{Y}|\boldsymbol{W})}{P(\boldsymbol{Y})} \tag{1.2}$$

The procedure for finding $\hat{\boldsymbol{W}}$ is performed on a fixed acoustic evidence. Thus, $P(\boldsymbol{Y})$ is fixed in the maximization of Equation 1.2, and we can ignore $P(\boldsymbol{Y})$. Hence, the problem can be reformulated as:

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})P(\boldsymbol{Y}|\boldsymbol{W}) \tag{1.3}$$

From Equation 1.3, it can be seen that statistical models for $P(\boldsymbol{Y}|\boldsymbol{W})$ and $P(\boldsymbol{W})$ must be constructed. The probability distribution of an acoustic evidence, $\boldsymbol{Y}$, given a sequence of words, $\boldsymbol{W}$, can be modeled by using acoustic modeling. The probability of a word sequence can be estimated by using language modeling. Thus, by using training data these models are constructed and this phase is known as the training phase. The outline of this phase is given in Figure 1.1.

After the construction of the models, an acoustic evidence is presented to the system for a transcription. Hence, the system must search for a sequence of words that best matches this acoustic evidence using the above statistical models. This process is known as decoding. The outline for the decoding phase is given in Figure 1.2

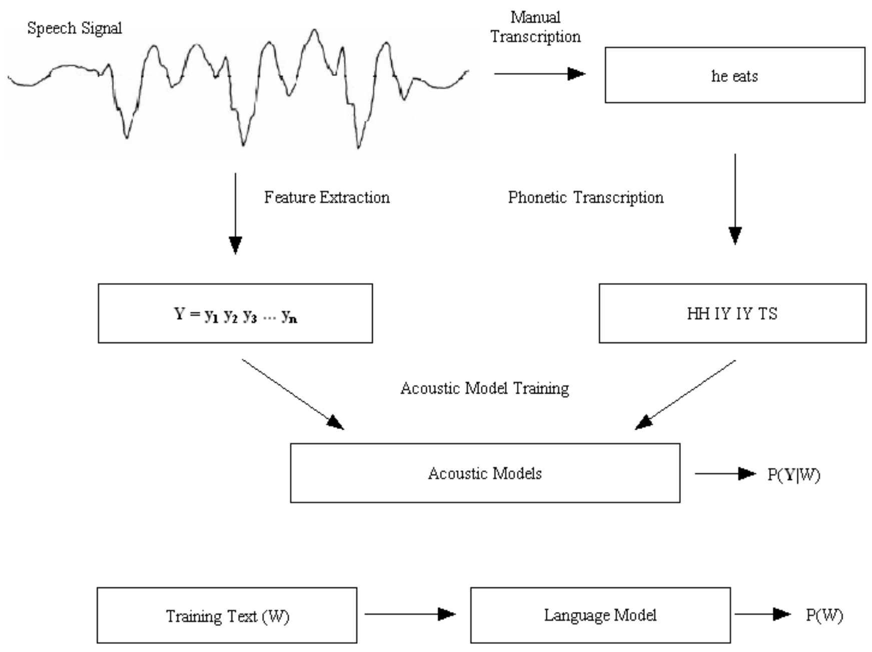

Figure 1.1: The outline of the training phase.

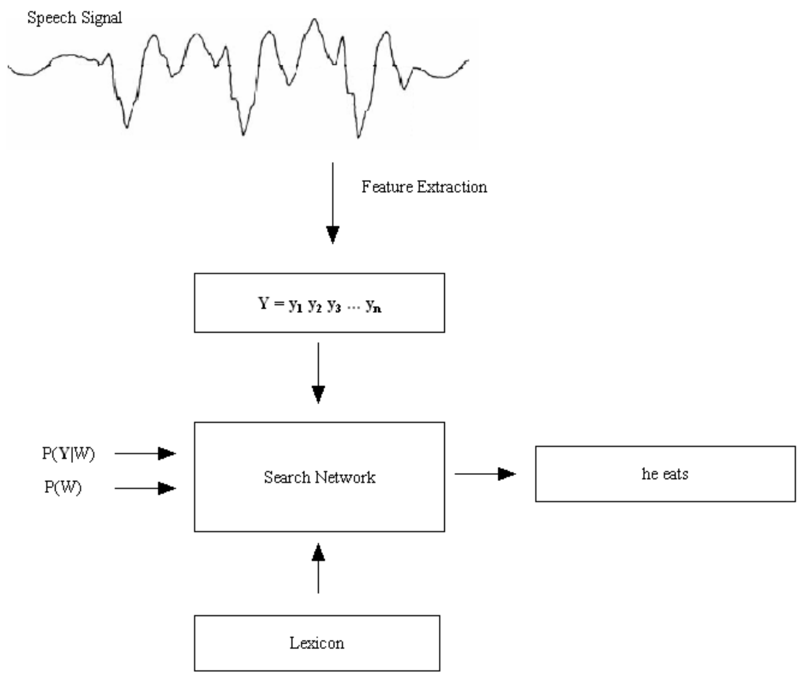

Figure 1.2: The outline of the decoding phase.

1.3 Turkish Speech Recognition

There are a few studies on Turkish speech recognition. Most of the studies were being done on isolated word recognition, where only a word is recognized at each search step [4] [5] [6].

Recently, large vocabulary continuous speech recognition systems are started to be studied [7] [8] [9] [10]. However, the performance of these systems are poor and not in the level of systems in other languages like English.

The acoustic modeling is almost universal for all languages, since the methods that are used yield similar results. However, the classical language modeling techniques cannot be applied successfully to Turkish.

Turkish is an agglutinative language. Thus, great number of different words can be built from a base form using derivations and inflections [11]. The productive derivational and inflectional structure brings the problem of high growth rate of vocabulary. Furthermore, Turkish is a free word order language. Thus, the language models that are constructed using the co-occurrences of words do not perform well.

1.4 Outline of the Thesis

The details of the statistical speech recognition problem and large vocabulary continuous speech recognition for Turkish will be presented in this thesis. The thesis mainly focuses on using different units in language modeling. In Chapter 2 the issues governing statistical acoustic modeling will be given. The problem of language modeling will be discussed in Chapter 3. Several issues about the decoding process will be mentioned in Chapter 4. Chapter 5 presents the work that is conducted in this thesis. Finally, the conclusion and the future work will be explained in Chapter 6.

CHAPTER 2

ACOUSTIC MODELING

The acoustic modeling is used for the estimation of the probability of the acoustic evidence given the word sequence. The speech waveform is not directly used as an acoustic evidence, rather it is passed through a process that extracts the useful information about the speech signal. This process is known as the front-end parametrization and it is used to obtain the feature vectors that are used as the acoustic evidence from the speech waveform [2] [3]. In the next section, the details of this process will be given.

2.1 Front-End Parametrization

The purpose of the front-end parametrization is to extract feature vectors from a speech signal. These feature vectors should be independent of the speaker and the environment but the characteristics of the sounds. For this purpose, the speech production methods are used for modeling the human vocal tract, which is seen as a filter. These models estimates the parameters of this filter. The speech signal is processed within short-time intervals, in which the speech signal can be considered as stationary.

2.1.1 Short-Time Analysis

The processing of speech in short-time intervals is known as *short-time* analysis. Basically, the signal is decomposed into segments and these segments are then processed independently. The short-time signal of a speech signal, $x[n]$ at m is defined as in Equation 2.1 [12].

$$x_m[n] = x[n]w[m-n] \tag{2.1}$$

where $w_m[n]$ is a window function. The Hamming window is typically used in short-time analysis of speech signal. The Hamming window of size N is given as

$$h[n] = \begin{cases} 0.54 - 0.46cos(2\pi \frac{n}{N}) & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

The discrete short-time Fourier representation at m is defined as [13]:

$$X(m,k) = \sum_{n=-\infty}^{\infty} w[m-n]n[n]e^{-j\frac{2\pi}{N}kn}$$

2.1.2 Mel-Frequency Cepstrum Coefficients

The discrete short-time Fourier transform is used for the extraction of Mel-Frequency Cepstrum Coefficients (MFCC). After taking the discrete short-time Fourier transform of the signal, the frequency is wrapped to a nonlinear frequency scale that approximates human perception, which is called the *mel-scale*. The mel-scale is given in Equation 2.2, where f denotes the frequency in *Hertz* and $B(f)$ denotes the mel-scale of f.

$$B(f) = 1125 \ln(1 + f/700) \tag{2.2}$$

Then, a filterbank with M triangular filters that are uniformly spaced in the mel-scale are constructed. The log-energy at the output of each filter is computed and discrete cosine transform of the M filter outputs gives as the MFCC. The coefficients are computed using

$$c[n] = \sum_{m=0}^{M-1} S[m]cos(\frac{\pi n(\frac{m-1}{2})}{M}) \quad 0 \leq n < M$$

2.1.3 Practical Issues

The typical speech recognition systems that use a speech signal sampled at 16kHz use the following parameters. The speech signal is short-time analyzed with Hamming window of 25 ms and with a window shift of 10 ms. Generally, 13 MFCC are used with delta coefficients and acceleration coefficients. The delta coefficients are the change of MFCC in time, and the acceleration coefficients are the change of delta coefficients in time [12].

2.2 Hidden Markov Models

The acoustics models can be constructed using the acoustic features that are extracted from a speech signal, and the transcription of that signal. The hidden Markov model (HMM) approach is the widely used statistical method. This section presents brief information about HMMs [14] [15].

An HMM has the following elements:

1. The number of states that the model has, N. The states will be denoted as $S = \{S_1, S_2, \cdots, S_N\}$.

2. The number of distinct observation symbols for each state, M. The individual observation symbols will be denoted as $V = \{v_1, v_2, \cdots, v_M\}$.

3. The state transition probability distribution for each state, $A = \{a_{ij}\}$. The probability a_{ij} denotes the transition probability from state i to state j.

4. The observation symbol probability distribution, $B = \{b_j(k)\}$. The probability distribution $b_j(k)$ denotes the probability of observing symbol k at state j.

5. The initial state distribution, $\pi = \{\pi_i\}$. The probability π_i denotes the probability of being at state i at $t = 1$.

The model parameters will be denoted in a compact form given as

$$\lambda = (A, B, \pi)$$

In addition to that, in the following sections the observation sequence that have T number of observations (it can also be considered as having observations starting at time $t = 1$ and ending at $t = T$) will be denoted as

$$O = O_1 O_2 \cdots O_T$$

2.2.1 Three Basic Problem for HMMs

There are three basic problems that needs to be discussed for HMMs:

1. **Evaluation Problem:** This is the problem of finding the probability that the given observation sequence is produced by the given model.

