LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION FOR TURKISH USING HTK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT ALİ ÇÖMEZ

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2003

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. Mübeccel DEMİREKLER
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.

Examining Committee Members

Prof. Dr. Mübeccel DEMIREKLER _____

Assoc. Prof. Engin TUNCER _____

Assoc. Prof. Buyurman BAYKAL _____

Assoc. Prof. Tolga ÇİLOĞLU _____

Assist. Prof. H. Gőkhan İLK _____

ABSTRACT

LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITON FOR TURKISH USING HTK

ÇÖMEZ, Murat Ali

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Tolga ÇİLOĞLU

JUNE 2003, 100 pages

This study aims to build a new language model that can be used in a Turkish large vocabulary continuous speech recognition system. Turkish is a very productive language in terms of word forms because of its agglutinative nature. For such languages like Turkish, the vocabulary size is far from being acceptable. From only one simple stem, thousands of new word forms can be generated using inflectional or derivational suffixes. In this thesis, words are parsed into their stems and endings. One ending includes the suffixes attached to the associated root. Then the search network based on bigrams is constructed. Bigrams are obtained either using stem and endings, or using only stems. The language model proposed is based on bigrams obtained using only stems. All work is done in HTK (Hidden Markov Model Toolkit) environment, except parsing and network transforming.

Besides of offering a new language model for Turkish, this study involves a comprehensive work about speech recognition inspecting into concepts in the state of the art speech recognition systems. To acquire good command of these concepts and processes in speech recognition isolated word, connected word and continuous speech recognition tasks are performed. The experimental results associated with these tasks are also given.

Keywords: Speech recognition, large vocabulary, continuous speech, language model, bigrams, stem, ending, parsing, Turkish morphology.

ÖZ

HTK İLE TÜRKÇE İÇİN GENİŞ DAĞARCIKLI AKAN KONUŞMA TANIMA

ÇÖMEZ, Murat Ali

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Tolga ÇİLOĞLU

HAZİRAN 2003, 100 sayfa

Bu çalışmada, Türkçe için geniş dağarcıklı bir akan konuşma tanıma sisteminde kullanılacak bir dil modeli geliştirilmesi amaçlanmıştır. Türkçe, eklemeli bir dil olarak, sözcük biçimleri açısından çok üretken bir dildir. Bu tür diller için, dağarcık boyutu kabul edilebilir olmaktan bir hayli uzaktır. Yalnızca basit bir kökten, yapım ve çekim eklerini kullanarak binlerce yeni biçimli sözcük türetilebilir. Bu tezde, sözcükler kök ve eklerine ayrılmışlardır. Daha sonra sözcük ikililerine dayalı ağ yapısı oluşturulmuştur. Sözcük ikililerine ait olasılıklar ya kök ve ekler üzerinden, ya da yalnızca kökler üzerinden elde edilmişlerdir. Önerilen dil modeli ise yalnızca kökler kullanılarak elde edilen sözcük ikililerine ait olasılıklara dayanmaktadır. Ek-kök ayrıştırma ve ağ dönüştürme işlemleri dışında tüm çalışma HTK (Hidden Markov Model Toolkit) ile gerçekleştirilmiştir.

Türkçe için yeni bir dil modeli geliştirilmesinin yanısıra bu tezde, günümüz konuşma tanıma sistemlerine özgü kavramlara değinen kapsamlı bir çalışma

yapılmıştır. Bu kavramlara ve konuşma tanıma tekniği içerisindeki süreçlere hakimiyetin sağlanması amacıyla ayrık kelime tanıma, ardışık kelime tanıma ve akan konuşma tanıma deneyleri gerçekleştirilmiştir. Bu deneylere ait sonuçlar ise ayrıca verilmiştir.

Anahtar Kelimeler: Konuşma tanıma, geniş dağarcık, akan konuşma, dil modeli, sözcük ikilisi, kök, ek, ayrıştırma, Türkçe biçimbilim.

ACKNOWLEDGEMENTS

Precedingly, I would like to express my honest gratitude to my supervisor Assoc. Prof. Tolga Çiloğlu for his guidance, leadership and support, either technical or mental. He has been a good igniter when I fell off in performance. My colleagues; Dinç Acar, to whom I ran immediately when I got problems with HTK, and Serkan Şahin, who dedicated his labour to preparing the training data are to be appreciated, additionally.

And the people behind the camera; they really deserve my gratitude with their encouragements in every aspect of our daily life; they are Öznur Önder, Emre Cebecioğlu and his family, Sıdıka Kuzucu and her children, Kadriye Çömez, İsmail Çalışkan, Savaş Cengiz, Ahmet Güven, Teoman Tepehan and Özgür Erken.

To

My mother,

who gave me my language as the best gift that I ever got

And

To

Oktay Sinanoğlu,

who reminded me of the importance of that gift.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

LIST OF ABBREVIATIONS

HMM : Hidden Markov Model
HTK : The Hidden Markov Model Toolkit of Cambridge University
ASR : Automatic Speech Recognition
IWR : Isolated Word Recognition
CWR : Connected Word Recogniton
CSR : Continuous Speech Recognition
LM : Language Model
LVCSR : Large Vocabulary Continuous Speech Recognition
DTC : Decision Tree Clustering
WER : Word Error Rate
CSRR : Correct Sentence Recognition Rate
CWRR : Correct Word Recognition Rate
Ac : Accuracy
DP : Dynamic Programming
DTW : Dynamic Time Warping
MFCC : Mel-Frequency Cepstrum Coefficients
LPC : Linear Prediction Coefficients
gen : genitive
acc : accusative

CHAPTER 1

INTRODUCTION

Speech is the primary communication medium between people. This communication process has a complex structure consisting not only of the transmission of voice. Gestures, the language, the subject and the capability of the listener contribute to this process. As the maxim says, what you can tell is restricted to what the auditor can understand. In this respect, the performance of a speech recognizer system heavily depends on how and for which task you designed it.

Speech recognition area of science has its roots in the idea of communicating with a machine by voice. Our ability to communicate with machines and computers through keyboards or other devices is slower and more cumbersome. Speech can be regarded as an important component in order to make this communication easier.

Actually, a computer or a machine is not expected to understand what is uttered. But it is expected to be controlled via speech or to transcript the acoustic signal to symbols. The ultimate goal of research on automatic speech recognition (ASR) is to build machines that are indistinguishable from humans in the ability to communicate in natural spoken language. In this sense, speech recognition is not a mature science but an emerging one.

The earliest attempts to build ASR systems were made in the 1950's, when researchers tried to exploit the fundamental ideas of acoustic-phonetics. In 1952, a system for isolated digit recognition for a single speaker was built [11].

The early ASR devices applied threshold logic to voltage analogs of acoustic patterns to recognize words or short utterances of a certain speaker. It was

understood then that no simple generalization of this technique would be sufficient to recognize the fluent speech. Indeed, it was well understood that the intelligence borne by the speech signal was encoded in a highly complex way [1].

In the 1960's, three key research projects were initiated. In the first one, a set of elementary time-normalization methods was developed, which was based on detection of speech starts and ends. Thus, the variability of the recognition scores was reduced. In the second project in the Soviet Union, Vintsyuk proposed the use of dynamic programming (DP) methods for time aligning a pair of speech utterances. The third key project was Reddy's research in the field of continuous speech recognition (CSR) by dynamic tracking of phonemes [11].

In the late 1960's, a recognition task of spoken chess moves was experimented at Stanford University. The field was in its infancy then. The task was based on a small vocabulary and a restricted syntax. Additionally, the actual set of hypotheses was rather small. As an example, a recognition hypothesis corresponding to a move of a piece to a square occupied by another piece of the same color could be rejected. The system was adjusted to the speaker.

In early ASR designs, a recognizer would segment the speech into successive phones, i.e. basic pronunciation units, then identify the individual phones corresponding to the segments. In the last step, the system would transcribe the recognized phone strings sequentially [2].

By the early 1970's, largely as a result of development in electronics and information sciences, the prospectus had changed. The computer had evolved into a powerful and flexible processor. Furthermore, a linguistic theory of speech had emerged. According to this theory, all spoken language was viewed as a composition of a relatively small number of primary symbols. These symbols were related to measurable acoustic events [1].

There are broadly three classes of speech recognition applications. In isolated word recognition systems each word is spoken with pauses before and after it, so that end-pointing techniques can be used to identify word boundaries reliably. Second, command-and-control applications with constraints use small vocabularies, limited to specific phrases, but use connected word or continuous speech. This type of recognition task can include phone calling, ticket reservation or opening pages when

surfing on internet. Finally, large vocabulary continuous speech recognition (LVCSR) systems have vocabularies of several tens of thousands of words. In LVCSR, sentences can be arbitrarily long and spoken in a fluent, natural way.

In the late of 1970's, the research tended from isolated word recognition problem to connected word recognition problem. With their famous paper in 1979, Sakoe and Chiba [3] offered a two-level DP algorithm. In the same year, researchers at AT&T Bell Labs began a series of experiments aimed at making speech recognition systems that were truly speaker independent [11].

Sakoe's algorithm was rather complex and inconvenient for real-time applications. The level building dynamic time warping (DTW) algorithm of Rabiner and Myers in 1981 [4] was related to this algorithm. It was more flexible and efficient, but more complex. Ney proposed a clarified version of the algorithm proposed by Vintsyuk [5]. Ney's algorithm was the simplest one in these algorithms offering a solution to the connected word recognition problem.

All these algorithms were based on template matching technique, i.e. on DTW. The property of ASR research in the 1980's was the tendency in technology to statistical modeling methods; especially the hidden Markov model approach. Although the basic theory of Markov chains has been known to mathematicians and engineers for close to 80 years up to then, it had been introduced to speech processing in the middle 1970's and became popular in 1980's. Refinements in the theory and implementation of Markov modeling techniques have greatly improved the ASR applications [6]. Rabiner *et al.*constructed a connected digit recognizer in 1989 based on hidden Markov models (HMM) [7].

The HMM technique has a common acceptance by the researchers to be the state of the art in ASR systems. HMM is agreed to be the most promising one. It is presented as a generalization of its predecessor technology, DP. It might be used successfully with other techniques to improve the performance, such as hybridizing the HMM with artificial neural networks. Because of that this thesis relies on HMM's, other techniques are not discussed in the detail Chapter 2.

Another new technology that was reintroduced in the late 1980's was the idea of neural networks. Neural networks were first introduced in the 1950's, but they did not prove useful then, because they had many practical problems. Finally, in the

1980's the researchers of the Defense Advanced Research Projects Agency (DARPA) community made great contributions to LVCSR [11]. DARPA is an abbreviation that speech researchers may come across frequently.

The HMM's correspond to the acoustic part of a speech recognizer. Actually, speech recognition is achieved with the analysis of two information sources. These are the speech signal itself (acoustic part) and the language (or grammar) used. A semantic level can be added to these, too. Humans use all these three levels when speaking. The combination of these levels in human is hard to resolve and is subject to different sciences such as physiology, linguistics, psychology etc. That is why speech recognition can be regarded as an interdisciplinary field.

In 1990's, the complexity of tasks had grown from the 1000-word DARPA resource management task to essentially unlimited vocabulary tasks such as transcription of radio news broadcast in 1995. Though, the word recognition accuracy has remained impressive in comparison with the increase in task complexity. On the other hand, the resource requirements had grown as well [12].

Due to the requirements of the large vocabulary and need for speed in real time applications, the research affords have been dedicated to memory reduction and efficient search algorithms. Terms such as look-ahead pruning techniques, triphones, lexical tree search, long-distance N-gram language modeling, across-word model search, phonetic fast match and multiple-pass decoding are subject to be applied in state-of-art ASR systems [13-26]. Additionally, the trend goes to implementation of multilingual ASR systems, which means that the system will respond to different languages.

Today, speech recognition is not a laboratory event. Personal computers and hand-phones can be operated with voice. In CEBIT 2003, a washing machine that has a multilingual vocabulary of four thousand words was demonstrated [27]. Speech recognition feature is especially making life easier for blind people.

In recent years, the study of systems that combine the audio and visual features emerged as an attractive solution to speech recognition. A number of techniques have been presented to address the audio-visual integration problem. In an audio-visual feature system, the observation vectors are obtained by the concatenation of the

audio and visual observation vectors. The resulting observation sequences are then modeled using one HMM [33].

On the other hand, the studies in Turkish speech recognition are few [8-10,28-31]. This is due to the difficulties of the structure of the Turkish language. Turkish is an agglutinative language and is highly productive in terms of word generation. For instance, from a verbal stem one can generate thousands of distinct words by concatenating various suffixes, each of them including different meanings. In addition, syntax rules in Turkish are not well defined, i.e. a sentence will not be regarded as wrong, if the positions of words are changed. That is why it is hard and not straightforward to implement a Turkish LVCSR system based on the studies made for languages like English. The studies on Turkish speech recognition are focused on developing a language model that will fit the characteristics of Turkish.