2. **Decoding Problem:** This is the problem of finding the most likely state sequence that the given observation sequence is produced by the given model.

3. **Training Problem:** This is the problem of adjusting the model parameters that will maximize the probability that the given observation sequence is produced by the given model.

2.2.1.1 The Forward-Backward Procedure

The first problem can be solved using the forward-backward procedure. The forward-backward procedure is composed of two procedures and the evaluation problem can be solved using one of them. For the forward procedure, the forward variable, $\alpha_t(i)$ is defined as follows:

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda)$$

Thus, $\alpha_t(i)$ denotes the probability of the partial observation sequence until time t and the state i at time t, given the model λ. We can solve the problem for $\alpha_t(i)$ inductively, using the following steps:

1. Initialization:

$$\alpha_t(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N.$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=t}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq t \leq T-1$$

$$1 \leq j \leq N.$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_{T}(i).$$

Similarly, the backward procedure is defined. In this procedure, the backward variable, $\beta_t(i)$ must be considered. This variable denotes the probability of the partial observation from time $t+1$ to T, given the state i at time t and the model λ and can be written as:

$$\beta_t(i) = P(O_{t+1}, Ot + 2 \cdots O_T | q_t = S_i, \lambda)$$

The problem can be solved for $\beta_t(i)$ using the following procedure:

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N.$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta t + 1(j), \quad t = T-1, T-2, \cdots, 1$$

$$1 \leq i \leq N.$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \beta_1(i).$$

2.2.1.2 The Viterbi Algorithm

The second problem, which is finding the single best state sequence can be solved using the Viterbi algorithm. The quantity, $\delta_t(i)$, which is the best score on a single path at time t and ends at state i is defined as follows:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P(q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t | \lambda)$$

By induction we have:

$$\delta_{T+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}) \tag{2.3}$$

The Viterbi algorithm finds the best state sequence, thus the state which maximizes the Equation 2.3 should be hold for each time t and state j. For this purpose, the term $\psi_t(j)$ is defined.

Using this definition, the Viterbi algorithm can be given as:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

$$\psi_1(i) = 0$$

2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T$$

$$1 \leq j \leq N$$

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

$$1 \leq j \leq N$$

3. Termination:

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

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

4. Path (state sequence) backtracking:

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

2.2.1.3 Baum-Welch Method

The third problem, the estimation of the model parameters, is the hardest problem and there is no analytic way to solve it. However, the parameters can be estimated using an iterative procedure, Baum-Welch method. This method maximizes $P(O|\lambda)$ locally for λ.

The description of the procedure is given below. For the estimation procedure the term $\xi_t(i,j)$ is defined as the probability of being in state S_i at time t, and state S_j at time $t+1$, given the model and the observation sequence. Hence, it can be written as:

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

Using the forward and backward variables we can rewrite $\xi_t(i,j)$ as:

$$\begin{aligned} \xi_t(i,j) &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\displaystyle\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

In addition to that, the term $\gamma_t(i)$, which is the probability of being in state S_i at time t given the observation sequence and the model can be related to $\xi_t(i,j)$ as:

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

If we sum these terms over the time interval, we get the following interpretations of these summations:

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

$$\sum_{t=1}^{T} \gamma_t(i) = \text{ expected number of times in } S_i$$

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

Using the above concepts, the re-estimation formulas for the HMM parameters π, A, and B can be given as:

$$\bar{\pi}_i = \text{expected number of times in } S_i \text{ at time (t = 1) } = \gamma_1(i)$$

$$\begin{aligned}\bar{a}_{ij} &= \frac{\text{expected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from } S_i} \\ &= \frac{\sum\limits_{t=1}^{T-1} \xi_t(i,j)}{\sum\limits_{t=1}^{T-1} \gamma_t(i)}\end{aligned}$$

$$\begin{aligned}\bar{b}_j(k) &= \frac{\text{expected number of times in } S_j \text{ and observing symbol } v_k}{\text{expected number of times in } S_j} \\ &= \frac{\sum\limits_{\substack{t=1 \\ O_t=v_k}}^{T} \gamma_t(j)}{\sum\limits_{t=1}^{T} \gamma_t(j)}\end{aligned}$$

The new parameters $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ can be estimated using the above formulas with the model parameters $\lambda = (A, B, \pi)$. This process can be used iteratively to reach a maximum likelihood estimate of the HMM. However, using this procedure only the local maxima can be reached.

2.2.2 HMMs for Speech Recognition

Speech signal properties change over time. Actually, the speech signal can be divided into intervals where it exhibits the same properties, which change over time. For this purpose, HMMs used for modeling speech signal have the property that state index increases or stays the same as the time index increases. This type of HMMs are known as *left-to-right* model or *Bakis* model [14].

In addition to that, discrete observation symbols may not be sufficient for modeling the speech signal accurately. Thus, continuous observation densities are used for the modeling. In this approach, the observations are continuous signals or vectors, and the observation probability distributions are modeled as Gaussians. Thus, the general representation of the observation probability distributions is given as:

$$b_j(\boldsymbol{O}) = \sum_{m=1}^{M} c_{jm}\mathfrak{N}[\boldsymbol{O}, \boldsymbol{\mu}_{jm}, \boldsymbol{U}_{jm}]$$

where $\boldsymbol{O}$ is the observation vector, c_{jm} is the mth mixture coefficient for state j and $\mathfrak{N}$ is a Gaussian density that has the mean vector $\boldsymbol{\mu}_{jm}$ and the covariance matrix $\boldsymbol{U}_{jm}$ [14].

2.3 Acoustic Units

One of the problems in the design of speech recognizers is the selection of the acoustic units. The acoustic units will be used to estimate the probability of the acoustic data given the word string.

In problems like, connected digit recognition or small-vocabulary speech recognition it is known that acoustic word models performs well [16] [17]. However, for a large vocabulary speech recognition task this approach is infeasible, since it is not possible to find enough occurrences for each word in the training database. Thus, *subword* units should be used. There are two criteria for the selection of subword units. The first one is the consistency of the units, i.e. every instance of the unit must have similar characteristics. The second one is the trainability which denotes that sufficient number of instances for each unit must appear for robust training. The researchers in the 1970's used syllables, diphones as acoustic units, these are consistent but difficult to train. On the contrary, selection of the units as phonemes brings the problem of consistency [18]. However, the phonemes are still used at the initial modeling phase and the models that are built using these units are referred to as *phone* or *monophone* models.

To solve the problem of consistency, the context of a phone should be considered.

It is very effective on the realization of a phone. Actually, initial and final portions of the phones are affected by the context. Thus, context-dependent phone units, where each phone is considered in its phonetic context, are emerged [19]. The successful application of the context-dependent phone units results in the concept of *triphone* models. In this approach, the immediately preceding and following phonemes are taken as the phonetic context [20].

Another problem with the acoustic units is the selection of the HMM topology. The current state of the art systems uses HMMs that have three emitting states and a left to right topology. The output observation distributions are continuous mixtures of Gaussian distributions, and the covariances of the Gaussians are constrained to be diagonal [3] [21]. A typical HMM that can be used for monophone or triphone modeling is given in Figure 2.1.

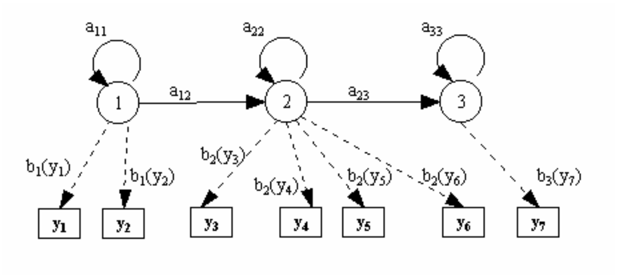

Figure 2.1: A left-to-right HMM that have three emitting states.

2.4 Parameter Tying

Triphone modeling solves the consistency problem by considering the context of a phone. However, for some of the triphones it may not be possible to find sufficient amount of training data. For this purpose, the idea of parameter tying can be used. With parameter tying, a part of the parameters or all the parameters between acoustic models can be shared. This will enable us to build robust models with small amount of training data. In addition to that, computation costs and memory space requirements can be reduced [22].

In Figure 2.2 [22], the hierarchical structure of the HMM parameters are given. The parameters of an HMM can be tied at any level of the hierarchy. The parameters can be tied within the model or across the models. The following rules are used for tying the denoted parameters [22].

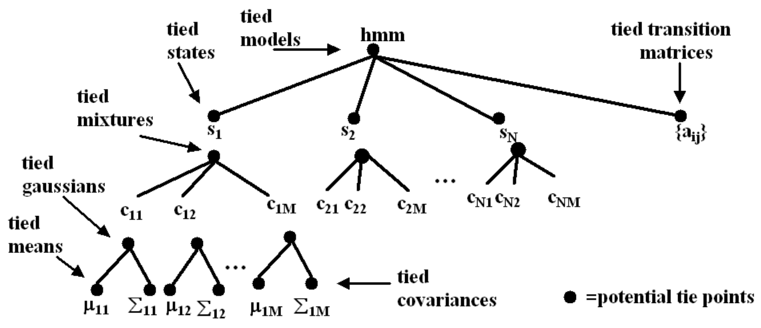
tied models
hmm
tied states
tied transition matrices
tied mixtures
s_1
s_2
s_N
$\{a_{ij}\}$
tied gaussians
c_{11} c_{12} c_{1M} c_{21} c_{22} c_{2M} c_{N1} c_{N2} c_{NM}
tied means
μ_{11} Σ_{11} μ_{12} Σ_{12} μ_{1M} Σ_{1M}
tied covariances
● =potential tie points

Figure 2.2: Hierarchical structure of HMMs.

1. **States:** Given the set of states that will be tied, the state with the largest overall variance is selected and all states are tied to this state.
2. **Mixtures:** All the mixtures that will be tied are merged.
3. **Gaussians:** The Gaussian that has the maximum mixture weight is selected, and all the Gaussians share these parameters.
4. **Means:** The means share the average of all the mean vectors.
5. **Covariances:** All variances share a matrix whose elements are the maximum of all the covariance matrices (only diagonal covariances are considered).
6. **Transition Matrices:** A randomly selected transition matrix is shared among the matrices.

2.4.1 Decision Tree Clustering

Decision tree clustering method uses phonetic decision trees for tying the parameters of the context-dependent HMMs to overcome the data sparseness problem. In addition to that, using this method enables to synthesize unseen context-dependent phone models. A phonetic decision tree is a binary tree where each node has a question attached. The questions are used to get information about the context of the investigated phone. An example decision tree is given in Figure 2.3 [23].

In this clustering approach, one tree is constructed for each state of each phone and using this tree the states of triphones are clustered. The tree is formed to maximize the likelihood of the training data. Initially, the states that will be clustered

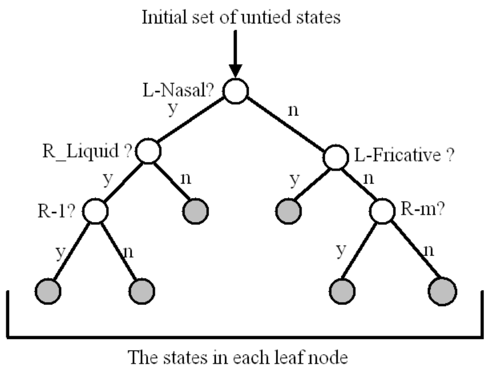

Figure 2.3: A sample decision tree.

are placed in the root node. All of these states considered to be tied and likelihood of the training data is calculated. Then, these states are divided into two groups using the question attached on the root node and the increase in the log likelihood is calculated. This process continues toward the leaves of the tree until the increase in the log likelihood falls below a threshold [23].

The unseen triphones can be built using the constructed trees. The unseen triphones' contexts are searched in the terminal nodes of the tree and the states of the unseen triphones are tied to the found state.

CHAPTER 3

LANGUAGE MODELING

Language models are used to estimate the probability of a word string. Thus, a probability $P(\boldsymbol{W})$ is assigned to every word string that is in a finite vocabulary $\mathcal{V}$. The word string $\boldsymbol{W}$ can be denoted as:

$$\boldsymbol{W} = w_1, w_2, \cdots, w_n \quad w_i \in \mathcal{V}$$

The probability $P(\boldsymbol{W})$ can be decomposed as:

$$P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i | w_1, \cdots, w_{i-1}) \tag{3.1}$$

The words $w_1, \cdots, w_{i-1}$ in Equation 3.1 is usually referred to as *history* and it is denoted by $\boldsymbol{h}_i$ [2] [24].

The histories must be mapped into some equivalence classes, otherwise there will be a huge number of distinct histories. Let $\Phi(w_1, w_2, \cdots, w_{i-1})$ denotes the equivalence class of the history $w_1, w_2, \cdots, w_{i-1}$. Then Equation 3.1 can be approximated by [2]

$$P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i | \Phi(w_1, w_2, \cdots, w_{i-1})) \tag{3.2}$$

3.1 N-Gram Language Modeling

In the current state of the art, the histories are treated as equal if they end in the same $n-1$ words. Thus n-gram modeling can be defined as:

$$P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i | w_{i-n+1}, \cdots, w_{i-1})$$

Even if the histories are mapped into some equivalence classes, n-gram modeling with $n \geq 4$ will result in huge number of equivalence classes. Because of that, *trigram* ($n = 3$) or *bigram* ($n = 2$) models are mostly used in the current speech recognition systems.

Direct estimation of n-gram models is done by just observing the frequencies of the words given the equivalence class of its history, and the frequency of that equivalence class. Assuming $C(\boldsymbol{x})$ denotes the count of a sequence of items, the n-gram probabilities can be estimated using the following equation:

$$P(w_i|w_{i-n+1}, \cdots, w_{i-1}) = \frac{C(w_{i-n+1}, \cdots, w_{i-1}, w_i)}{C(w_{i-n+1}, \cdots, w_{i-1})}$$

For example, the estimation of trigrams can be done using

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

The estimation of n-gram probabilities even in trigram and bigram modeling, will suffer from the data sparseness problem even large corpus is used [24]. For that purpose, to have reliable estimates smoothing must be performed.

3.2 N-Gram Smoothing

The problem of data sparseness is solved by using smoothing techniques. For English text in the size of several million words, more than 50% of trigrams occur only once and more than 80% of trigrams occur less than five times [12]. Thus, the estimates of the probabilities would not be realiable. Smoothing techniques make the probability estimates more robust, and aim to improve the accuracy of the model. In the following sections some smoothing techniquies will be presented.

3.2.1 Deleted Interpolation Smoothing

In this approach, higher order n-grams are interpolated with lower order n-grams. Trigram models can be interpolated with trigram, bigram and unigram probabilities as [2]:

$$\hat{P}(w_i|w_{i-2}, w_{i-1}) = \lambda_3 P(w_i|w_{i-2}, w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i)$$

with the constraint that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

The parameters λ_i either estimated using iterative methods or calculated with analytical methods. For both of these procedures, the training data is divided into sets. One group of data is used for finding the maximum likelihood estimates of n-gram probabilities, this group is called *kept* data and it consists of a large portion of the actual data. The rest can be used for finding the weights λ_i, and this group is called *held-out* data. For further information on deleted interpolation smoothing refer to [2].

3.2.2 Backoff Smoothing

The idea of backoff smoothing is to use the n-gram relative frequency if there are sufficient data for the estimation of an n-gram probability; otherwise, the lower-order n-gram probability that can be estimated reliably can be used for the estimation of the higher-order n-gram [2].

3.2.2.1 Good-Turing Estimates and Katz Smoothing

The Good-Turing estimate is a smoothing technique that can be used with infrequent n-grams. If we consider a text of size N words, where n_r denotes an n-gram that occurs r times, the following can be written [12] [25].

$$N = \sum_r r n_r$$

Good-Turing estimate of an n-gram which occurred r times is

$$\hat{P} = \frac{r^*}{N}$$

where

$$r^* = (r+1)\frac{n_r + 1}{n_r}$$

The modified count r^* is called discounting, and the ratio r^*/r is called a discount coefficient.

The Good-Turing estimate does not provide the combination of higher order models with lower order models. Katz smoothing [25] provides this by the idea that if the observation count for an n-gram is higher than some limit, k, the n-gram probability is modeled by the maximum likelihood estimate of that n-gram; if it is observed but smaller or equal to k, then Good-Turing estimate is used; and if it is not observed (n-1)-gram probability is used. This idea is given in Equation 3.3.

$$\hat{P}_n(w_i|h_n) = \begin{cases} P_n(w_i|h_n) & \text{if } r > k \\ d_r P_n(w_i|h_n) & \text{if } r \geq r > 0 \\ \alpha(h_{n-1}) P(w_i|h_{n-1}) & \text{if } r = 0 \end{cases} \tag{3.3}$$

where

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$$

$$\alpha(h_{n-1}) = \frac{1 - \sum_{w_j : r > 0} \hat{P}(w_j|h_{n-1})}{1 - \sum_{w_j : r > 0} P(w_j)}$$

r denotes the count of n-grams, k denotes the limit, and h_n denotes a history for the n-gram.

3.2.3 Class Based Models

The class-based n-gram models firstly map the words into some classes. Having this mapping all the n-grams are converted to their class n-gram forms and the probabilities are estimated over these classes. Let $\mathfrak{C}(w)$ denotes the class of the word w. N-gram probability can be given as:

$$P(w_i|w_{i-n+1} \cdots w_{i-1}) = P(\mathfrak{C}(w_i)|\mathfrak{C}(w_{i-n+1}) \cdots \mathfrak{C}(w_{i-1}))$$

There are two approaches to class-based smoothing; rule-based classes can be constructed, where class mapping is performed using some rules, and data-driven classes can be formed using some statistical methods. In addition to that, one word can be put into several classes using a probabilistic mapping [12].

3.3 Other Language Modeling Techniques

There are many techniques besides n-gram language modeling. They will be presented briefly in this section since only n-gram modeling is used in this thesis.

As described in [26], probabilistic context-free grammars (PCFGs) can be used for language modeling. PCFG is a context-free grammar where each rule has a probability. Context-free grammars assign a structure to a given string in the language, whereas probabilistic context-free grammars also assign a probability to the string. The probabilistic context-free grammars perform better than bigram language modeling in medium vocabulary (about 1000 words) speech recognition systems where the system expects sentences from a small set of grammar rules like in [27].

Maximum entropy approach is also used for language modeling [28]. Actually, it is used for estimating the conditional probability of a word given a context, then these probabilities can be used for the computation of the probability of a given string. In this approach, the conditional probabilities are assigned probabilities that maximize the entropy and that satisfy some constraints. The constraints are placed using binary feature functions over the context and the word. The conditional probabilities are estimated using iterative methods. The performance of the models depends heavily on the selection of features.

Structured language models [29] [30] are also used for this purpose. Structured language models assign a probability to every sentence and its every possible parse. These models are very effective in modeling the long-range dependencies [30]. However, syntactically annotated corpora are needed for training these models.

3.4 Complexity Measure for Language Modeling

The complexity of a language model can be measured using the concept of cross-entropy. The cross-entropy $H(\boldsymbol{W})$ of a language model on a word string $\boldsymbol{W}$ that has N_w words is given as [12]:

$$H(\boldsymbol{W}) = -\frac{1}{N_w}\log_2 P(\boldsymbol{W})$$

The perplexity of a language model is given as:

$$PP(\boldsymbol{W}) = 2^{H(\boldsymbol{W})}$$

The perplexity can be seen as the geometric mean of the branching factor of the text when given to the language model. Thus, high perplexity denotes a high branching factor from a word.

3.5 Vocabulary Selection

Recognition is performed over a search space that is composed of a vocabulary of a fixed size. The vocabulary is extracted from the text that the language model is trained. As the vocabulary size increases, it will grow the search space and the growth will increase the confusion of the recognizer. On the contrary, the decrease in the vocabulary size decrease the coverage of the test data. It is clear that recognizing any word that is not covered by the vocabulary will fail and also this will increase the probability of making an error while recognizing the next word. Thus, vocabulary size should be hold at an optimum point between these two conflicting points [2].

The words that occur in the test data and that are not covered by the vocabulary is called out-of-vocabulary (OOV) words. The rate of these words in the test data is referred to as OOV word rate.

3.6 Language Modeling in Agglutinative Languages

Word-based n-gram models does not work well for agglutinative languages. The agglutinative nature brings the problem of very fast vocabulary growth. This property also brings the problem of high out-of-vocabulary rates. These are caused by the derivation of different words from a basic word form [5] [31] [32] [33].