Taking into account the term "*continuous*", there is not a system developed for Turkish, which processes naturally spoken utterances. This thesis attempts to build a LVCSR system for continuously spoken Turkish. The language model proposed is based on bigrams. However, the bigram probabilities are obtained in two different ways: First, only from stems; second, from stems and endings.

1.1 Outline of the Thesis

In Chapter 2, background information about speech recognition generally based on statistical methods will be given. Language modeling and acoustic modeling techniques are discussed in detail. Advanced readers can skip to the Chapter 3.

Chapter 3 takes the search problem under scope.

Chapter 4 demonstrates the use of the toolkit HTK in short. Some practical aspects that the author faced during this study are also given.

Chapter 5 includes a briefing about Turkish morphotactics and the work that is actually done in this thesis. The language model used is the core of the thesis. Statistics of the training corpus and the experimental results are given in tables.

Chapter 6 concludes this thesis and describes ideas for future research.

CHAPTER 2

SPEECH RECOGNITION

Speech recognition systems assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols. The basic goal is to "*decode*" this message and then convert it either into writing (e.g. dictation machine) or into commands to be processed (e.g. hands free dialing).

There are three approaches to speech recognition, such as [11];

1. The acoustic-phonetic approach
2. The pattern recognition approach
3. The artificial intelligence approach

The acoustic-phonetic approach is based on the theory of acoustic phonetics. The theory proposes that there exist finite, distinct phonetic units in spoken language. The phonetic units are characterized by a set of properties that are embedded in the speech signal or its spectrum.

In pattern-recognition approach to ASR, the speech patterns are used directly without explicit feature determination and segmentation. The method has two steps: Training of speech patterns and recognition of patterns via pattern comparison. Speech knowledge is supplied into the system via the training procedure. Current ASR systems are based on the principles of statistical pattern recognition. The basic methods of applying these principles to the problem of speech recognition were proposed by Baker and Jelinek in 1970's [35].

The artificial intelligence approach is a compound approach that utilizes the ideas of the first two approaches. The intention here is to mechanize the recognition

procedure like the way a person applies his intelligence in analyzing and making decision on the acoustic knowledge. The aim is to integrate phonemic, lexical, semantic and pragmatic knowledge together [11].

The use of neural networks is regarded as a separate structural approach that can be used in each of the above approaches [11]. This leads to a hybrid structure of ASR systems.

There are two main problems in a speech recognition system. The first one is the modeling problem. The underlying question is "*How should be the speech signal represented in order to simulate the production and perception of the speech by human*?". The answer is strictly related to the acoustic modeling. In acoustic modeling, the features of the speech are extracted in terms of vectors and then the observation probabilities of these vectors are computed.

This probability is represented by

$$P(\mathbf{o}_t \backslash w_j) \qquad (2.1)$$

$\mathbf{o}_t$: vector observed at time t

w_j: *j*'th vocabulary word

The second problem is search or decoding problem. The underlying question is "*How will the system find the right word or words uttered among all words in the vocabulary in an efficient way?*". To find the most likely word or word sequence, the ASR system searches a network of words. The size and the shape of the search space are mainly determined by the language model. Language model involves the utterance probability of words and depends on the subject spoken about. Language model probability is represented by

$$P(w_j)$$

In fact, an ASR system tries to maximize the probability that the uttered word is w_j given the observation vector sequence **O**,

$$\underset{j}{\text{argmax}}[P(w_j \backslash O)]$$

$\mathbf{O} = o_1\ o_2\ o_3\ \ldots\ o_{t-1}\ o_T$

This probability is not computable directly but using Bayes' rule gives

$$P(w_j \setminus O) = \frac{P(O \setminus w_j)P(w_j)}{P(O)} \qquad (2.2)$$

For an isolated word recognition (IWR) task, it is sufficient to compute (2.1). For a LVCSR system, the formula 2.2 takes the form;

$$\mathrm{P(W \setminus O)} = \frac{P(O \setminus W)P(W)}{P(O)}$$

$$\mathrm{W= w_1\ w_2\ w_3\ \ldots\ldots\ w_{Q-1}\ w_Q\ \ Q \le N}$$

Where N is the number of words in the vocabulary. The probability P(W) includes the syntactic and semantic conditions together. If only the syntactic conditions are held, the language model becomes a grammar. This type of language models are generally represented in the form of finite state networks.

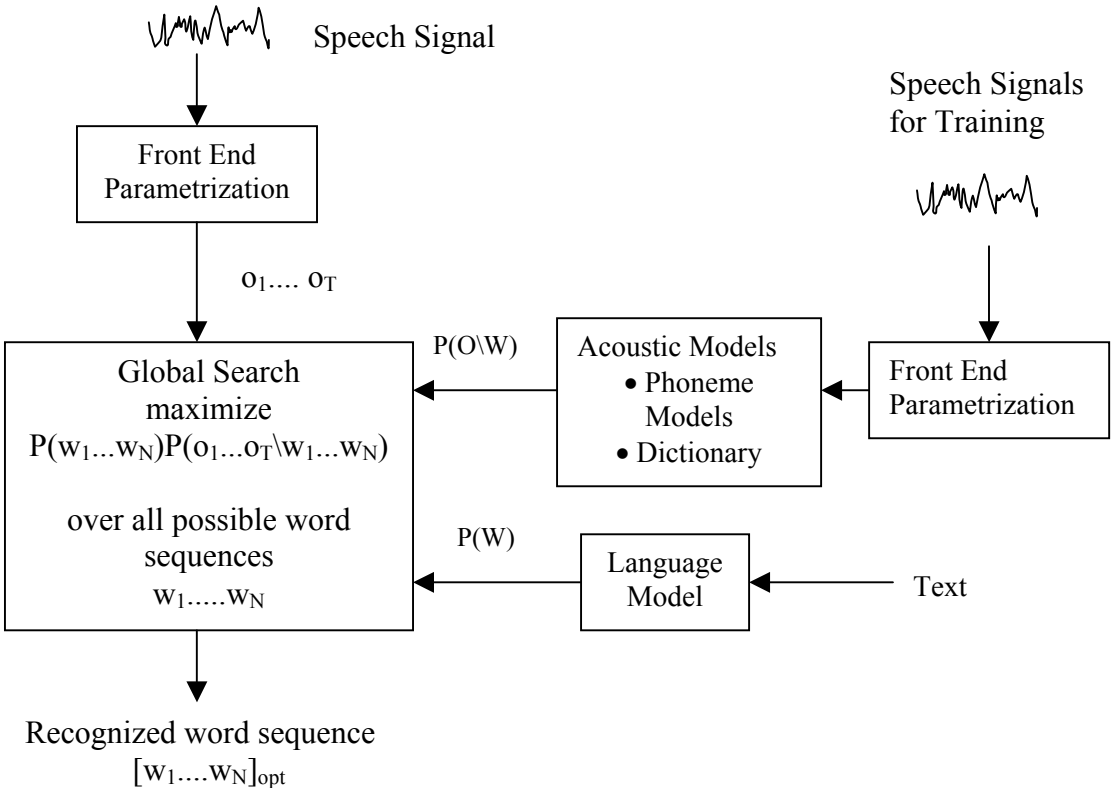


Fig. 2.1. Overview of a statistical LVCSR system

An ASR system is built up of two main parts: Front end and back end. In front end part, the feature vectors of the speech signal are extracted. In the back end, the actual recognition takes place. According to the system design and techniques used,

these parts consist of various components. A general demonstration of the structure of a statistical LVCSR system can be seen in Figure 2.1 [13].

The diagram in Figure 2.1 shows the computation of the probability P(W**O**). The prior probability P(W) is determined directly from a language model and can be obtained from a text. The likelihood of the acoustic data P(O\W) is computed using a composite HMM representing W constructed from simple HMM phone models that are joined in sequence according to word pronunciations stored in a dictionary.

The next section is about the acoustic model, namely obtaining the feature vectors and obtaining the observation probability P(O\W).

2.1 Acoustic Model

The core of an acoustic model lies in the capabilities of the feature vectors to capture the distinctive properties of the speech. A good acoustic model should take into account speaker variations, pronunciation variations, environmental variations and context dependent phonetic coarticulation variations. For this reason, the acoustic training corpus has to be quite large to obtain a robust acoustic model.

To realize the design in Figure 2.1 requires the solution of the acoustic modeling at first. Initially, a front end parameterization is needed which can extract from the speech waveform all of the necessary acoustic information to construct HMM's. The HMM's must accurately represent the distributions of each sound in every context that may occur. Furthermore, HMM parameters must be estimated from speech data that might never be sufficient to cover all possible contexts.

In this chapter we focus on the solutions that work well in practice and are used in state of the art systems.

2.1.1 Front End Processing

Speech signal is a non-stationary signal in its nature and its characteristics may change in very short time instances. In order to capture the variability in the waveform, every state of the art system firstly segments the signal into frames. At a given, very short time frame, e.g. 25 ms, the speech segment is close to stationary. In this interval, speech signal waves remain unchanged. These frames are taken at distances less than the frame length, for instance at every 10 ms, so that every frame overlap. The features are extracted from these frames. The feature can be the linear prediction coefficients (LPC), mel-frequency cepstrum coefficients (MFCC), LP based cepstra or spectrum coefficients of this frame. A block diagram of a front end processor can be seen in Figure 2.2.

Today's systems still cannot match human's performance. In spite of construction of a very accurate speech recognizer for a particular speaker, in a particular language and speaking style, in a particular environment and limited to a particular task, it remains as a problem to build a system that can understand anyone's speech, in any language, on any topic, in any fluent style and in any environment.

The frequency range of human voice does not include informative components at frequencies higher than 5 kHz, so it is reasonable to reduce the amount of high frequency noise by low pass filtering. Moreover, the acoustic transfer function of human ear balances the spectrum before perception.

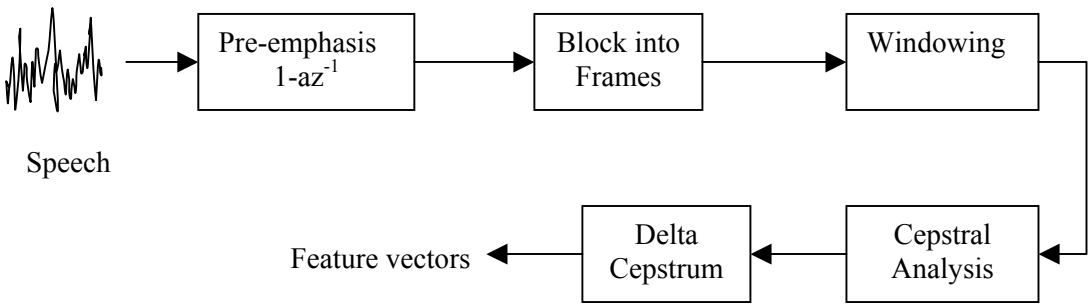


Fig. 2.2. Block diagram of a front end processor

The frames have to be multiplied by a window function (windowing), so that the transitions from frame to frame can be smoothed. Windowing is also beneficial to recover all parts of the signal and to eliminate possible gaps between frames. Without windowing, the spectral envelope has sharp peaks and the harmonic nature of a vowel is not apparent. The window function used in this thesis is Hamming window. Its formula is given by;

$$\omega(n) = 0.54 - 0.46\cos(2\pi \frac{n}{N-1})$$

where N is the length of the frame in terms of samples.

In the next step, the Fourier transformed signal is passed through a set of band pass filters that have triangular shape. The bandwidths (critical band) and center frequencies are determined by experimental results on human hearing. The Mel-frequency scale is designed to approximate the frequency resolution of the human ear. The frequency resolution of the human ear is linear up to 1000 Hz and logarithmic thereafter. This means that the band pass filters get larger at higher frequencies.

The log filter outputs are then used to obtain the cepstral coefficients with a discrete cosine transform formula;

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

where N is the number of filterbank channels and m_j is the output of the j'th filter. This has the effect of compressing the spectral information into lower order coefficients and it also de-correlates them. De-correlation allows the statistical modeling to use diagonal covariance matrices [35].

The acoustic model assumes that each feature vector is uncorrelated with its neighbors. This is a rather poor assumption, because the physical structure of the human vocal apparatus ensures that there is continuity between successive spectral estimates [17]. To capture the changes in the signal, a feature vector used to describe the signal for one frame can additionally contain features of neighbor frames or the difference between features of the predecessor and successor frames. Furthermore, the difference of the differences can be added to the feature vector, too. So, a feature

vector consisting of 12 cepstral and an energy coefficient can be of length 39, by concatenating the delta- and acceleration cepstral and energy coefficients.