In order to overcome these problems, the general idea is to use subword units in language modeling, rather than to take words as units. There are several alternatives for the selection of subword units. One of them is using syllables as the units and training a language model over these units [31]. Another choice is to use morphemes as subword units, these are investigated in [31] [34]. In addition to that, using stems and endings as subword units is another approach [9] [10] [33]. Using subword units decreases the out-of-vocabulary rate and also increases the recognition performance [5] [7] [8] [9] [10] [31] [32] [33] [34].

3.6.1 Language Modeling for Turkish

There are a few studies on language modeling for Turkish continuous speech recognition [7] [8] [10].

In [7], stem based and stem-ending based bigram language models are used. In the first one a language model is trained over all the stems. The second approach uses stems and endings as different units and language models are trained over them. In both of the models endings which are one character long are not separated from the stem. The results show that stem-ending based models perform better.

In [8], also stem based and stem-ending based language models are used. However, in this study the endings that are one character long are also parsed. Two-pass recognition is performed. The first pass is performed with bigram models over stems, and the second pass is performed using trigram models over stems, and 'stems and endings'. The results show that stem modeling performs better than stem-ending modeling on the second pass. The bigram model that is trained in this experiment is trained with same data that is used in [7]. The trigram model is built using a larger corpus.

In the above studies, the test sentences are chosen from the corpus that is used to build the language model. These sentences are not removed before training the language models. Thus, the test sentences are modeled with the language model and there is no out-of-vocabulary words.

In [10], bigram language models are trained over words, syllables, and stems and endings. It is stated that stem-ending models perform better than the others. In addition to that, trigram models are trained over stems and endings and a rule-based weighted finite state machine (WFSM) is integrated with the language model. The WFSM is used for guaranteeing that the stems are followed by the correct group of endings. The second pass with the trigram model increases the performance effectively, however, the integration of the WFSM does not provide a good progress, it only increases the performance less than 1 percent.

CHAPTER 4

DECODING

The speech recognition problem is to find the best sequence of words given the acoustic signal. Since we are using the statistical pattern recognition approach, we have acoustic and language models as knowledge sources. Thus, the problem is a search process with the given acoustic and language models as knowledge sources.

The decoding process can be investigated in terms of two issues. The first one is the search strategy; the search can be conducted in the depth-first manner which is referred to as *time-asynchronous* strategy, or breadth-first manner which is referred to as *time-synchronous* strategy. The second one is the organization of the search space; the search space can be static or dynamic [12].

4.1 Basic Issues

The decoding process is a search that is conducted to find the word string, $\boldsymbol{W}$, that maximizes the probability of the word string given an acoustic evidence. As stated earlier, this problem can be formulated by [12]:

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W}|\boldsymbol{X}) = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})P(\boldsymbol{X}|\boldsymbol{W})$$

Using this equation a cost function is defined by taking the multiplicative inverse $P(\boldsymbol{W}|\boldsymbol{X})$, also to avoid the multiplications the logarithms are used. Thus, the cost function $C(\boldsymbol{W}|\boldsymbol{X})$ is defined as:

$$C(\boldsymbol{W}|\boldsymbol{X}) = \log[\frac{1}{P(\boldsymbol{X})P(\boldsymbol{X}|\boldsymbol{W})}] = -\log[P(\boldsymbol{W})P(\boldsymbol{X}|\boldsymbol{W})]$$

The following cost functions are also be defined:

$$C(\boldsymbol{X}|\boldsymbol{W}) = -\log[P(\boldsymbol{X}|\boldsymbol{W})]C(\boldsymbol{W}) = -log[P(\boldsymbol{W})]$$

Another problem is the integration of language model probabilities with the acoustic model probabilities. The language model log probabilities can be scaled by a constant called the *scale factor*. Moreover, a fixed penalty, which is called the *insertion penalty* can be added. If the scale factor is given as s, and the fixed penalty as p; then every language log probability x is converted to $sx + p$.

4.2 Viterbi Beam Search

Viterbi search is a time-synchronous search strategy that finds the most likely state sequence for the given observation sequence [12] [35]. When using Viterbi search for speech recognition, the most likely word sequence is approximated with the most likely state sequence and this is referred to as *Viterbi approximation*.

The Viterbi search is on the order of $O(NT)$ for space and on the order of $O(N^2T)$ for computation complexity, where N is the total number of states and T is the length of the observations. For large vocabulary speech recognition tasks, the complexities are so high that a pruning mechanism is needed.

A simple and very effective pruning mechanism that can be used with Viterbi search is the use of the *beam*. The beam is actually a threshold value, T_b. For each time t, the minimum cost D_{min} is identified, and then all the states having their costs grater than $D_{min} + T_b$ are pruned.

4.3 Stack Decoding

Stack decoding [2] uses A^* search. To find an effective heuristic function is very hard for speech recognition, therefore, time synchronous search performs better. However, stack decoding allows using language models that can handle long range dependencies.

To determine which node to expand an evaluation function is needed. For this purpose an evaluation function of the form $f(t) = g(t)+h^*(t)$ is used. In this function $g(t)$ is the likelihood of the partial search, and $h^*(t)$ is the heuristic function, which tries to estimate the likelihood of this search after time t. The computation of $g(t)$ is done using Viterbi algorithm. Partial paths are extracted word at a time, the node that is selected to be extracted is the one that maximizes $f(t)$.

This algorithm works in best-first search manner, where at each step the best hypothesis is expanded. In addition, the search space is extracted word at a time, thus the node that gives the best score is expanded by the candidate words. The candidate words are found using the language model; also, some fast matching acoustical methods can be used for further elimination of the candidate words.

4.4 Search Space Organization

Viterbi beam search is used in this thesis, thus in this section the search space organization is investigated only for time-synchronous search.

4.4.1 Effect of Language Model

As the order of the n-gram language model increases, the search space becomes more complex. The search space having two words for bigram language modeling is given in Figure 4.1. The bigram language model needs N^2 transition.

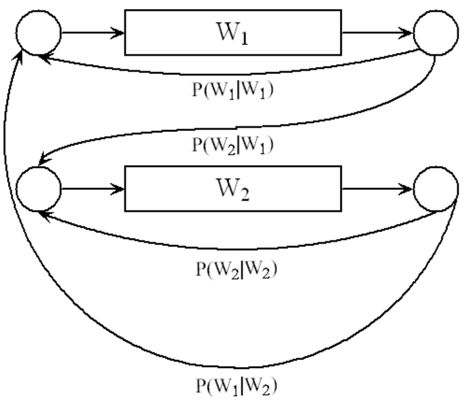

Figure 4.1: The search space that is constructed using a bigram model.

The complexity of bigram networks can be reduced by the introduction of the backoff paths. In this approach, only observed pairs have transitions between them, the unseen pairs are connected through backoff nodes and the bigram probabilities are modeled using unigrams with backoff weights. The use of backoff nodes is illustrated in Figure 4.2.

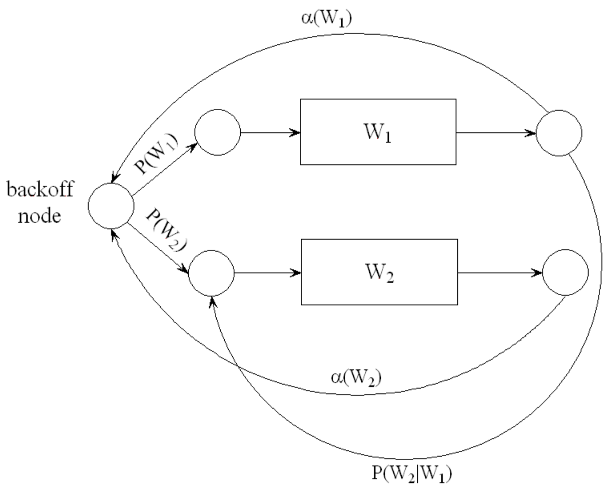

Figure 4.2: Search space using backoff nodes (α denotes backoff weights).

The search space becomes so complex with the trigram language models that the implementation of trigram search becomes infeasible. An example of trigram network that has two words is given in Figure 4.3 [12].

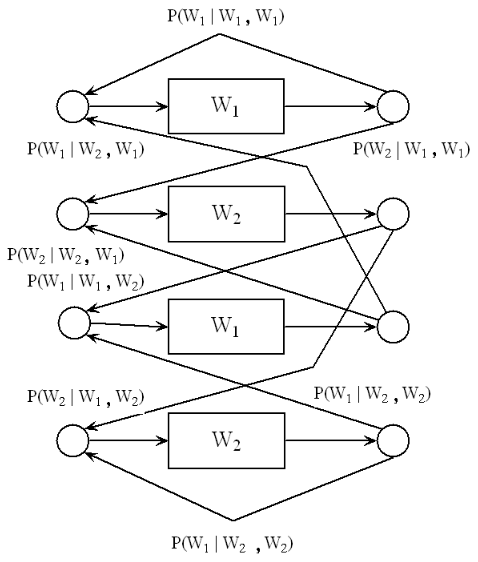

Figure 4.3: Search space that is constructed using trigrams.

4.4.2 Tree Lexicons

The representation of the lexicon also plays a role on the search space. Lexical trees, in which the common beginning phonemes are shared, the state search space is reduced.

One draw back of the lexical tree is the word in the lexicon is determined only when the leaf node is reached. Thus language models can be applied only at the end of the tree. For that purpose, a separate copy of the entire tree should be kept for each language model history. The search space using lexical trees for bigram language models is given in Figure 4.4 [12].

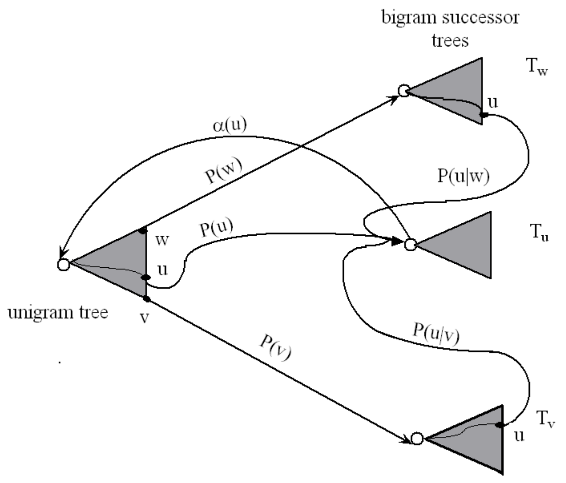

Figure 4.4: Search space that is constructed using lexical trees.