2.1.2 Hidden Markov Model

A hidden Markov model is not used only in acoustic modeling. It has a wide application area in statistical signal processing. The basic component of a Markov model is the *state*. These states correspond to the positions of the vocal apparatus of the human when speaking. As understood from the name, these states are hidden and the underlying task in an ASR is to decode the actual state sequence that causes the speech signal be observed. Then, the word (words) or phoneme (phonemes) to which these states belong is determined via the help of the language model. Before discussing the HMM and its role in speech recognition, its elements have to be known. The discussion in this section is based on [38] and on the well known papers of Rabiner [6,37].

1. There are a finite number of states in the model, N. Within a state the signal has some measurable and distinctive properties.
2. At each time instance, t, a new state is entered based on a transition probability distribution that depends on the previous state.
3. After each transition is made, an observation symbol is output according to a probability distribution which is related to the current state. This probability distribution is fixed for each state.
4. The initial state distribution that determines the state in which the system can be at the start up.

The definitions used in formulas are as follows:

$\mathbf{S}=\{S_1, S_2 \ldots . S_N\}$, the individual states

$\mathbf{q_t}$: the state entered at time t

N: the number of states

$\mathbf{a_{ij}}$= P[q_{t+1} =S_j \ q_t = S_i], the transition probability that the state entered at time t+1 is S_j given that the state, q_t , at time t was S_i . $1 \leq i,j \leq N$

A=$\{a_{ij}\}$, the state transition probability distribution. **A** is called as transition probability matrix

$\mathbf{o_t}$**:** observation vector at time t

$\mathbf{b_j(o_t)}$= P[o_t \ q_t = S_j], probability of observing $\mathbf{o_t}$ given that the system is in state S_j at time t. $1 \leq j \leq N$

$\boldsymbol{\pi_i}$= P[q_1 = S_i], probability of being in state i at time 1.

$\boldsymbol{\pi}$ = { π_i }, the initial state distribution. $1 \leq i \leq N$

$\boldsymbol{\lambda}$ = (A, B, π), the compact notation to indicate the complete parameter set of HMM.

These concepts can be seen on Figure 2.3. The topology shown in Figure 2.3 is not the only choice to be built. It can be modified according to the application. The HMM topologies used in this thesis can be seen in Figure 4.3, Figure 4.4 and Figure 5.1.

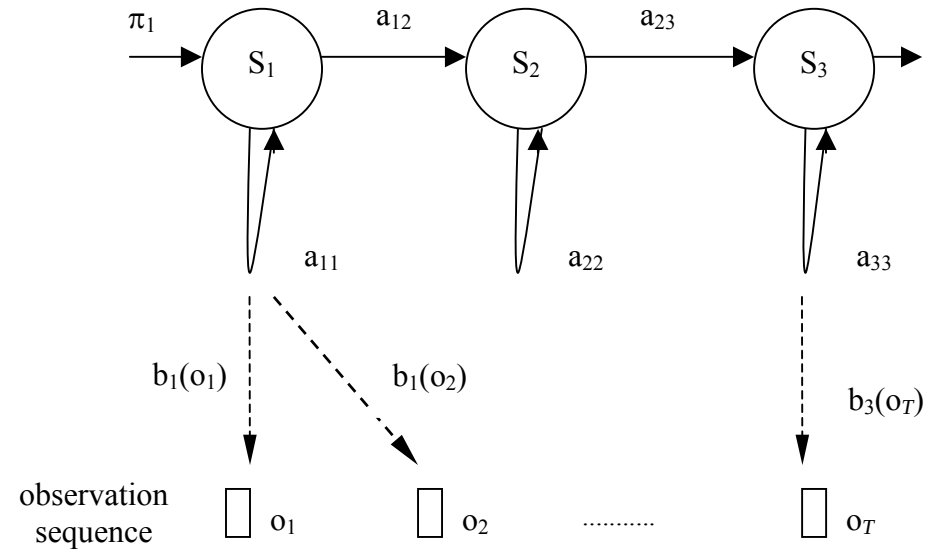


Fig. 2.3. A hidden Markov model topology.

There exist three problems when constructing HMM's. These are;

Evaluation Problem: How do we compute the probability of the observation sequence given the model, P (O \ λ)? The solution to this problem enables us to evaluate the probability of different HMM's generating the same observation sequence. If we have a different HMM for each word, then the recognized word is the one whose model has the largest probability of generating the data.

Decoding Problem: Given an output sequence **O** and model λ, how do we compute the most probable state sequence **Q**? This problem is concerned with uncovering the hidden state sequence from knowledge of the symbols output by the system.

Training Problem: How do we adjust the model parameters A, B, and π to maximize the likelihood of the model λ producing the output sequence? For the solution of this problem, training data is needed.

2.1.2.1 Forward-Backward Algorithm

The total number of possible paths conducting to the last state of the trellis in Figure 2.4. increases exponentially with the increasing number of states and observation instances. Reduction in the computational cost can be achieved by the forward-backward procedure. Actually it is a compound procedure composed of forward and backward procedures. In the evaluation case we need only one of them. The backward procedure is used in the solution of training, i.e. in Baum-Welch algorithm.

Initially define a new forward probability variable α_t (i), at instant t and state i;

$$\alpha_t (i) = P(o_1 o_2 \ldots..o_t , q_t = S_i \backslash \lambda)$$

This probability function could be solved for N states and T observations iteratively;

1. Initialization

$$\alpha_1(i) = \pi_I b_i(o_1) \qquad 1 \leq i \leq N$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \qquad 1 \le t \le T\text{-}1 \ , \ 1 \le j \le N$$

3. Termination

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

Termination stage is just a sum of all the values of the probability function α_t (i) over all states at instant T. The result represents how likely the given model produces the given observations.

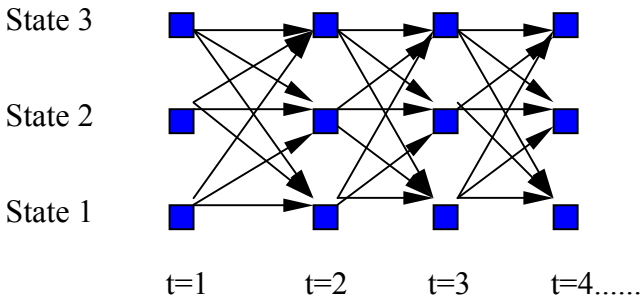


Fig. 2.4. The flow of the process of a HMM in time.

The backward procedure is similar to the forward procedure. However, the state flow in computation is backward from instant T to instant 1. Let us define a backward probability function β_t(i);

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots\ldots o_T \setminus q_t = S_i, \lambda)$$

1. Initialization

$$\beta_T(i)=1 \qquad 1 \le i \le N$$

These initial values for β's of all states at instant T can be arbitrarily selected.

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad t = T\text{-}1, T\text{-}2, \ldots 1 \ ; \ 1 \le i \le N$$

The probability P(O\λ) can be computed from both forward and backward functions. This is illustrated in Figure2.5. The forward probability is a joint probability whereas the backward probability is a conditional probability. The probability of state occupation is determined by taking the product of the two probabilities.

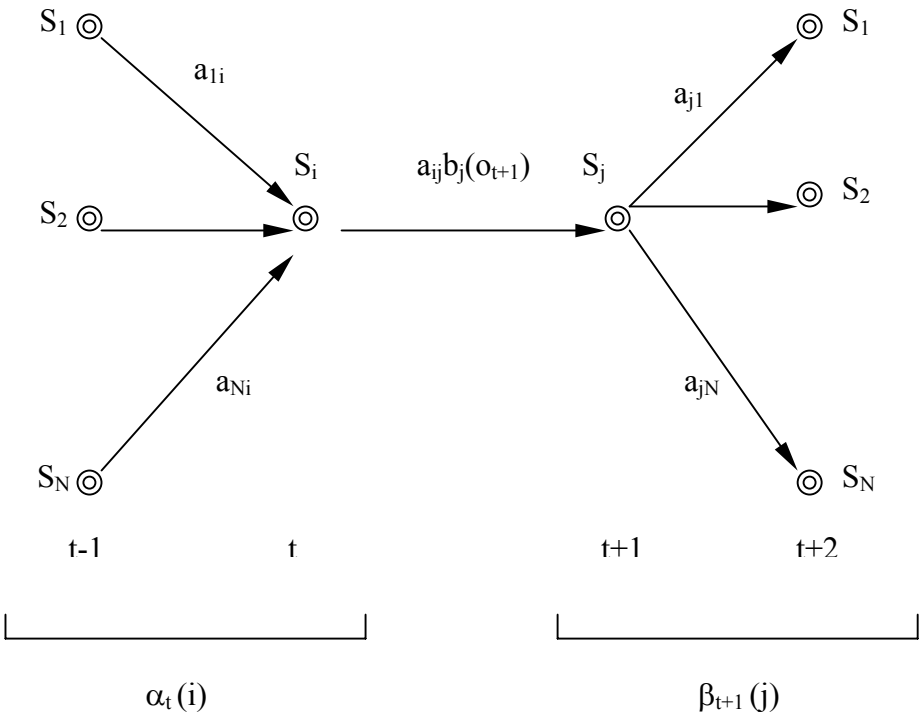


Fig. 2.5. Forward-backward probability functions to obtain P(O\|λ)

2.1.2.2 Viterbi Algorithm

In solution of the decoding problem, the popular Viterbi algorithm is used. The criterion of optimality here is to search for a single best state sequence through modified DP technique. Viterbi algorithm is a parallel search algorithm, namely it searches for the best state sequence by processing all the states in parallel. We need to maximize P(Q\O,λ) to detect the best state sequence. Let us define a probability

quantity $\delta_t(i)$ which represents the maximum probability along the best probable state sequence path of a given observation sequence after t instants and being in sate i;

$$\delta_t(i) = \max_{q_1, q_2 \cdots q_{t-1}} P[q_1 \ldots . q_{t-1}, q_t = S_i, o_1 \ldots . o_t \setminus \lambda]$$

The best state sequence is backtracked by another function $\psi_t(j)$. This function holds the index of the state at time t-1, from which the best transition is made to the current state. Complete algorithm is given as follows:

1. Initializing

$$\delta_1(i) = \pi_i \, b_i(o_i) \qquad 1 \le i \le N$$

$$\psi_1(i) = 0$$

2. Recursion

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

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

3. Termination

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

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

4. Backtracking

$$q_t^* = \psi_{t+1}[q^*_{t+1}] \quad T\text{-}1 \ge t \ge 1$$

It is clear that Viterbi recursion is similar to forward induction, except the interchange of summation by maximization. So, it is obvious that $P(O\setminus\lambda)$ can be computed approximately with Viterbi algorithm, taking P^* as the score. This is illustrated in Figure 2.6.

In HTK, a modified version of Viterbi algorithm is used, which is called '*token passing algorithm*'. It is assumed that every state of HMM at time t holds a single moveable token that represents the partial match path that goes from the observation vector o_1 to o_t throughout the search space. Each token contains a path identifier and a partial alignment cost. The path identifier is a pointer to a record named *word link record* (WLR) that includes word boundary information. Then the recursion formula

of $\delta_t(j)$ is replaced by the *token passing algorithm*. The algorithm is equally applicable to DTW and HMM recognition. It brings implementational advantages.

The key steps in this algorithm are as follows [34]:

1. Pass a copy of every token in state i to all connecting states j, incrementing the probability of the copy by $a_{ij}b_j(o_t)$.
2. Examine the tokens in every state and discard all but the token with the highest probability.

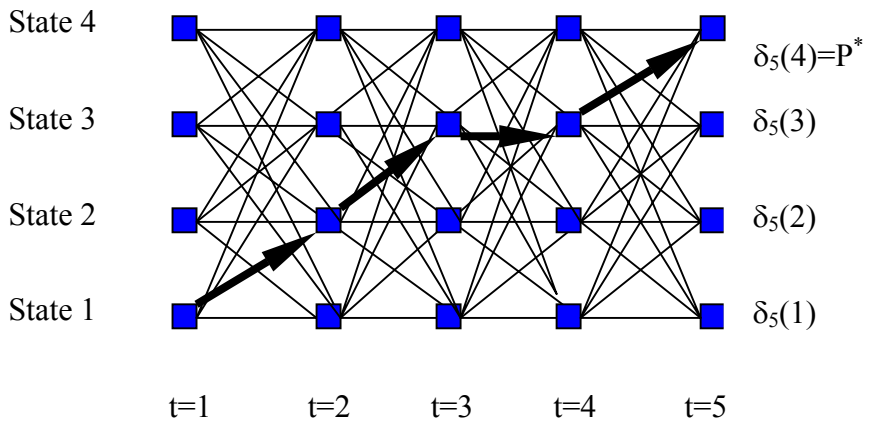


Fig. 2.6. Flow of the Viterbi algorithm. The best path is given in bold.

The exact algorithm for CWR case is given in Section 3.1.1.

The number of tokens in a state can be more than one if N-best search (See Section 3.2.4) is accomplished. However, if a pruning threshold (See Section 2.3) is applied, not all of the tokens having worse scores than the one of the best token are discarded.

2.1.2.3 Baum-Welch Algorithm

This algorithm is related to the training problem that is the most difficult one. The aim is to adjust parameters of the model according to an optimality criterion.

Baum-Welch algorithm is strictly related to forward-backward algorithm and it tries to reach the local maxima of the probability function P(O\λ). The model always converges but the global maximization is not guaranteed. That is why the initial point of search is very important.

Let us define the probability of being in state S_i at time t, and state S_j at time t+1;

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

Its relation with forward and backward variables is;

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

The numerator term equals to $P(q_t = S_i , q_{t+1} = S_j , O\backslash\lambda)$ and the denominator term equals to P(O\λ).

Now define the posteriori probability of being in state S_i at time t, given the observation sequence and the model;

$$\gamma_t(i) = P(q_t = S_i \backslash O,\lambda)$$

Its relation with forward and backward variables is;

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

If we sum $\gamma_t(i)$ over the index t, we get a quantity which can be interpreted as the expected number of times that state S_i is visited and the summation of $\xi_t(i,j)$ over t can be interpreted as the expected number of transitions made from S_i to S_j.

With the help of these, the formulas for re-estimating the parameters A, B and π are given as;

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

$$\bar{a}_{ij} = \frac{\text{xpected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from } S_i}$$

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

expected number of times in state j and observing symbol v_k

expected number of times in state j

$$= \frac{\sum_{\substack{t=1 \\ O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

After the re-estimation of the model parameters we will have a new model which is more likely to produce the observation sequence O. The iterative re-estimation procedure continues until no improvement in $P(O\backslash\lambda)$ is achieved.

It should be noted here that HMM's have some limitations. HMM's assume;

1. The transition probability depends only on the origin and destination.
2. All observation frames are dependent only on the state that generated them, not on the neighboring observation frames.
3. There is no information about the state duration.

After a model with certain parameters is built, some refinement has to be done, i.e. fine tuning is needed. The proposed solution can be incrementing the number of mixtures in the probability distribution $b_j(o_t)$, or clustering states according to some criteria or tying the parameters of different models. Section 2.1.3 discusses decision tree clustering (DTC) and state tying.

2.1.2.4 Embedded Training

Embedded training uses Baum-Welch procedure, in which all models are trained in parallel. It is used especially in continuous speech recognition, where the particular HMM's are generally sub-word models. In addition, there is no need to have boundary information to invoke embedded training. Its outline can be given as follows [34].

1. Load the set of HMM definitions used in the system.
2. Allocate zero accumulators for all parameters of all HMM's.
3. Load the next utterance.
4. Construct the composite HMM from the HMM's in the HMM set by concatenating them as it is shown in Figure 2.8. The construction is made according to the transcription of the corresponding utterance.
5. Compute the forward and backward probabilities for the composite HMM.
6. Use the forward and backward probabilities to compute the probabilities of state occupation at each frame of the utterance and update the accumulators.
7. Repeat from Step 3 until all the training utterances are processed.
8. Use the accumulators to compute new parameter estimates for all of the particular HMM's.

The steps above can be repeated until a convergence criterion is met. Embedded training updates all of the HMM's using all of the training data.

2.1.3 Model Refinement

One has to build more accurate models, whereas he is obliged to obtain the parameters from sparse training data. Accuracy of the model can mean using more parameters per model or increasing the number of subword models (See Section 2.1.4). One way to increase the accuracy without increasing the complexity of the

model is to build models that share parameters. This sharing is also called as *tying*. Parameter tying can be done in different levels in Gaussian mixture based ASR systems. Means, covariances, mixture components, states or transition matrices can be tied.

State tying cannot be done arbitrarily. It can be achieved via data driven clustering or decision tree clustering. In both cases, the training data is used to determine which states should be tied. In data driven clustering, the most similar states are tied. Similarity measure is the distance between these states. In DTC, optimum decision tree is found and the states remaining in the same leaf of this tree are tied. In this thesis, DTC is used. That is why DTC will be given here.

2.1.3.1 Decision Tree Clustering

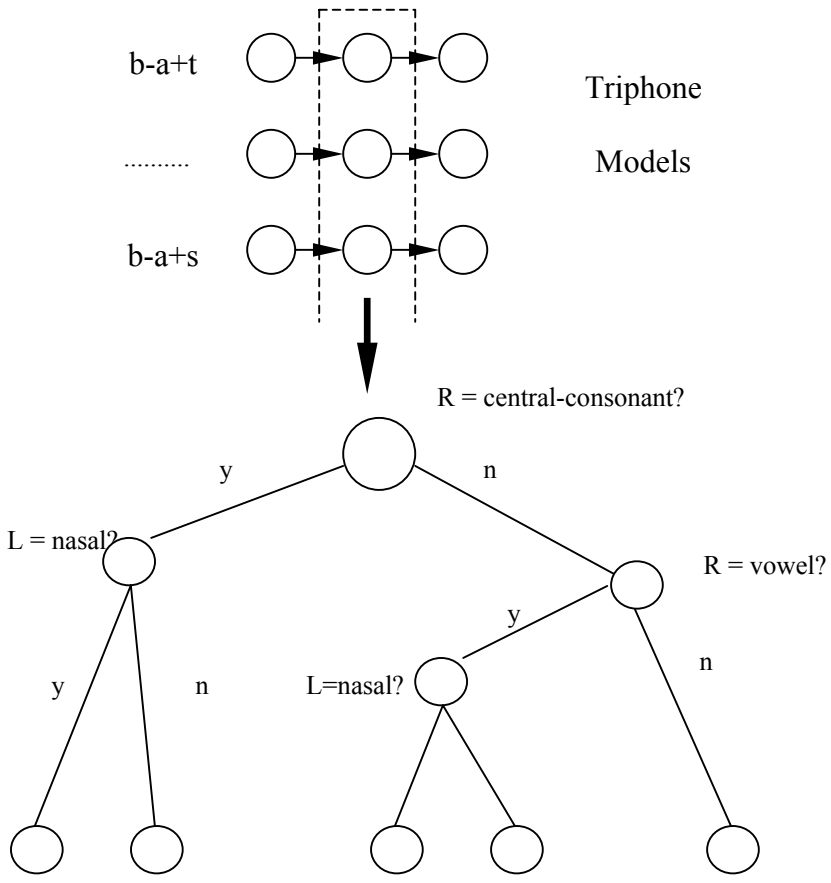


Fig.2.7. Clustering the center states of the phone ‘a’

One major property of decision trees is the capability to produce models that are not seen during acoustic training. Decision trees constructed are stored to use in the future. Referring to these trees, the unseen models can be constructed artificially.

Decision trees are built by asking a set of binary questions, i.e. the answer is yes or no, when clustering the states. These questions are asked at branching nodes of the tree. Initially, all states belonging to models to be clustered are placed in the root node of the tree. As the questions are being asked, the space of states is divided into two. The number of states in a half space depends on the answer 'yes' or 'no'.

Splitting the state space ends up when the optimality criterion is met. This criterion depends on two different thresholds defined by the user. The threshold can be the minimum number of states that must be in one cluster or the increase in the total likelihood of the training data on the current set of clusters. Then the states in the cluster are tied.

An example of decision tree built for triphone models can be seen in Figure 2.7 [35]. The questions used in this example are like; "*Is the right context a central-consonant?*", or "*Is the left context a nasal?*". These questions for a triphone based system can be denoted as;

QS L_nasal {*-a+*}

QS R_cons {*-a+*}

The questions used in thesis can be found on Appendix A. This question list is mainly taken from [32], but some modifications are made according to the phonemes used in this thesis. The large amount of questions is not harmful; anyway, clustering ends up not according to the number of questions, but according to the optimality criterion.

2.1.4 Recognition Units

Speech can be viewed as the sequence of the sounds that represent phonemes. Although an alphabet of a language has a certain number of letters corresponding to

phones, the amount of phonemes can exceed this number. This is due to the style of speaking, vocal apparatus of the speaker and the context in which the phoneme is uttered. For example, the phoneme '*r*' is differently uttered in the last position of the word or in the beginning of the word. Additionally, a speaker may not voice this phoneme exactly. For this reason, obtaining robust models for phonemes could be a hard work and so the recognition of them could be a hard task because of the acoustic confusion.

One way to overcome the problem, the models may correspond to a whole word. Because the acoustic representation of whole words is well defined and the acoustic variability occurs mainly at the beginning and at the end of the word. Another advantage of using whole-word speech models is that it prevents from constructing a pronunciation dictionary.

For IWR and CWR tasks this aim can be achieved, but in a LVCSR system increasing the amount of models consumes great memory and effort in search. Additionally, to obtain reliable whole word models, the number of word utterances in the training set needs to be sufficiently large. This means that each word in the vocabulary should appear in every possible context several times in the training set. Collecting such an amount of training data and training the models is not practical.

Another problem with using whole-word models in a LVCSR system is that the phonetic content of the individual words will overlap. So, storing and comparing whole word patterns will be redundant.

The sub-word units are employed in state of the art systems. Sub-word units can be monophones units, syllables, demisyllables, biphones or triphones. Monophones cannot model the context, whereas others do.

Triphones are very popular in use. A triphone is a subword unit model modeling the effect of the phones on left and right of the actual phone. As an example, the representation "e – l + a" belongs actually to the phone 'l' but it contains the effect of 'e' and 'a', too. The frames falling in the region of the phoneme 'l' with context "e – l + a" are used in training separately from frames that fall into region of the phoneme 'l' with different contexts, unless their parameters are tied.

Word models are built up appending the triphone models according to a pronunciation dictionary. The dictionary can contain triphone or monophone

expansions of words (for details see Section 4.2.2). Constructing the whole model for the word ‘kale’ from triphone models can be seen in Figure 2.8.

Fig. 2.8. Connecting triphone HMM’s to build a composite HMM for the word ‘kale’. The sign ‘*’ corresponds to any context.

2.2 Language Modeling

A large vocabulary speech recognition system is dependent on the linguistic knowledge, because the knowledge contributed by acoustic model, P(O\W), becomes insufficient in a huge space of different models. Therefore, it is a must to make use of the language. The probability P(W) has a great effect on search process. It provides effectiveness and efficiency in meantime (See Chapter 3). Since it does not depend on the acoustic data, only a text corpus is needed to compute.

Language models are used to model regularities in natural language. The most popular methods are statistical n-gram models, which attempts to capture the syntactic and semantic constraints by estimating the probability of a word in a sentence given the preceding (n-1) words. Hence, this probability strictly depends on the amount of the text corpus and its subject. For example, probability of occurrence of word ‘santrafor’ in a text corpus taken from an art journal is almost ‘0’ as expected, whereas in a corpus taken from sports pages of a newspaper it may be much higher.

In this chapter, the role of the language model based on statistical concepts is discussed.

2.2.1 N-grams

The priori probability of the word sequence W = w_1 …….w_N can be computed as;

$$P(W) = P(w_1)P(w_2 \setminus w_1)P(w_3 \setminus w_1\ w_2)\text{…….}P(w_N \setminus w_1\ w_2\ \text{……}w_{N-1})$$

It is impossible to obtain this conditional word probability for all possible word sequences of different lengths. Therefore, N-gram models, especially bigrams and trigrams are used. The term can be approximated as;

$$P(W) \approx P(w_j \setminus w_{j-N+1\ \ldots\ldots} w_{j-1})$$

The value of N is a trade-off between the stability of the estimate and its appropriateness. A trigram (N=3) is a common choice with large training corpus whereas a bigram (N=2) is often used with smaller ones. As the amount of N gets larger, it gets more difficult to reliably estimate the priori probability. This probability can be estimated by relative frequency approach.

$$P(w_i \setminus w_{i-N+1\ldots\ldots\ldots} w_{i-1}) = \frac{F(w_{i-N+1}\ldots\ldots w_{i-1} w_i)}{F(w_{i-N+1}\ldots\ldots w_{i-1})}$$

where *F* is the number of occurrences of the string in its argument in the given training corpus. It is obvious that some many possible word sequences may not be observed in the training corpus. This means, a *zero* probability is appointed to the unseen N-grams. In addition, distribution function of the frequencies may have sharp edges, i.e. some of the words may occur lots of times whereas some of them may occur only a few times (See Chapter 5 for the statistics of the corpus used in this thesis). Instead of straightforward estimation from counts, various smoothing techniques have been proposed to balance the probabilities. These include discounting the estimates, recursively backing-off to lower N-grams and interpolating N-grams of different order [17].

Referring to what we said in the beginning of this chapter, that the size and the shape of the search space are mainly determined by the language model, an example of simple search space based on bigrams is illustrated in Figure 2.9 [2].