4.5 N-Best Decoding

High order language models make the search space very complex that the search on that network would be infeasible. One of the solutions that enables us to use higher order models is to use multi-pass decoding. In the multi-pass decoding approach, a simple language model is used to reduce the search space, and then complex models can be used over the reduced space.

The most simple approach is called n-best decoding. N-best decoding can be implemented only by holding N cost values for each state and the corresponding histories, and at each time step, the costs that are in top N scores will be selected. At the end of the recognition we have a list of n-best hypotheses.

There are more compact representations than n-best list like word lattices. Word lattices are composed of word hypothesis. Each word hypothesis has an explicit time interval and a score. All the word hypothesis that are in the non-overlapping time intervals are linked together.

In order to perform a second pass over the generated lattices with higher order language models, the lattices must be expanded to the same order [36]. This must be done since each word in the lattice must have a unique history of the same order. An illustration of an expansion from bigram to trigram is given in Figure 4.5 [36].

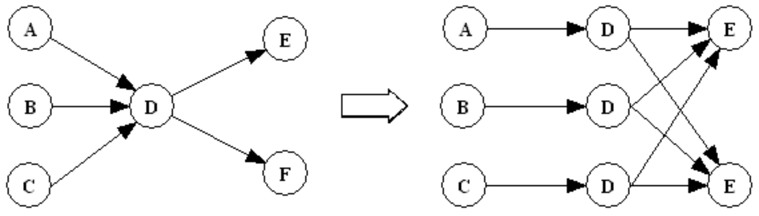

Figure 4.5: Lattice expansion form bigram to trigram.

4.6 Evaluation of the Recognition

The recognition performance can be evaluated in terms of two measures; recognition rate, and accuracy on the word level. Sentence recognition rate is also used, which is the ratio of number of correctly recognized sentences to number of all the sentences. There are three types of errors that a recognizer could do:

1. Substitution (S): an incorrect word is placed instead of the correct one
2. Deletion (D): a correct word is omitted in the sentence
3. Insertion (I): an extra word is added to the sentence

Using these definitions the following terms are defined, if N denotes number of total words.

$$\text{Word Recognition Rate} = \frac{N - S - D}{N}$$

$$\text{Accuracy} = \frac{N - S - D - I}{N}$$

CHAPTER 5

EXPERIMENTAL WORK

In this chapter, the experimental work on Turkish large vocabulary continuous speech recognition that is conducted will be presented.

5.1 Previous Work

The previous work on language modeling is presented in Chapter 3. The selection of the language model strategy greatly effects the performance of Turkish continuous speech recognition. Thus, the main issue that is stressed is the language modeling.

It would be very beneficial to compare the result of this thesis with the previous studies. However, the results in [7] and [8] could not be compared with the results of this study, because the test data is a subset of the training data that is used for language modeling in [7] and [8]. This results in a high recognition performance that could mislead researchers on the performance of the language models that are used.

The study given in [10] is very similar in the approaches that is used for language modeling. Thus, the results seem comparable. However, because of the differences in the size of training data that is used and in the size of the test data, the results should not be compared directly. The results given in this thesis are better than the results in [10], however, direct comparison is misleading. The results of [10] is given in Table 5.1.

5.2 Audio Corpora

We have two sets of corpora. The first one is used for training the acoustic models and the second one is used for testing the recognition performance. The details of

Table 5.1: Recognition results of the using stem-ending-based language models (SRR: ′sentence recognition rate′, WRR: ′word recognition rate, WFSM: ′weighted finite state machine′)

	SRR	*WRR*
Bigram	11.74%	59.49%
Trigram	19.70%	66.94%
Trigram + WFSM	19.77%	67.46%

these corpora is given in the following subsections. All the recordings are sampled at 16kHz and quantized with 16 bits.

5.2.1 Training Corpus

The audio corpus that is used to build acoustic models is collected by METU. There are 2462 distinct sentences in the database. 193 speakers (89 female, 104 male speakers) are recorded. Each speaker reads 40 sentences, which are selected randomly from the set of sentences. The speakers are mainly students, faculty and staff in METU. The audio corpus is constructed using this database. The audio files are segmented into sentences and the orthographic transcription of each sentence is written into a text file. The misreadings are corrected in the transcription file, or completely deleted [37]. After this operation, the audio corpus contains 6935 sentences.

5.2.2 Test Corpus

The audio recordings that are used for testing the speech recognition system consist of 223 sentences having totally 1862 words that are taken randomly from the text corpus that will be presented in the next section. There are 6 speakers (1 female, 5 male speakers), who are also recorded in the training corpus. The vocabulary size of the test data is 1288.

5.3 Text Corpus

The text corpus that is used is composed of two different parts. The first portion is the one that is used in [8] and the test sentences are chosen from this part. The second part is collected recently.

The data is composed of sport news, that are collected from the web pages of the newspapers in the Internet. A single context, sport, is used since a language model extracted from a context performs better in that context.

5.4 Hidden Markov Model Toolkit (HTK)

HTK is used for conducting our experiments. HTK is a toolkit for constructing HMMs. Although general purpose HMMs can be built using HTK, it is designed for speech recognition tasks and it is widely used in the area. In this section, HTK will be briefly presented, for further information refer to [38].

A complete speech recognition system can be built using HTK. Thus it is a complete toolkit that can perform front-end parametrization, building and training HMMs, constructing n-gram language models, and viterbi decoding. The HTK tools that are used in this thesis are described below.

Cluster performs data-driven clustering of words into classes using n-gram occurrences of the words.

HBuild is used for building standard HTK lattice using a bigram language model. The resulting network can be used in the decoding process as a search space.

HCompV is used for the computation of the global mean and covariance of a set of data. This is especially useful for the initialization of the acoustic units with the global mean and covariance.

HCopy is used for feature extraction from the speech signal.

HERest is be used for the training of the acoustic models.

HHEd is used for a large variety of HMM modifications. These involve parameter tying, mixture incrementing, and modification of state transitions.

HLEd is used for manipulating label files i.e. transcription files. Using this tool, monophone transcription files can be converted into triphone transcription files, also all observed triphones in the training data can be listed .

HLRescore is used for post-processing of the generated lattice files. It is mainly used for expanding the lattices with a higher-order language model.

HResults is used for evaluating the recognition results.

HVite is a general purpose Viterbi decoder. The tool performs search on a static network. It can provide two basic operations. The first one is *forced-alignment*, in which phonetic alignment is done on the training data; this is mainly used when multiple pronunciations available for a word. The second one is to do recognition to find the most likely sentence of a given set of observation vectors. N-best recognition can also be performed, in this mode word level lattices can be produced.

LBuild is used to construct back-off n-gram language models using the n-gram data that are created with 'LGCopy'.

LGCopy is used to construct n-gram files, to limit the size of the vocabulary, and also to map word n-gram data to class n-gram data using a word-to-class mapping.

LGPrep is used for extracting all n-grams that occur in a given text. This tool also adds the newly seen words into the word map file.

LNewMap is used for creating an empty word map.

5.5 Acoustic Model Training

Acoustic models are trained by using the training part of the audio corpus. HTK tools provide all needed mechanisms for training acoustic models. However, the data should be composed into some format for using with HTK. The next section continues with the data preparation part.

5.5.1 Data Preparation

The audio corpus was containing some errors. Some of these were transcription errors for words or a group of words. The other errors were mismatches between the audio file and the transcription file. These errors should be corrected. In addition to that, Turkish *speech assessment methods phonetic alphabet* (SAMPA) [39] was to be used for the phonetic modeling of the language. Thus, SAMPA transcription of the files should also be done. Thus, each sentence is listened and the transcription files were modified according to the errors. This was a very time consuming and also error prone procedure.

Using the transcription files, phonetic transcription files that contain the monophones that will be used in the training process are extracted automatically. The list of monophones were stored in a different file. All of the monophones that are in Turkish SAMPA is used.

In addition to that, feature vectors for each audio file were extracted and stored in a file to avoid the re-computation of the feature vectors at each iteration. This was done using the parameters:

1. Parameter kind: Mel-frequency cepstral coefficients with delta and acceleration components and using 0^{th} coefficient
2. Window: Hamming window of length 25ms with 10ms shifting
3. Number of channels: 23 - 27
4. Number of coefficients: 12 - 14

The topology of the HMMs that will be used should also be determined. We have used a simple left-to-right topology, with three emitting states. HTK tool places a non-emitting state at the beginning and end of an HMM; this brings an ease to the concatenation process. The topology is given in Figure 5.1. Using this topology, a general prototype model is constructed. The observation probability distribution uses a continuous density Gaussian that has a diagonal covariance matrix. The prototype has zero means and the diagonal elements of the covariance matrix is all ones. The transition matrix is non-zero if there is a transition between the states, and zero if there is no transition. The non-zero values are selected as to satisfy the property that the sum of the transition probabilities for each state is 1, and they are chosen randomly.

5.5.2 Training

The training process contains several steps. At each step, the system is modified, and after each step re-estimation of the acoustic model parameters is done several times.

The initial values for the states and variances were computed using the prototype model. The tool 'HCompV' computes the global mean and variance of the training

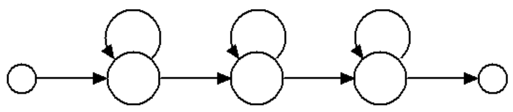

Figure 5.1: The HMM topology for the phone models (small states denotes none emitting state).

data. Using the global values, an HMM model for each monophone were created. In addition to that, a silence model, which models the silent portion at the beginning and at the end of the speech, was built with the same mean and variance values and with the topology given in Figure 5.2.

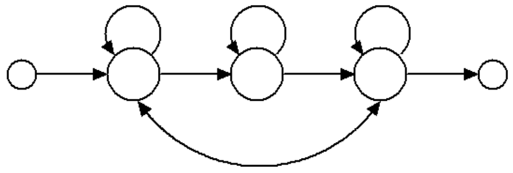

Figure 5.2: The HMM topology for the silence model (small states denotes none emitting state).

After the initialization of the models, the training of these models are done using the tool 'HERest'. This tool performs embedded training for the models, for this purpose, phonetic segmentation is not needed. We have performed 10 iterations, however, the number of iterations can be adjusted by observing how much improvement is gained after each iteration.