2.2.2 Back-Off N-Grams

One solution to the zero probability is known as the “back-off”. In back-off smoothing, the N-grams that are not observed or the number of occurrences of which are below a threshold during training are assigned a nonzero probability related to the unigram probabilities. Assigning all strings a nonzero probability helps prevent errors in speech recognition. This is the core issue of smoothing.

A back-off system has three main components: the back-off node that connects the unseen and less seen bigrams, a back-off weight and a unigram probability $P(w_j)$ for each word. Unigram probabilities have to be smoothed, too. Because unigram probability distributions may not be in balance as in the case of bigrams.

In this thesis, back-off bigram models are built using the given formulae below [34].

L: Number of distinct words

N(i,j): number of occurrences of the word pair (w_i , w_j)

N(i): number of occurrences of the word w_i.

For unigram probabilities, P(i);

$$P(i) = \begin{cases} N(i)/N & \text{if N(i)} > \text{u} \\ u/N & \text{otherwise} \end{cases}$$

where u is unigram floor count set by the user and $N = \sum_{i=1}^{L} \max[N(i), u]$. The back-off bigram probabilities are given as;

$$P(i,j) = \begin{cases} [N(i,j) - D]/N(i) & \text{if N(i, j)} > \text{t} \\ \alpha(\text{i})\text{P(j)} & \text{otherwise} \end{cases}$$

where D is a discount and t is a bigram count threshold that are set by the user. The back-off weight α(i) is computed to ensure that $\sum_{j=1}^{L} P(i,j) = 1$.

It is worth of note here that the back-off probability strictly depends on the quantity P(j). It performs a smoothing over the observed bigram probabilities. But we think that it would be better to associate a back-off weight α(i) that takes into

account the word pair (w_i , w_j); thus it could be denoted as $\alpha(i,j)$. For example, in a text corpus taken from sports pages, the word 'santrafor' (w_j) has a high probability $P(w_j)$. So, the probability P(santrafor\sarı)= α(sarı)P(santrafor) will be determined by way of back-off node. We suppose that this probability should be very close to '0'. However, because of that the probability P(santrafor) is high, the probability P(santrafor\sarı) will get high, too. That is why we think that the back-off weight of a particular word might be determined not only depending on the probability $P(w_j)$.

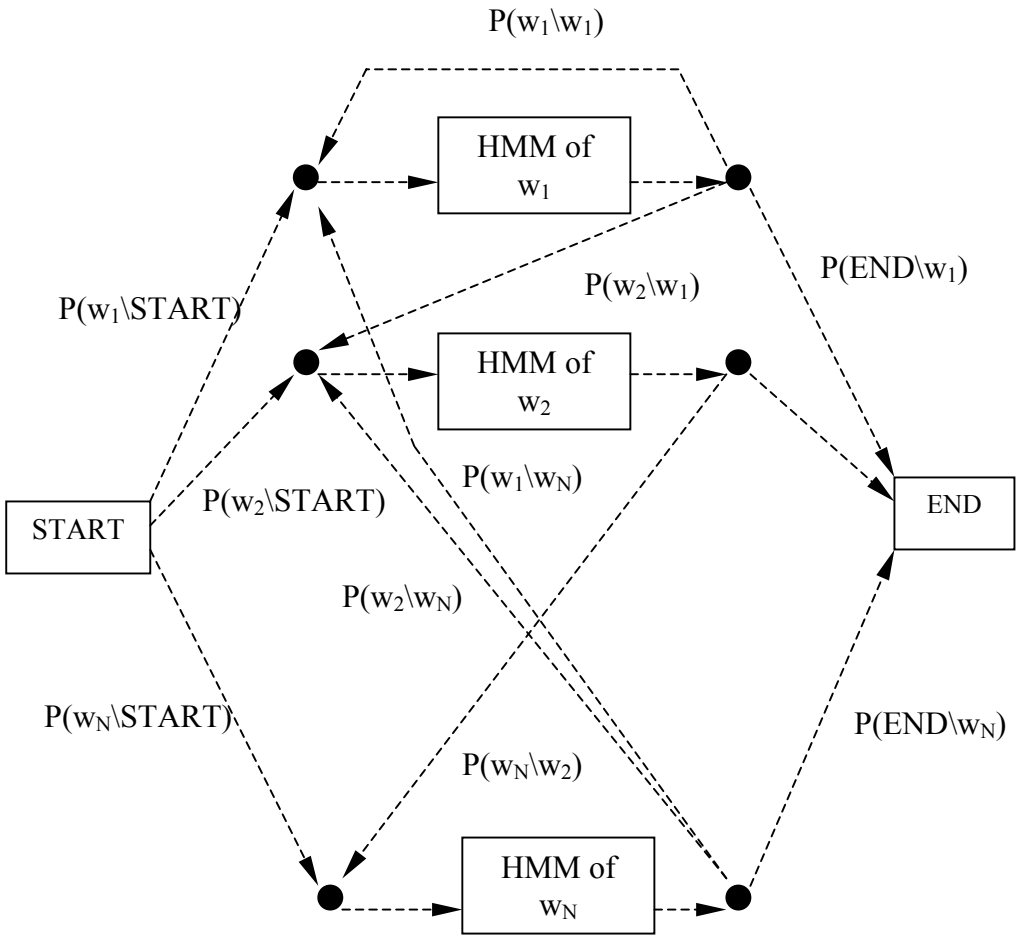


Fig.2.9. A simple search space based on bigrams. START and END boxes represent the start and end of the sentence (some transitions are not shown).

There are other back-off techniques proposed in the literature such as Kneser-Ney smoothing and Katz smoothing based on Good-Turing estimates. Kneser-Ney method slightly outperforms other smoothing techniques for both bigrams and

trigrams [17]. Whenever training data is sparse, smoothing can help performance and sparseness is almost always an important issue in statistical modeling.

In Figure2.10, a simple search space based on bigrams is illustrated. There is not a direct connection between w_k and w_j , because in training corpus the word pair (w_k , w_j) is not observed. Only observed bigrams are connected by direct transitions with correspondent bigram probabilities. This significantly reduces the bigram expansion [17].

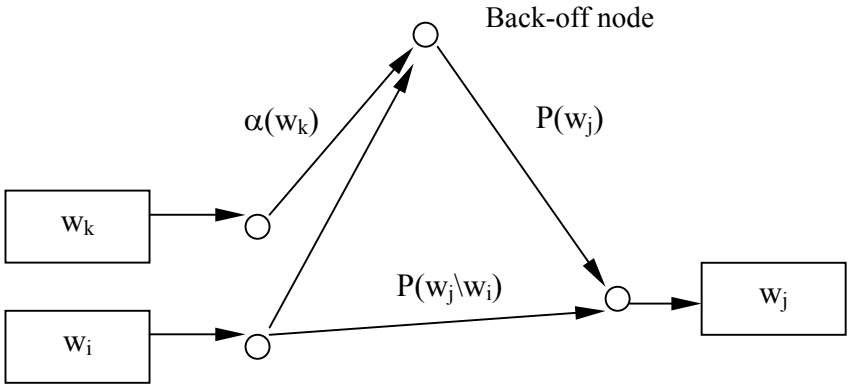


Fig. 2.10. Illustration of the back-off node for a bigram model.

2.2.3 Complexity

The performance of the language model built can be measured invoking the concepts of information theory such as entropy and perplexity. Assume that an information source generates word sequences chosen from a vocabulary according to a stochastic rule. The entropy of this source is given as [11];

$$H = -\lim_{Q \longrightarrow \infty} (\frac{1}{Q}) \{ \sum P(w_1 ..... w_Q) \log[P(w_1 ..... w_Q)] \}$$

The sum is over all string sequences w_1.....w_Q. This quantity can be considered as the average information of the source when it puts out a word w. The meaning of H is that a source with an entropy of H produces as much information as a source that puts out words equiprobably from a vocabulary of size 2^H. It can be viewed as the

degree of difficulty that the speech recognizer encounters, when it is to determine a word from the same source. If an N-gram language model is used, it can be estimated using the formula;

$$H = -\frac{1}{Q}\sum_{i=1}^{Q} \log[P(w_i \setminus w_{i-N+1}.....w_{i-1})]$$

Associated with entropy, perplexity can be defined as

$$B = 2^H$$

Perplexity can be viewed as the average word branching factor of the language model or as the average number of possible words following any string of (N-1). Perplexity is an important parameter that gives an idea about the perfomance of the language model.

2.2.4 Other Techniques

There are several problems with n-gram language models, for example;

1. They assume that all the contextual information is encoded in the previous n words.
2. It is difficult to automatically adapt them to a particular topic or task
3. They don't use any notion of word classes or categories.

Decision trees, linguistically motivated models, exponential models and adaptive models are other techniques of language modeling [26].

On the other hand, N-grams are not appropriate for an agglutinative language. For example, syntax rules in Turkish are not well defined, i.e. a sentence will not be regarded as wrong, if the positions of words are changed. Hence, N-gram probabilities obtained from a text will not be robust. But one way to apply them to an agglutinative language can be obtaining the N-gram probabilities over stems, i.e. the endings should not be taken into account. This will change the number of occurrences of the words and smoothing automatically takes place. The underlying idea is that the actual informative part of the word is the stem. Last to say; current

language models are very sensitive to changes in style, topic or genre of the text on which they are trained.

2.3 Performance

Results output by the ASR system have to be compared with expected correct results in order to measure system's performance. The question "Does the proposed method improve the recognition of the data?" is to be answered by performance measuring.

In this thesis, the correct word rate is measured with the formula;

$$Word\ Correct\ \text{Rate} = 100\frac{\text{N - D - S}}{\text{N}}$$

where N denotes the total number of words in recognized sentences, D denotes deletions and S denotes substitutions. This formula does not include insertions. Another measure is recognition accuracy, given as;

$$Accuracy = 100\frac{N - D - S - I}{N}$$

where I denotes insertions.

In the next chapter, the decoding process combining acoustic model and language model is discussed. The discussion deals with search methods based on static expansion of the search network.

CHAPTER 3

DECODING

Decoding means to search for the uttered word or word sequence among the words in the vocabulary. In this respect, the ASR system can also be called as *decoder*. Hence, in analogy with the terminology of finite-state methods for decoding in information theory, the search algorithm in speech recognition is often referred to as decoding algorithm. Problem is formulated as;

$$\overline{W} = \arg\max_{w_1, w_2 \ldots w_N} [P(O \setminus w_1, w_2 \ldots w_N) P(w_1, w_2 \ldots w_N)]$$

Search for the uttered word sequence is approximated by finding the best path that goes through the best state sequence that fits the feature vector sequence of the input speech signal (See Figure2.6). All knowledge sources at hand, such as the acoustic models, the language model and the constraints of the pronunciation dictionary are combined in this search. Then the search is called as an *integrated search*. As progressing from IWR to LVCSR going through CWR, decoding techniques have become more complex in order to be more effective and efficient.

Search in LVCSR systems is a challenging problem, because the word boundaries are not known. This means that each of the words in the vocabulary may start or end at each frame of the acoustic data. On the other hand, the search effort has to be reduced by limiting the search to a small part of the search space.

As mentioned earlier, decoding is much simpler in IWR and in systems with small and medium sized vocabulary. The one-pass time synchronous Viterbi search remains as a sufficient tool. The reference model that gives the largest $\delta_T(i)$ value is

put out as the recognized word. But in LVCSR, Viterbi algorithm cannot be used in a straightforward manner. The alternative to the frame synchronous Viterbi search is a time asynchronous search based on the A^* or *stack decoding* [17]. The algorithm is based on the ideas of heuristic search.

Decoding techniques in LVCSR have their roots in CWR techniques, which are based on DP techniques. In this section, several concepts in decoding techniques will be given.

3.1 Connected Word Models

Connected word models are small vocabulary tasks based on a task-oriented grammar that defines the syntax rules. This grammar constructs the search network. The recognizer puts out one of the paths in this network as the recognition result. The simplest network is the one that is not grammar based, namely each word can follow each other. A sample connected word model network can be seen in Fig. 3.1.

If a CWR system is based on subword models, every word is built up via the procedure shown in Fig. 2.8. Then the network becomes a HMM network. But the early connected word models were based on template matching using DP [3-5,40,41]. In template matching, one or more acoustic patterns (templates) for each word in the vocabulary are stored. The recognition process then includes matching the incoming speech with the stored templates. In IWR, the speech signal is matched with templates separately. The template with the lowest distance from the input is the recognized word. In CWR, matching process takes place in parallel, i.e. each template is compared with the speech in meantime and transitions between words are allowed.

There are three general approaches to CWR problem. These are addressed in [3], [4] and [41]. Because of the simplicity and most common use and closeness to the token passing algorithm, the one proposed by Ney [41] will be discussed here.