The next step is the introduction of the short-pause model. The short-pause model is used to model the pauses of the speaker between words. Since there may not be any pause between the words, the short-pause model uses *tee-model*. In the tee-model, a transition from the non-emitting starting node to the non-emitting final node is added [38]; with this approach if there is no pause between the words, this model is skipped without consuming any observation. In addition to that, short-pause model has only one emitting state and this state is tied to the third state of the silence model. The topology of the short-pause model is given in Figure 5.3. The transcription files were modified and a short-pause model is placed after the phonetic transcription of each word. The introduction of this model is followed by 10 iterations of embedded training.

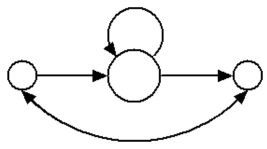

Figure 5.3: The HMM topology for the short pause model (small states denotes none emitting states).

The context-dependent phone models are added to the system at this step. The triphones are constructed from the monophone models. The list of triphones that occur in the training data is extracted and the monophone transcriptions are expanded to triphone transcriptions using the tool 'HLEd'. There are totally 6433 distinct triphones in the training data. In addition, all the triphones in the triphone list are constructed using the tool 'HHEd'. Also at this step, the transition matrices of the triphones that model the same phone are tied. The triphone models are trained with two iterations.

There is not enough data for training all the triphones individually. The states of the triphones should be tied for reliable estimation. Decision tree clustering method is used with the tool 'HHEd', since it allows adding triphones that are not seen in the training data. The decision tree used is adapted from [7]. Different threshold values are used for state tying, thus different number of physical triphones left at the end of the clustering process. The results of using different threshold values is given in the next section. After the decision tree clustering procedure, the unseen triphones are added to the system using the decision tree. All possible combinations of the monophones are added as a context for each triphone. It is clear that some of these triphones will not occur in the natural data, but adding all of them would not affect the computation performance of neither the training procedure nor the decoding part. Five iterations of embedded training are performed on the clustered triphones.

The observation probability distributions of the states are single Gaussian distributions. With the aim of increasing the accuracy of the models, the observation probabilities are split into several mixtures. Mixture incrementing is performed using the tool 'HHEd'. The mixtures are incremented in a multiple steps as 3, 5, 8, and 12. After each incrementation, training is performed for 5 iterations.

Table 5.2: Comparison of the models having energy parameters with the models having zero cepstral coefficients.

	Word Recognition Rate	*Accuracy*
Using Energy Coefficient	64.50%	24.60%
Using Zero Cepstral Coefficient	78.30%	57.57%

Finally, we have triphone models that can be used for recognition. The next section discusses the performance of the acoustic models.

5.5.3 Acoustic Model Performance

The performance of the acoustic models are investigated to choose the best acoustic model among all the trained models. To test the acoustic performance no language model is used. Moreover, only the vocabulary of the test sentences is used to decrease the search space. Thus, the following results do not reflect the actual performance of the models in large vocabulary recognition.

The first comparison is made between the acoustic models that are used in [7] [8] and the models that use the zero cepstral coefficient instead of the energy parameter. The results are given in Table 5.2.

The second comparison is among the models with different number of cepstral coefficients and different number of filters. The word recognition rate (WRR) and accuracy (ACC) of these models are given in Table 5.3 and Table 5.4 respectively.

Table 5.3: Comparison word error rates of acoustic models that have different number of coefficients and filters

No. of Filters	*23*	*24*	*25*	*26*	*27*
No. of Coefficients					
12	78.73%	79.22%	79.11%	79.00%	79.16%
13	78.86%	79.65%	79.75%	79.05%	79.11%
14	80.67%	80.24%	80.34%	79.27%	79.75%

The best model in terms of word recognition rate and accuracy is the model with 14 cepstral coefficients and 23 filters.

The next performance is tested with the level of clustering and the number of Gaussian mixtures. The word recognition rates of models in Table 5.5 and the accu-

Table 5.4: Comparison of accuracies of acoustic models that have different number of coefficients and filters

No. of Filters / No. of Coefficients	*23*	*24*	*25*	*26*	*27*
12	58.00%	58.54%	58.49%	58.27%	58.59%
13	60.20%	59.13%	59.77%	58.49%	58.11%
14	61.22%	60.85%	61.01%	59.72%	59.77%

racy of the models are given with different number of physical HMMs and mixtures in Table 5.6.

Table 5.5: Word error rates for different number of physical HMMs and Gaussian Mixtures.

No. of Physical HMMs / No. of Gaussian Mixtures	*3019*	*3910*	*4667*
8	76.85%	81.26%	79.22%
12	78.84%	81.58%	78.79%

Table 5.6: Accuracies for different number of physical HMMs and Gaussian Mixtures.

No. of Physical HMMs / No. of Gaussian Mixtures	*3019*	*3910*	*4667*
8	55.32%	62.84%	57.25%
12	57.79%	63.64%	56.02%

The best acoustic model performance is reached by the model that uses 14 cepstral coefficients and 23 filters, and that has 3910 physical HMMs, which have 12 Gaussian mixtures for their observation probability distributions. In the following sections of this thesis, this acoustic model is used.

5.6 Language Model Training

The language models are trained with two portions of the data. The first part was corrected for the study [8]. However, the second one was full of errors. The data preprocessing and error correction procedures are presented in the next section.

5.6.1 Data Preparation

The data preparation part involves two steps. In the first step, the errors in the data are corrected or cleaned, and it is brought to a format where each sentence is on a new line. Then, the morphological processing is conducted on the data in order to prepare it for language model training.

5.6.1.1 Tailoring the Data

The second portion of the data was containing too many errors. The authors' knowledge of Turkish writing rules were poor; for example, there were many errors about the clitics 'de' and 'mi', and also about the use of apostrophe for the inflected proper names. Another problem with this part of the data was that, it was containing too many foreign proper names. Thus, it was a very problematic corpus.

In the first step, a distinct word list is extracted from the data. This word list is sent to TurkLex which is the morphological analyzer [40]. TurkLex contains PC-Kimmo rules for Turkish morphology. The parsed words are put into the known words list and the words that have no parses are put into the unknown list. Some of the words in the unknown list are corrected manually. The rest is tried to be corrected automatically; for each word in the unknown list, the known list is searched for a similar word. If a similar word is found, the unknown word is mapped to the known one. Two words are considered similar if there is a letter repetition and the rest is the same, or if two letters are swapped and the rest is the same. At the end of this step, a correction list is constructed for both the manual corrections and the automatic corrections.

The next step was the segmentation of the data into sentences. Each sentence is written between the labels '<S>' and '</S>'. The determination of a piece of text as a sentence was as follows. If one of the punctuation marks that occurs at the end of sentences ('.', '!', '?', and '...') is found and if this mark is not between single or double quotations, then the sentence boundary is placed. To avoid the problem of placing sentence boundaries in the abbreviations like, 'Prof.Dr' some known abbreviations are entered manually, also periods that occur between letters is not determined as a boundary.

After this step, the sentences that have less than three words and the sentences that have more than 50 words are eliminated, since the sentence boundaries for these sentences may not be placed correctly.

There were too many foreign proper names in the text. To decide which are the proper names, the non-initial words of a sentence that begin with a capital letter and the words that have an apostrophe mark are marked as proper names. These are listed with their occurrence counts. In addition to that, some of the proper names are listed manually.

In the next step, text is preprocessed. The first aim of this process is to eliminate some meaningless characters that occurred when copying the text from the Web. The second aim is to convert each character into lower case and to represent Turkish characters that are not in the English alphabet with the upper case letters of their orthographically similar letter, i.e. to represent 'ç' with 'C'. Some punctuation marks that occur one after the another are singled. Finally, all the punctuation marks but the apostrophes are deleted from the text, since the punctuation marks are not used in language modeling but the apostrophes are necessary for the morphological analysis of the proper nouns.

There are numerical values in the text. They are converted to the corresponding word forms and placed between the labels '<NUM>' and '</NUM>' automatically. The derivational morphemes and inflectional morphemes are also marked for the numerical values.

The proper names that are extracted manually and also the most frequent 100 proper names that are extracted automatically are marked in the text. They are placed between the labels '<PROP_N>' and '</PROP_N>'. The most frequent 100 names are added, since some words may be misclassified with the presented method of proper noun extraction.

Furthermore, abbreviations are found by searching for periods. The abbreviations are listed and the corresponding long forms are mapped to these words manually. Using this mapping, the abbreviations are replaced with their actual forms.

Finally, the sentences that contain one or more unknown words are deleted from the text. As a result, about 5.7 million word text that can be used in language modeling is extracted from a corpus of 20 million words. To correct the rest of the data, proper names should be listed manually, and also some of the erroneous words should be corrected manually.

The first part of the data was corrected manually in [8], thus there was no need to perform the above steps for this part of the data.

5.6.1.2 Morphological Processing

The general approach for building language models for agglutinative languages is to use subword units especially stems, as mentioned earlier. Thus, the text corpus should be morphologically analyzed to divide the words into their stems and endings. We have one more time used TurkLex for finding the unknown words.

The first part of the text data contains test sentences. In [8], the test sentences are not excluded from the text that is used to train the language model. However, the test sentences are deleted from the first part of the text in this study, since as language model that contains the test sentences dominates the decoding process, the recognition performance increases. Thus, first portion of the text is analyzed directly after deleting the test sentences. The second part is also processed with the analyzer but the proper names and the numeric expressions are not passed to the morphological analyzer.

The main problem of morphological analysis, is the morphological ambiguity. Thus, the analyzer produces multiple parses for some of the words. Thus, a selection should be made among these parses. The parse with the shortest stem is selected for each multiple parse. This will not guarantee to find the correct parse at each step, however, we do not have a morphological disambiguator that can be used for that purpose.

Using the morphologically parsed data, we have extracted three groups of files. One of them is the word file; it contains no labels except the sentence boundaries. The second one is the stem file. The final, one is the stem-end file; this file contains stem and endings separately on each line, also endings are preceded with a '+' sign to avoid confusion.

The above files may be used directly for training language models. Before going into the details of language model training, the statistics of the text that will be used in the language modeling is given in Table 5.7.

Table 5.7: Statistics for language modeling text

No. of Total Words	6,507,742
No. of Sentences	534,666
No. of Distinct Words	161,013
No. of Distinct Stems	45,293
No. of Distinct Endings	4,622
No. of Distinct Stems and Endings	49,915

5.6.2 Training

The training of the language models are performed using the language modeling tools of HTK. HTK provides n-gram language modeling and back-off smoothing. In addition to that, class-based language models can also be trained. In this study, three different language models will be tested; class-based word models, stem models, and stem-ending models. The training process of these models are given in the following sections.