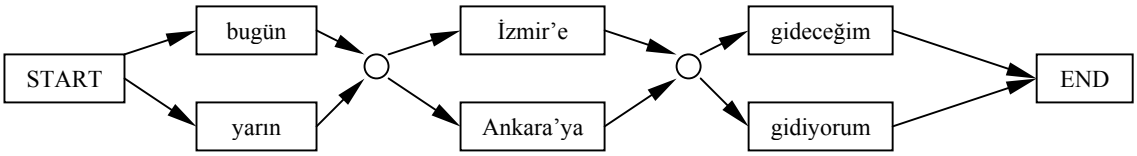


Fig. 3.1. An example of connected word model network. START and END denotes the start and end of the sentence.

3.1.1 One Stage Algorithm

In a system being either word or subword based, HMM's have a fixed length equal to the number of states in an individual model. However, the templates in a template based system have different lengths, namely the size of the search space depends on the number of templates and the total length of them. The search space for one stage algorithm can be seen in Figure 3.2. The accumulated distance for every point in search space is computed in parallel, i.e. simultaneously for every frame of input acoustic pattern as time goes on. The best path drawn bold in the Figure 3.2 is determined with backtracking. It means that this path cannot be determined until the search reaches the end.

Assume, the input speech signal consists of i = 1,….N feature vectors. This input pattern is assumed to be a concatenation of individual words. The words of the vocabulary correspond to a set of K reference patterns or templates obtained from single word utterances. The templates are distinguished by the index k = 1, …K. The frames of the template k are denoted as j = 1, …. J(k), where J(k) is the length of the k'th template. The aim is to determine the sequence of templates q(1)…..q(R) that best matches the input pattern.

The Y is given as a sequence of grid points;

$$Y = [\ y(1)...y(l).....y(L)\ ]$$

where y(l) = [i(l), j(l), k(l)] and l is the path parameter for indexing the ordered set of path elements.

The minimization problem is now;

$$\min_{Y} \sum_{l} d(y(l))$$

i.e. minimize the global accumulated distance with respect to all allowed paths. From the best path, the associated sequence of templates can be recovered as in Figure 3.2.

Referring to Figure 3.2, each grid point in the search space is defined by i'th frame of the input pattern and j'th frame of the k'th reference pattern, i.e. the grid point is shown as (i, j, k). the problem is to find the best path going thorough these grid points (i, j, k).

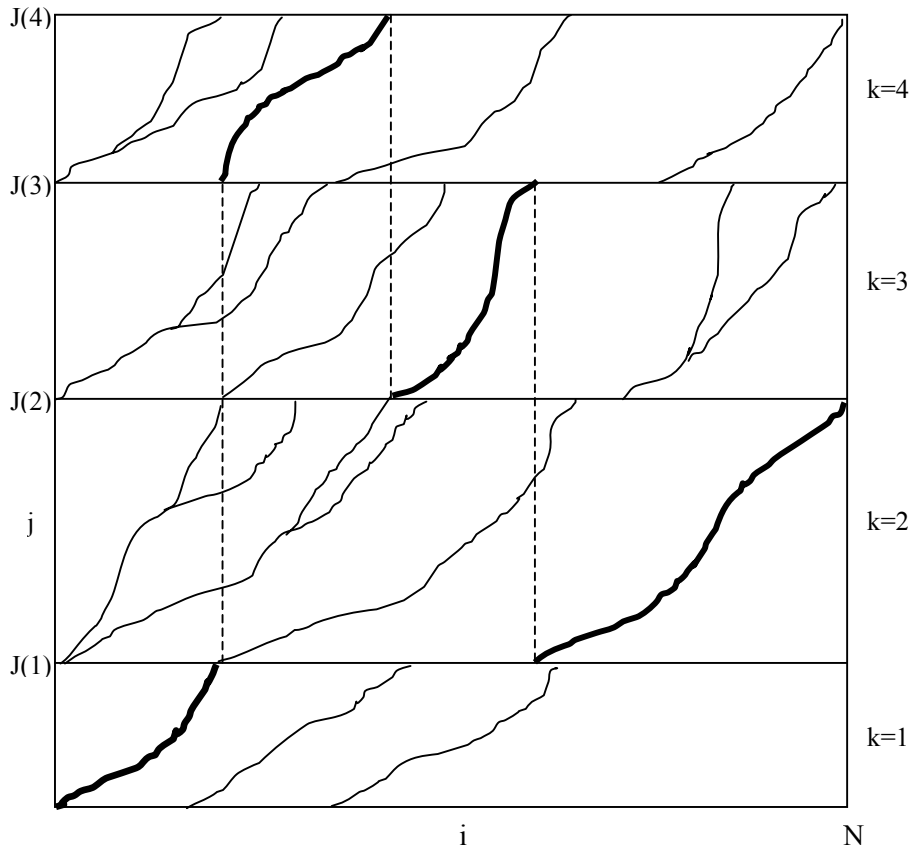


Fig. 3.2. CWR search space. Best path is drawn bold.

There two types of transition rules when going from one grid point to the other one:

1. Rules in the template interior called within-template rules

$$\text{If } y(l) = (i, j, k) \;\&\; j > 1 \text{ then}$$

$$y(l\text{-}1) \in \{ (i\text{-}1, j, k), (i\text{-}1, j\text{-}1, k), (i, j\text{-}1, k) \}$$

2. Rules at the template boundaries called between-template rules.

$$\text{If } y(l) = (i, 1, k) \text{ then}$$

$$y(l\text{-}1) \in \{ (i\text{-}1, 1, k), (i\text{-}1, J(k^*), k^*) : k^* = 1,\ldots,K \}$$

Since the grid point (i, 1, k) corresponds to the beginning frame of template k, it is necessary that it can be reached from the ending frame of any template k^* including k itself. These rules are illustrated in Figure 3.3.

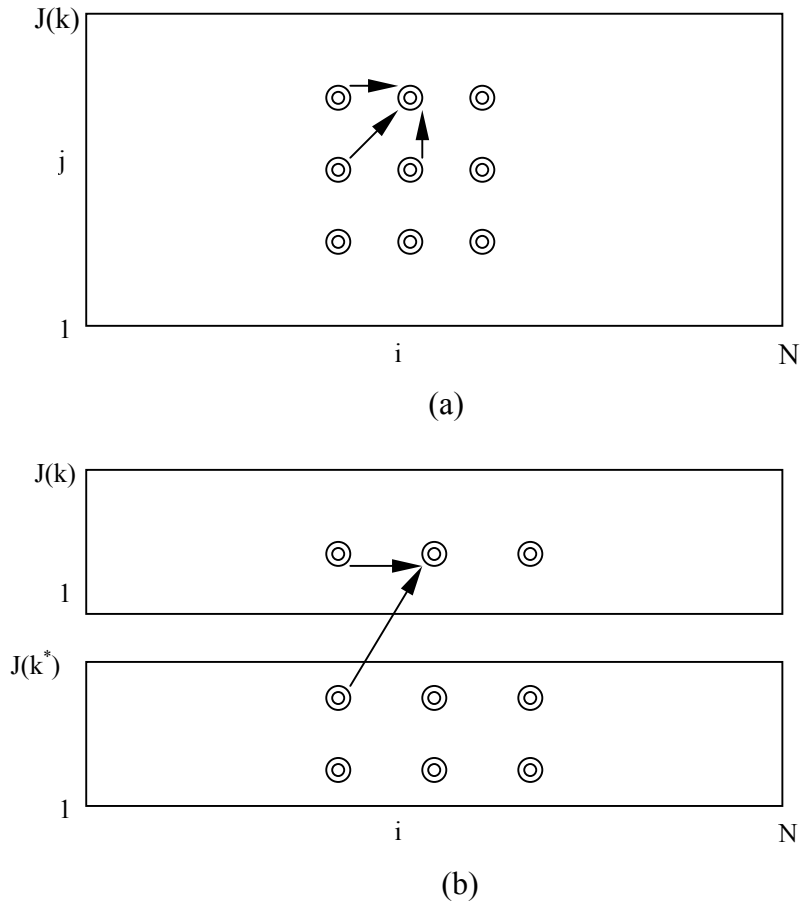


Fig. 3.3. (a) Within-template transition rules, (b) between-template transition rules

Let us now define optimality criterion elements;

d(i, j, k): local distance between i'th feature vector of the input speech and j'th feature vector of the k'th template

D(i, j, k): minimum accumulated distance score along any path to grid point (i, j, k)

Accumulated distance score is defined for within-template and between-template cases differently. For within-template case;

$$D(i, j, k) = d(i, j, k) + \min\{D(i-1, j, k), D(i-1, j-1, k), D(i, j, k)\}$$

and for between-template case;

$$D(i, 1, k) = d(i, 1, k) + \min\{ D(i-1, 1, k), D(i-1, J(k^*), k^*) : k^* = 1, \ldots., K\}$$

For grid points at the beginning frame i = 1 of the test pattern, the transition rules must be modified, since there is no preceding frame on the time axis of the test pattern. A grid point (1, j, k) can be reached only from a grid point (1, j-1, k).

Now, we can give the exact algorithm [41]:

1. Initializing

$$D(1, j, k) = \sum_{n=1}^{j} d(1, n, k) \quad k = 1, ...,K \quad j = 1,.....,J(k)$$

2. Recursion

For i = 2,…. ,N **do**

For k = 1,……, K **do**

$D(i, 1, k) = d(i, 1, k) + \min\{ D(i-1, 1, k), D(i-1, J(k^*), k^*) : k^* = 1, \ldots., K\}$

For j = 2,.....,J(k) **do**

$D(i, j, k) = d(i, j, k) + \min\{D(i-1, j, k), D(i-1, j-1, k), D(i, j, k)\}$

End

End

End

3. Termination

Trace back the best path from the grid point at a template ending frame with minimum total distance using the array D(i, j, k) of accumulated distances.

The uttered word sequence is recovered in third step. If HMM's are used, the templates are replaced by the HMM's and the search goes through the state space. The algorithm can be transported into token passing algorithm, too. The modified

algorithm is as follows [39]. The aspects of this algorithm are demonstrated in Figure 3.4.

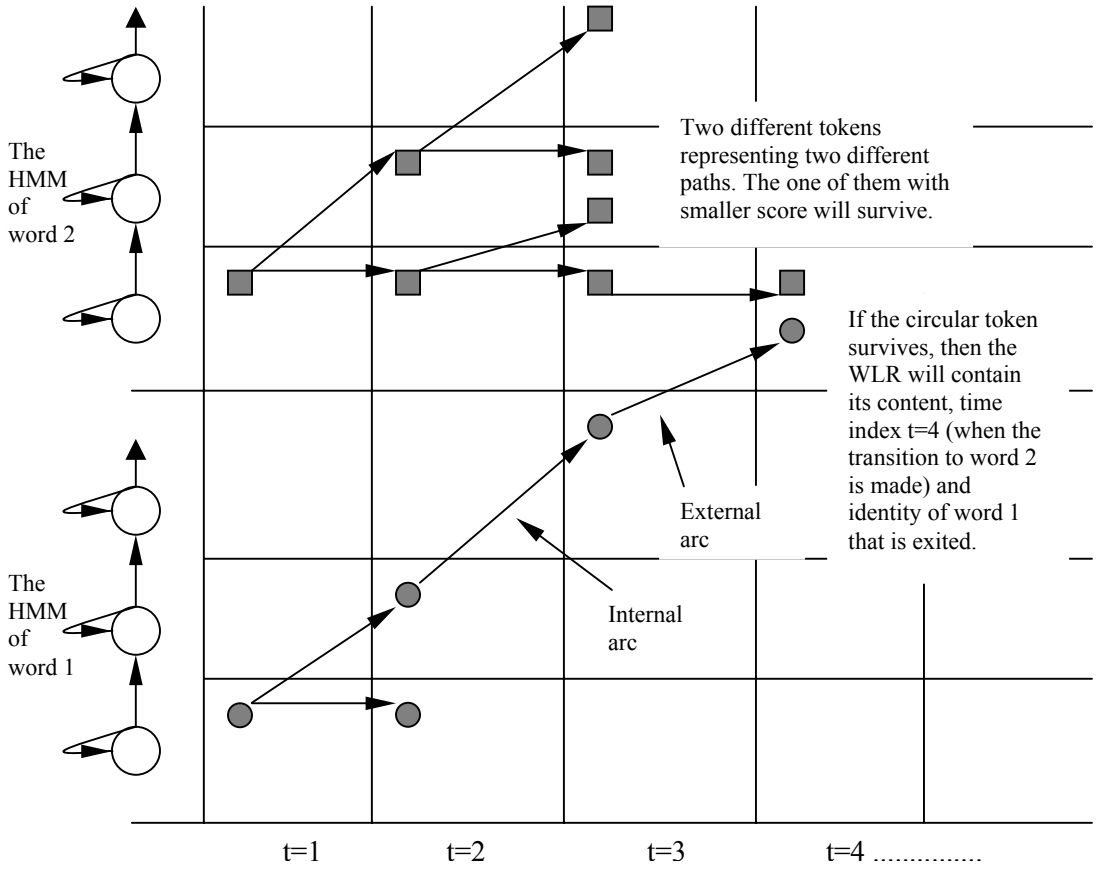


Fig. 3.4. The graphical concept of token passing algorithm for CWR.

Token Passing Algorithm

1. Initialization

   Each model initial state holds a token with value 0;

   All other states hold a token with value ∞

2. Recursion

   for t=1 to T **do**

   for each state i **do**

   Pass a copy of the token in state i to all connecting states j, incrementing its $\delta_t(j)$ value by $a_{ij}b_j(o_t)$;

   end

for each token propagated via an external arc at time t **do**
create a new *word link record* (WLR) containing
{token contents, t, identity of word exited}
end
discard the original tokens;
for each state i **do**
find the token in state i with the smallest value and discard the rest
end;
end;

3. Termination

Examine all final states, the token with the smallest value gives the required minimum matching score.

The uttered word sequence is recovered utilizing the *word link record* (WLR) that includes word boundary information.

Now, it is time to discuss LVCSR search issues.

3.2 LVCSR Decoding

Taking decisions in the presence of ambiguity and context is a great problem in every work on ASR. The ambiguity gets higher in a LVCSR system because of that the end of a word interferes with the start of another word. On the other hand, some of the phonemes are not pronounced exactly in continuous speech. If it were possible to recognize phonemes or words with high reliability, then the decision techniques, error correcting techniques and statistical methods would not be necessary. A LVCSR system has to deal with a large number of hyphotheses at every time instance of the process. To make this decision process reliable and quick for real time implementation, some techniques are developed. These techniques will be discussed in this section. The discussion is mainly about a system built up of triphone HMM models and N-gram language model.

3.2.1 Linear Lexicon vs. Tree Lexicon

In a subword based ASR system, a lexicon that shows how to construct word models from subword models is needed. The word models are constituted according to the pronunciation expansions written in the lexicon. If the lexicon is linear, i.e. each word is represented as a linear sequence of phonemes, the search space can be as the one shown in Figure 3.5. The search is called as *linear lexical search* then.

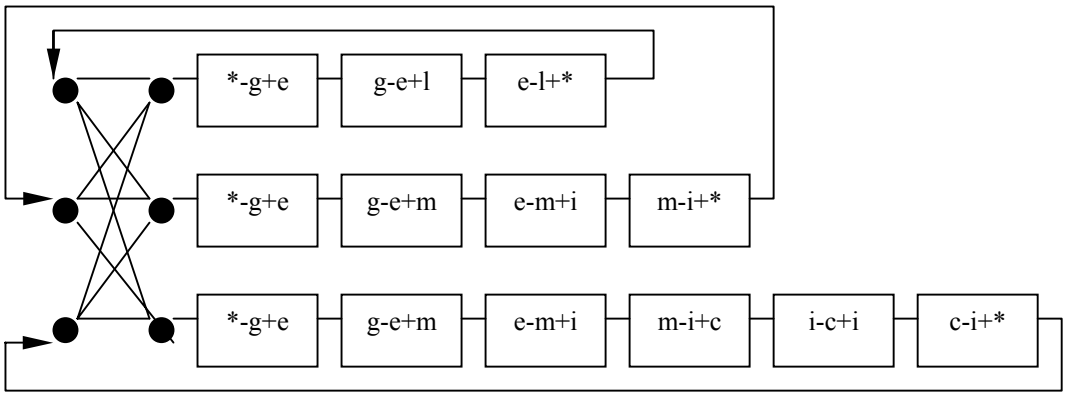


Fig. 3.5. A simple linear lexical search space based on triphones.

For a small vocabulary task, it is sufficient to have a separate representation of each word in terms of monophones or triphones. However, in a large vocabulary system, many words share the same beginning phonemes. For a large vocabulary task, it is useful to organize the pronunciation lexicon as a tree, since many phonemes can be shared to get rid redundant acoustic evaluations. The lexical tree based search is thus necessary for real time implementations. Reorganization of the lexicon in form of a tree lexicon saves time and storage. In Figure 3.6, a tree lexicon can be seen.

The linear structure shown in Figure 3.5 does not allow trigram language model to be used since the available word history is limited to 1 word, thus bigram language model is applicable. A linear lexical search space based on trigram language model without back-off node is illustrated in Figure 3.7 [2].

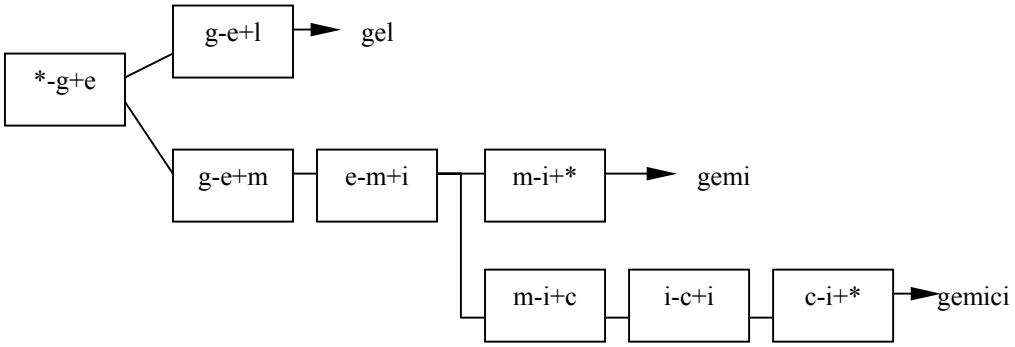


Fig. 3.6. Construction of word models using a tree lexicon.

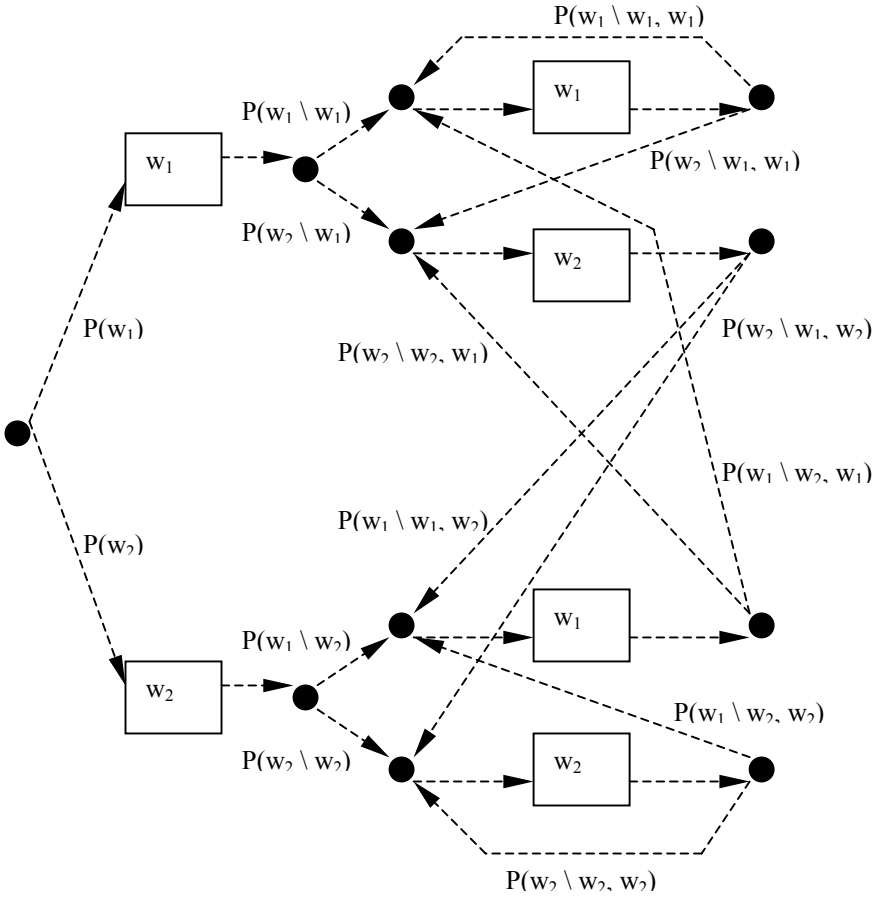


Fig. 3.7. A linear lexical search space based on trigram language model when the vocabulary consists of only two words.

Lexical tree representation effectively reduces the state space. For example, a lexical tree representation of a 12,306 word lexicon with only 43,000 phoneme arcs

had a saving of factor of 2.5 over the linear lexicon with 100,800 arcs [17]. Lexical trees are also referred to as *prefix trees*.

There is an important feature of the lexical tree. Unlike a linear lexicon, where the language model score can be applied when starting the acoustic search of a new word, the lexical tree has to delay the application of the language model probability until the leaf is reached. This aspect can be seen in the lexical tree search space based on bigram language model given in Figure 3.8 [23]. The triangles in the figure correspond to a lexical tree organized search sub-space like the one in Figure 3.5.

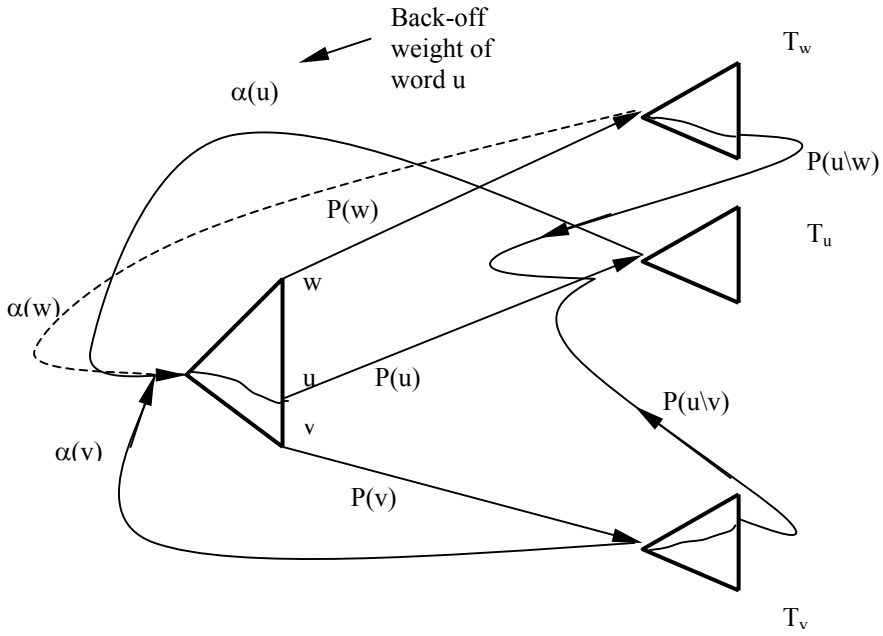


Fig. 3.8. Search space based on lexical tree with bigram language model.

Explanation of Figure 3.8: If no language model is used, it is sufficient to use only one lexical tree. Because the decision at time t depends on the current word only. For bigrams, a tree copy is required for each predecessor word. If an appropriate pruning is applied, the search space does not become as huge as expected; a small number of tree copies are processed.

The triangle on the left corresponds to the main tree and includes all the words in the vocabulary. The smaller triangles denoted as T_u, T_v and T_w correspond to the trees which contain words that can follow word u, v and w respectively. It is obvious that they will be smaller in size. The search begins with the main tree. Assume that

the hypotheses at the end of this tree are word u, v and w. Acoustic scores of these hypotheses are multiplied by the unigram probabilities P(u), P(v) and P(w).

Then, assume that we prune the hypotheses v and w because of their bad score, i.e. we will carry on with u. Now the search goes on through the tree T_u including the follower words of u. The bigram probabilities cannot be applied before ending up with this tree, because the successor word is not determined yet.

Meanwhile, another part of the search takes place in the main tree for the successor words that are not included in bigrams of u, T_u.

At the end of search for the second word, there are two groups of scores to be compared: one at the end of the unigram tree obtained by back-off mechanism and the other one at the end of the successor tree T_u , which is multiplied by bigram probabilities associated with follower words. The hypothesis with the best score in these groups is chosen as the recognized second word that is the successor of u.

It is clear that this process takes long time and brings computational load. Hence, a technique called "*language model look-ahead*" is proposed, which applies language model probabilities before ending the trees [13, 19], so that pruning can be applied in the tree, not at the end of it (See Section 3.2.2).