5.6.2.1 Class-based Models

This model will be based on the words, since the vocabulary size of the text is very large. The vocabulary size should be limited. Before training the language model, a list of words with their occurrence counts is formed and the words that occur more than 5 times are put into the limited vocabulary list. Thus, the vocabulary size is decreased to 44,678 from 161,013.

First, all the n-gram occurrences should be prepared before training a language model or clustering the words. For this purpose, the following steps are followed using the HTK tools on the word file. First, 'LNewMap' is used to create an empty word map. Secondly, the bigrams are extracted using 'LGPrep', this tool also updates the word map file by adding newly seen words. The next step is to convert the extracted bigrams to data files that contain the counts of the bigrams using 'LGCopy'. Finally, the vocabulary limit is put by one more execution of 'LGCopy'. This maps the words that are not in the limited vocabulary to an *unknown* symbol, modifies the data files, and maps these words to the unknown symbol.

Class-based language modeling techniques, clusters words into several classes, and then builds language models over these classes. Using the bigram occurrences reached in the previous step, data-driven clustering is performed. 'Cluster' is used for clustering 44,678 words into 2500 clusters. This procedure works in the following way. It takes each word, places it to each cluster, and calculates the bigram or trigram likelihood of the text when the considered word is in the considered cluster one by one. At the end, the considered word is placed to the cluster that maximizes the likelihood. This process is very complex in time, thus, it is very time consuming [38].

The language model is constructed over the classes. The bigram occurrences in the data files for the limited vocabulary are replaced with the corresponding class bigram occurrence by using the class map with another execution of 'LGCopy'.

The class-based language model is constructed afterwards by using 'LBuild'. In this study, bigram and trigram class-based language models are constructed. For the trigram model, trigram occurrences in the data files are used but the clustering also done with respect to bigram probabilities.

The limitation of the vocabulary brings the problem of out-of-vocabulary words. The out-of-vocabulary rate of our test sentences is 5.42%.

5.6.2.2 Stems

The number of distinct stems in the training text is not very large, however, the recognition is conducted over the words. Thus, the word size should also be limited here. To be able to make a reliable comparison between the two models, the same limited vocabulary list that is used in class-based modeling is used with stem modeling. The limiting word vocabulary decreased the size of the stem vocabulary from 45,293 to 17,198.

The sequence of operations using the tools 'LNewMap', 'LGPrep', 'LGCopy', and 'LGCopy' is performed on the stem file as in the class-based language modeling.

The language model is constructed using 'LBuild' over the n-gram occurrences of the stems. Bigram occurrences are used for bigram modeling and trigram occurrences are used for trigram modeling.

The vocabulary of this model is the same with the above model, thus, the out-of-vocabulary rate is the same for the test data.

5.6.2.3 Stems and Endings

The number of distinct stems and endings are close to the number of limited words in the previous models. Thus, for stems and endings model a limit would not be placed on the vocabulary size. Hence, a language model with the vocabulary size of 49,915 is trained. Stems and endings are used as modeling units for training. The history relationships for bigram and trigram modeling is illustrated in Figure 5.4 and in Figure 5.5 respectively. It can be seen in Figure 5.4 that, in bigram modeling the information about co-occurrences of stems cannot be modeled if there is an ending between two neighbor stems.

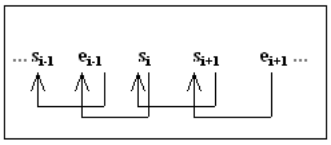

Figure 5.4: Illustration of the history relationship in bigram modeling.

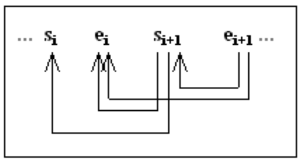

Figure 5.5: Illustration of the history relationship in trigram modeling.

The same sequence of operations is performed to extract the n-gram data files. However, vocabulary would not be limited. Thus, the procedure starts with 'LNewMap' creating an empty word map, then using 'LGPrep', n-gram occurrences are listed and with 'LGCopy', the n-gram occurrences are converted to data files.

The language model is constructed using 'LBuild'. We have built both a bigram language model and a trigram language model.

This modeling scheme divides the stems and endings, thus the coverage of the model in terms of vocabulary is higher than the other schemes. The out-of-vocabulary rate of the test data is about zero. Only 7 stems and 5 endings that are observed in the test set is not covered.

5.7 Recognition of the Test Data

The experiments on recognition using the previously trained acoustic and language models are presented in this section.

5.7.1 Data Preparation

The data preparation in testing phase involves parametrization of the test data, building the pronunciation dictionary, and building the static search space for each approach in the language modeling.

The parametrization of the test data is done using the tool 'HCopy'. The parameters used in feature extraction must be consistent with the parameters that the acoustic models uses. Thus, the parameters of the feature extraction is the same with the parameters of the training data with 14 cepstral coefficients and 23 filters.

The pronunciation dictionary is very critical in the search process, since the word models in the search space are replaced with corresponding sequence of acoustic models using these pronunciations. Thus the phonetic models corresponding to the pronunciation of the items should be given precisely. Because of the size of the vocabulary, an automatic tool that implements the mapping of phones to given orthographic representation is used. The output of this tool is not compatible with SAMPA, therefore these pronunciations are converted to SAMPA using simple mappings.

In HTK, the dictionary entries consist of the following fields; word, output symbol (optional), pronunciation probability (optional), and pronunciation. In the decoding process, the acoustic models for words are constructed using the given pronunciation. If there are multiple pronunciations, parallel acoustic models are constructed for the same word. The output symbol is given as the recognized string, if the corresponding word is recognized. This mechanism can be used to perform recognition using class-based and stem based language models. The class or stem of a word is put into the 'word' field and the actual word is put into the 'output symbol' field, and the pronunciation of that word is given in the pronunciation part. A sample from the dictionary that is used for decoding using class-based models is given in Table 5.8.

HTK performs viterbi-beam search over a static network. Thus, the search space must be constructed before the search process starts. The tool 'HBuild' constructs a

Table 5.8: A Sample from the pronunciation dictionary

...		
CLASS10	zehirli	z e h i r l i
CLASS10	yararlI	j a r a r 5 1
CLASS11	ve	v e
CLASS11	ama	a m a
...		

search space using a vocabulary list and a bigram language model. The transition probabilities among the words are assigned using the bigram language model. Thus, using this tool and the bigram models we have trained at each step, the search spaces for all of the language model strategies are constructed.

5.7.2 Decoding

Decoding is performed to test the performance of the language models. However, to see the contribution of the language model on the performance of large vocabulary speech recognition, we have performed recognition without any language model with the same vocabulary of stem and class-based language models i.e. with 44,678 words. The results are given in Table 5.9.

Table 5.9: Recognition Performance without a Language Model (OOV: 'out of vocabulary')

Sentence Recognition Rate	*Word Recognition Rate*	*Accuracy*	OOV Rate
1.35%	49.81%	12.14%	5.42%

Later on, the decoding is performed using the language models that are trained. In the first step bigram language models are used. Different 'language model scaling' and 'insertion penalty' values are used. These values are fixed by just observing the recognition results of first 10 sentences. The 'beam width' is selected as 300 and 'word end pruning' value is selected as 50. Even that much pruning decoding over all test sentences last about 8 hours. The results of the experiments are given in Table 5.10.

Only bigram language models can be used in the first pass of the recognition in HTK, because HTK conducts search on a static network. To use higher order n-grams two pass decoding must be performed. For that purpose, n-best recognition using the bigram language models are performed and lattices are generated for each sentence.

Table 5.10: Recognition Performance using bigram Language Models (SRR: 'sentence recognition rate', WRR: 'word recognition rate, OOV: 'out of vocabulary', s: 'language model scale factor', p: 'insertion penalty')

	SRR	*WRR*	*Acc*	*OOV Rate*	*s*	*p*
Class-Based LM	26.01%	76.53%	74.06%	5.42%	15	15
Stem LM	15.77%	72.38%	67.80%	5.42%	8	-5
Stem and Ending LM	19.46%	72.36%	70.20%	≈ 0.00%	8	-5

10-best recognition is performed for classed-based and stem based language models; 15-best recognition is performed for 'stem and end' language model. The maximum possible recognition rates that can be obtained from 10-best recognition for all the models are given in Table 5.11.

Table 5.11: Maximum possible recognition rates (SRR: 'sentence recognition rate', WRR: 'word recognition rate, OOV: 'out of vocabulary')

	SRR	*WRR*	*Acc*	*OOV Rate*
Class-Based LM	39.01%	82.92%	81.31%	5.42%
Stem LM	22.52%	77.94%	74.87%	5.42%
Stem and Ending LM	34.39%	80.53%	79.02%	≈ 0.00%

The lattices that are constructed in n-best recognition are expanded with the trigram models and recognition performed on these lattice using trigram models. The 'beam width' is given as 500 and 'word end pruning' value is given as 250 the scale probability of the language model and the insertion penalty is given as 15 for all models. Since the search is performed over the lattices, it runs very quickly. The results are given in Table 5.12.

Table 5.12: Recognition results of the second pass with trigram models (SRR: 'sentence recognition rate', WRR: 'word recognition rate, OOV: 'out of vocabulary')

	SRR	*WRR*	*Acc*
Class-Based LM	26.13%	77.78%	74.92%
Stem LM	17.12%	73.46%	69.42%
Stem and Ending LM	30.32%	78.54%	76.38%

As can be seen from the results, class-based bigram language model performs better than stem-based and stem-ending-based language models. Actually, stem-

Table 5.13: Number of sentences that have erroneous words in class-based language models.

Number of Erroneous Words	*Number of Sentences*
1	54
2	37
3	16
4	18
5	14
6	6
7	4
8	2
9	2

Table 5.14: Number of sentences that have erroneous words in stem language models.

Number of Erroneous Words	*Number of Sentences*
1	59
2	43
3	24
4	25
5	9
6	7
7	5
8	1
9	1
10	1

Table 5.15: Number of sentences that have erroneous words in stem-ending language models.

Number of Erroneous Words	*Number of Sentences*
1	51
2	40
3	24
4	17
5	8
6	3
7	1
8	3
9	1

based models can also be seen as a class-based model, where the words having the same stems are clustered into the same classes. However, since the class-based model has smaller number of classes with respect to the number of stems, the estimates of the class bigram probabilities are more reliable than stem based models. The stem-ending bigram language model, on the other hand, does not provide the information that which stem occurs after an inflected stem. This information loss, may explain us the reason that these models perform worse than the class-based model.

Using Table 5.13, Table 5.14, and Table 5.15 it can be seen that, in the unrecognized sentences most of the time only one or two words are recognized.

The stem-ending based model almost covers the whole vocabulary of the test sentences, where the other two models have about 5 percent of out of vocabulary rate. In Table 5.16, it can be seen that about 30 percent of the sentences contain out of vocabulary words. Thus, with the limited vocabulary, 30 percent of the sentences cannot be recognized. This stresses the importance of the vocabulary coverage. Hence, it is clear the stem-ending-based language model performs best in the last experiment because of the high coverage of the vocabulary. In addition to that, in the trigram modeling over stems and endings the information about the co-occurrences of neighboring stems is not lost. This must be the reason of the improvement of the stem-based model. The clustering of the words is done on the bigram occurrences, thus the word to class mappings may not be significant for the trigram model. This explains the little improvement, while we were expected a greater one.

Table 5.16: Number of sentences that have out of vocabulary words in class-based and stem language models.

Number of Out of Vocabulary Words	*Number of Sentences*
1	48
2	16
3	6

In conclusion, decreasing the out of vocabulary rate is very important for accurate speech recognition. Moreover, when clustering done with lower-order occurrences, class-based models do not perform well with higher-order language models.

CHAPTER 6

CONCLUSION

This thesis focuses on Turkish continuous speech recognition. Statistical speech recognition approach is selected as the methodology. Statistical approach involves the construction of acoustic and language models. Once these models are constructed, recognition can be performed by conducting a search with the orientation of the models. Each of these items must be analyzed with great care, since they play an important role on the system's performance.

Hidden Markov models (HMMs) are used for modeling acoustic characteristics. the HMMs uses the feature vectors that are supplied by the front-end. Thus, the parameters of the front-end determines the robustness of the system. This thesis investigates several front-end parameters to build robust models . The acoustic models, on the other hand, faces two conflicting problems, trainability and consistency. To find an optimum point, several values are searched on the clustering thresholds with the mixture counts of observation densities.

Language modeling is done using text corpus. A procedure for preparing large amount of data for language modeling is presented. This procedure is successful to correct and prepare a small portion like one fourth of the data, however, when large texts are considered this amount is very satisfying. It also minimizes the manual modification of the data, thus it saves a lot of time that could be lost during the modifications.

The agglutinative nature of Turkish prevents the speech recognition system from reaching to high performance with using classical n-gram language modeling approach. The high growth rates of vocabulary, bring the data sparseness problem. To overcome this problem, three methods are investigated. The class-based language modeling had not been applied to Turkish speech recognition before. We have showed that they perform better than stem-ending language models, which are mostly used for modeling agglutinative languages. However, class-based models cannot perform well when they are used with higher order language models, since they are clustered by using lower order n-grams. In addition to that, vocabulary coverage is stressed. The increase in the out of vocabulary rate decreases the performance of the system, since the words that are not in the dictionary cannot be recognized. This will affect the whole utterance like a chain reaction, since there is no boundary between the words in the continuous speech.

The training of both the acoustic models and the language models are performed using the Hidden Markov Toolkit (HTK). HTK is a complete tool such that, one can build a complete speech recognition system without using any other tools. However, HTK is poor on decoding. Static search space with linear lexicons are used, this brings memory problems with large vocabularies. Furthermore, because of the static search space, HTK can only use bigram models in the first pass of the decoding. Bigram language modeling limit decreases the charm of HTK for agglutinative languages.

In conclusion, class-based models can be used effectively in the first pass of the recognition. However, when using higher order language models they would not be effective. The out of vocabulary words effects the recognition performance very much. Thus, a language modeling strategy should try to increase the coverage of the vocabulary. HTK is very practical for building statistical models, however, it is poor on decoding.

6.1 Future Work

Considering the above conclusions a decoder that uses dynamic search strategy will be implemented as the next step. To overcome the vocabulary coverage problem, a weighted finite state machine will be used for modeling the inflectional structure and morphemes. Integrating this with the decoding process, it would be possible to increase the coverage of vocabulary. Using this approach all possible inflections would be available for each stem that is in the dictionary. In addition to that, syntax should be integrated in the language modeling using structured language models this will definitely decreases the grammatical errors. However, for that purpose, syntactically annotated text should be built as the first step. The integration of morphological modeling with the syntax should boost the performance significantly.

REFERENCES

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

[2] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA: The MIT Press, 1997.

[3] S. Young, "A review of large-vocabulary continuous-speech recognition," *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45–57, 1996.

[4] O. Çilingir, "Large vocabulary speech recognition for Turkish," Master's thesis, Middle East Technical University, Turkey, 2003.

[5] K. Çarkı, P. Geutner, and T. Schultz, "Turkish lvcsr: Towards better speech recognition for agglutinative languages," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 1563–1566, June 2000.

[6] C. Yılmaz, "A large vocabulary speech recogntion system for Turkish," Master's thesis, Bilkent University, Turkey, 1999.

[7] M. A. Çömez, "Large vocabulary continuous speech recognition for Turkish using HTK," Master's thesis, Middle East Technical University, Turkey, 2003.

[8] S. Şahin, "Language modeling for Turkish continuous speech recognition," Master's thesis, Middle East Technical University, Turkey, 2003.

[9] T. Çiloğlu, M. Çömez, and S. Şahin, "Takılı bir dil olarak Türkçe için dil modelleme," in *IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı*, April 2004.

[10] O. Büyük, H. Erdoğan, and K. Oflazer, "Konuşma tanımada karma dil birimleri kullanımı ve dil kısıtlarının gerçeklenmesi," in *IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı*, May 2005.

[11] K. Oflazer, E. Göçmen, and C. Bozşahin, *An outline of Turkish morphology*. Report on Turkish Natural Language Processing Initiative Project, 1994.

[12] H.-W. H. Xuedong Huang, Alex Acero, *Spoken Language Procesing: A Guide to Theory, Algorithm, and System Development*. Englewood Cliffs, NJ: Prentice Hall, 2001.

[13] T. F. Quatieri, *Discrete-Time Speech Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2002.

[14] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.

[15] L. R. Rabiner and B. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Magazine*, vol. 3, pp. 4–16, January 1986.

[16] R. P. Lippmann, E. A. Martin, and D. B. Paul, “Multi-style training for robust isolated-word speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Dallas, USA).

[17] L. R. Rabiner, J. G. Wilpon, and F. K. Soong, “High performance connected digit recognition using hidden markov models,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1214–1225, August 1989.

[18] K. F. Lee, “Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 599–609, April 1990.

[19] L. Bahl, R. Bakis, P. Cohen, A. Cole, F. Jelinek, B. Lewis, and R. Mercer, “Further results on the recognition of a continuously read natural corpus,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Denver, USA), pp. 872–875, 1980.

[20] R. Schwartz, Y. Chow, S. Roucos, M. Krasner, and J. Makhoul, “Improved hidden Markov modeling of phonemes for continuous speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (San Diego, USA), pp. 21–24, 1984.

[21] S. Young, “Statistical modelling in continuous speech recognition (CSR),” in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, (Seattle, USA), pp. 562–571, 2001.

[22] S. Young, “The general use of tying in phoneme-based HMM speech recognizers,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (San Francisco, USA).

[23] S. Young, J. Odell, and P. Woodland, “Tree-based state tying for high accuracy acoustic modelling,” in *Proceedings ARPA Workshop on Human Language Technology*, pp. 307–312, 1994.

[24] R. Rosenfeld, “Two decades of statistical language modeling: Where do we go from here?,” *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[25] S. M. Katz, “Estimation of probabilities from sparse data for the language model component of a speech recognizer,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, pp. 400–401, March 1987.

[26] E. Charniak, *Statistical Language Learning*. Cambridge, MA: The MIT Press, 1993.

[27] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan, “Using a stochastic context-free grammar as a language model for speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 189–192, May 1995.

[28] A. L. Berger, S. D. Pietra, and V. J. D. Pietra, “A maximum entropy approach to natural language processing,” *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[29] C. Chelba and F. Jelinek, "Recognition performance of a structured language model," in *Proceedings of Eurospeech, 1999*, pp. 1567–1570, 1999.

[30] C. Chelba and F. Jelinek, "Exploiting syntactic structure for language modeling," in *procedings of the COLING-ACL*, pp. 225–231, 1998.

[31] V. Siivola, T. Hirsimaki, M. Creutz, and M. Kurimo, "Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner," in *Proceedings of Eurospeech 2003*, pp. 2293–2296, September 2003.

[32] V. Siivola, M. Kurimo, and K. Lagus, "Large vocabulary statistical language modeling for continuous speech recognition in finnish," in *Proceedings of the 7th European Conference on Speech Communication and Technology*, 2001.

[33] W. Byrne, j. Hajic, P. Ircing, F. Jelinek, S. Khudanpur, P. Krbec, and J. Psutka, "On large vocabulary continuous speech recognition of highly inflectional language - Czech," in *Proceedings of Eurospeech 2001*, pp. 487–490, September 2001.

[34] M. Szarvas and S. Furui, "Finite-state transducer based modeling of morphosyntax with applications to hungarian lvcsr," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 368–371, 2003.

[35] H. Ney and S. Ortmanns, "Progress in dynamic programming search for lvcsr," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1224–1240, 2000.

[36] F. Weng, A. Stolcke, and A. Sankar, "Efficient lattice representation and generation," in *Proceedings of ICSLP*, pp. 2531–2534, 1998.

[37] Ö. Salor, B. Pellom, T. Çiloğlu, K. Hacıoğlu, and M. Demirekler, "On developing new text and audio corpora and speech recognition tools for the Turkish language," in *Proceedings of 7th International Conference on Spoken Language Processing*, pp. 349–352, September 2002.

[38] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2.1)*. Cambridge University Engineering Department, 2002.

[39] SAMPA, Computer Readable Phonetic Alphabet, http://www.phon.ucl.ac.uk/home/sampa/home.htm.

[40] K. Oflazer, "Two-level description of Turkish morphology," *Literary and Linguistic Computing*, vol. 9, no. 2, 1994.