3.2.2 Pruning Techniques

It is clear that an exhaustive search in a state space of large numbers of HMM's is prohibitive. However, it is sufficient to evaluate 5000-10000 and fewer state hypotheses on average per 10-ms time frame (See Section 2.1.1). Thus, we retain only the most promising states alive. Then, the time synchronous Viterbi search becomes *Viterbi beam search*. Number of surviving states is called as *beam width*. Pruning approaches in time synchronous search consist of three steps:

1. **Acoustic pruning** is used to retain the states, the acoustic scores of which are closer than a given threshold to the score of the best hypothesis. If we define the best score as Q_{best} , the states having the acoustic score q_i satisfying

$$q_i < f_{AC} Q_{best}$$

are discarded. The beam width is determined by the acoustic pruning threshold f_{AC} .

2. **Language model pruning** is also known as *word end pruning.* The bigram probability is incorporated into the accumulated score and the best score for each predecessor word is used to start up the corresponding new hypothesis. Thus, the number of follower words is reduced. So, defining the best score after incorporation of the language model probability as Q_{LM} , a new start-up hypothesis is removed if its score q_i satisfies;

$$q_i < f_{LM} Q_{LM}$$

where f_{LM} is the language model pruning threshold.

In a prefix tree search, language model look-ahead technique is invoked to be able to apply this pruning before reaching the end of tree. This is achieved by applying the LM probabilities as a function of the nodes of the lexical tree. By way of doing that each node corresponds to the maximum LM probability over all words that can be reached via this specific node [25].

3. **Histogram pruning** limits the number of surviving hypotheses to a maximum number m. If the number of surviving states determined by the factor f_{AC} in acoustic pruning exceeds m, only the best m hypotheses are retained.

For example, if there are 5000 states alive determined by acoustic pruning threshold and if m equals to 4000, 1000 of the alive states will be discarded. But if m were given as 6000, the number of alive states would remain as 5000.

3.2.3 Cross-Word Expansion

In the case of continuous speech, many word boundaries are not clear and it is hard to separate the transition region between two words depending on acoustic features. To take into account the transitional effect between words, triphones including the end of the predecessor word, the start of the successor word and the short pause between them can be built during training process. A simple search

network built up of cross-word triphones can be seen in Figure 3.9. The expansion type shown in the figure is used in this thesis.

However, there is a problem with cross-word models during search. The actual cross-word triphone model depends on the successor word, but its identity cannot be determined before the identity of the successor word is known. Before recognizing the successor word, current word has to be recognized. One solution is to create copies of the current word for each possible successor, then recognize the successor and only keep the copies that are actually required. Different techniques are used to manage left contexts, right contexts and single phone words. For a detailed discussion refer to [12,14,16].

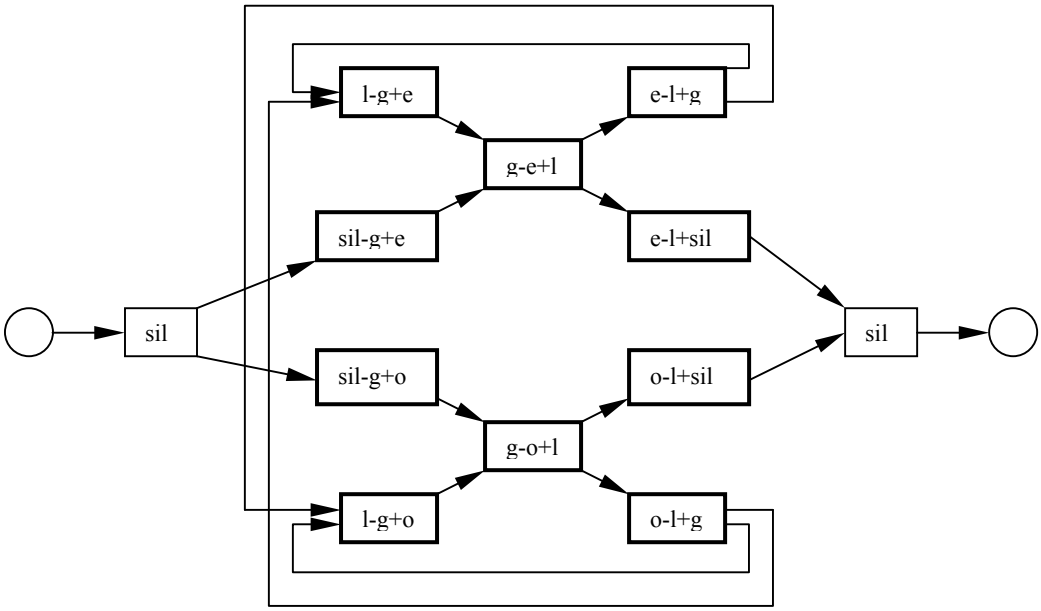


Fig. 3.9. Cross-word triphone expansion network with only two words ‘gel’ and ‘gol’.

3.2.4 Single Best vs. N-best and Word Graph

The term ‘single best’ is used to denote a search concept which determines the single most likely word sequence. The alternatives are n-best and word graph

methods. In n-best method, the output of the system is not a unique word sequence, but an ordered list of the n-best sentences. A word graph is an organized network that holds the high ranking sentence hypotheses and whose edges correspond to single words. Word graphs are also called as *word lattice*. An example for word graph can be seen in Figure 3.10 and a graphical comparison of word graph process and *integrated search* process is demonstrated in Figure 3.11 [13]. Integrated search is the one that combines language model, acoustic model and decoding techniques together in a one-pass search strategy.

Referring to Figure 3.11, another concept emerges: multiple pass search. After obtaining the word graph as a result of the integrated search, it can be inserted into a second pass search based on a higher order n-gram LM to rescore the results.

In token passing algorithm, n or more tokens are hold in each state to keep n-best hypotheses. Every token corresponds to a different path through search space ending at current state.

For further details refer to [13, 15].

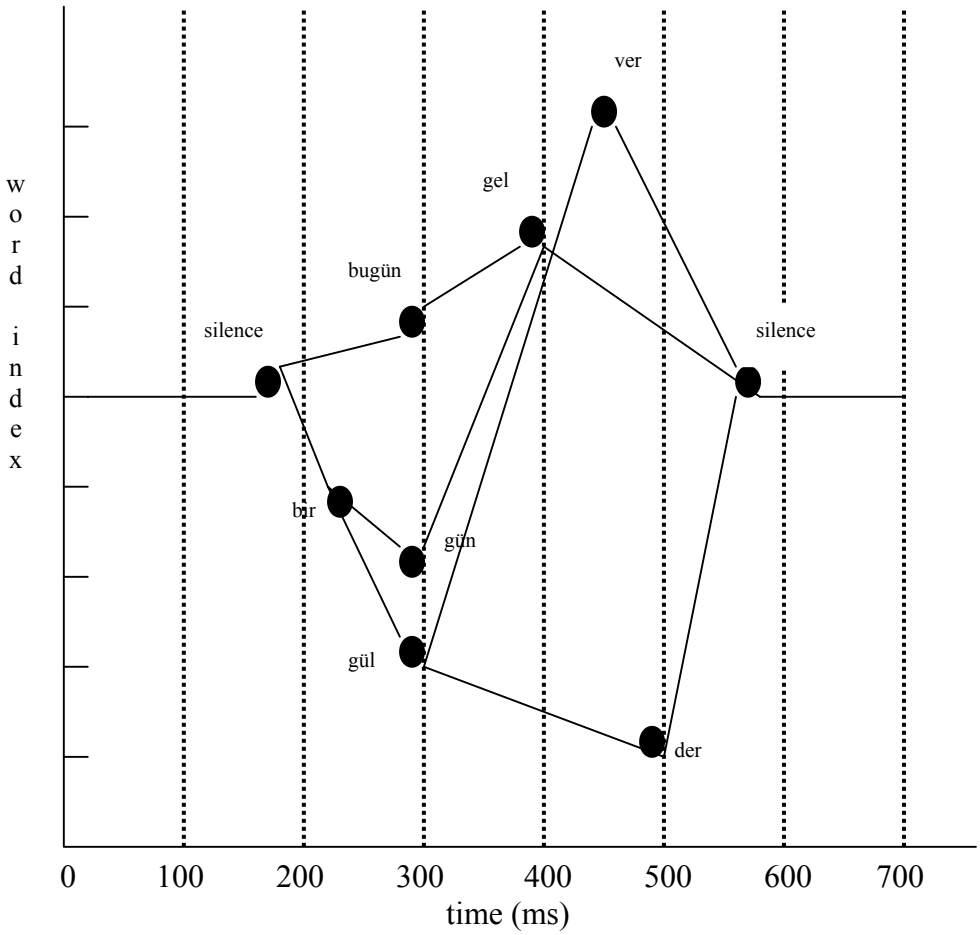


Fig.3.10. A simple word lattice generated by the first stage of a two-pass speech recognizer.

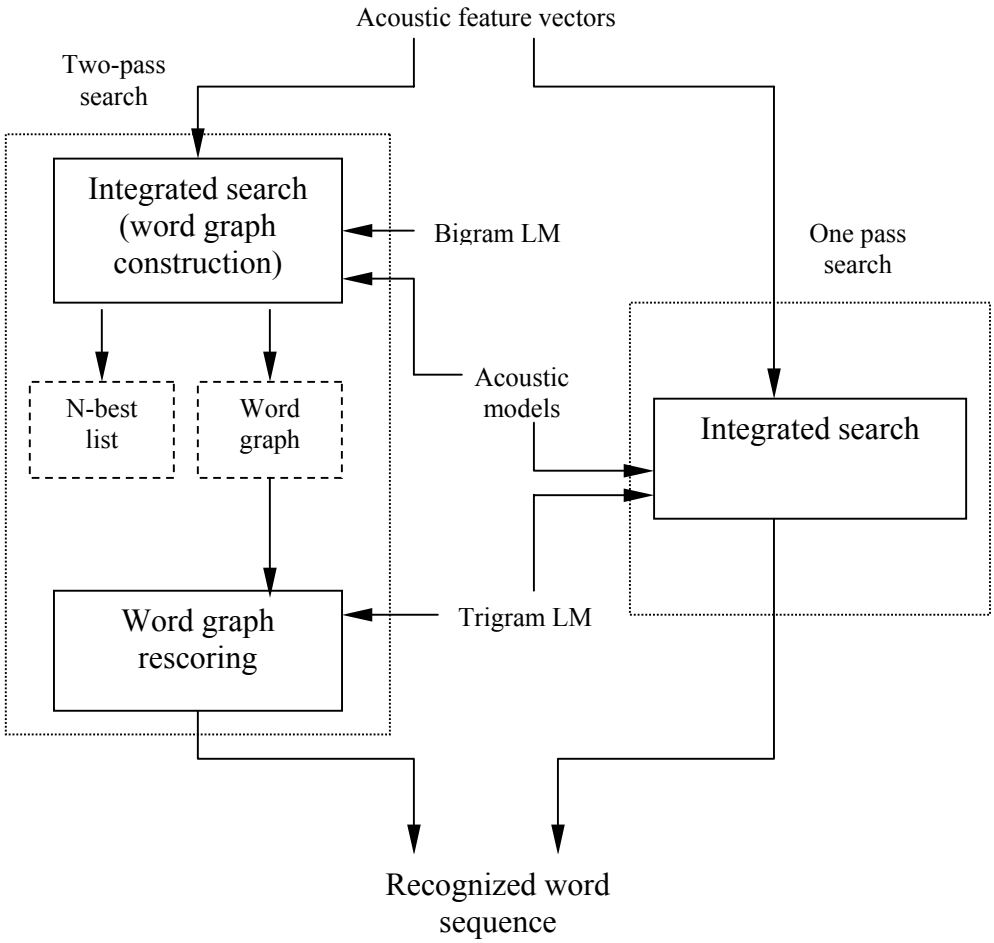


Fig. 3.11. General view of one-pass and multiple pass search strategies

CHAPTER 4

HTK IN BRIEF

The description of a software package in a research monograph is unusual. However, the use of HTK (Hidden Markov Model Toolkit) has been a fundamental part of speech recognition research and at the same time, resolving certain issues is not so trivial. The description here has been prepared to provide a better understanding of what has been done in this thesis and of the process involved in the recognition task by utilizing the modular structure of HTK.

The HTK is a software toolkit for building speech recognition systems. It can perform either continuous density, semi-continuous density or discrete probability HMM based tasks. It is developed by the Cambridge University Speech Group. It has been improved and added properties over years since the beginning of the nineties.

In this thesis, Version 3.1 is used to implement the recognition system. It is a freeware that can be downloaded from internet, but not with all of the features developed. For instance, one can only perform linear lexical search although it is reported that tree lexical search had also been performed [36].

The HTK is designed to be flexible enough to support both research and development of HMM systems. By controlling the tool software via commands with some options as desired, a speech recognition system can be implemented, tested and then its results can be inspected. A wide range of tasks can be performed, including isolated or continuous speech recognition using models based on whole word or sub-word units. HTK consists of a number of tools that perform tasks such as coding data, various styles of HMM training including embedded Baum-Welch re-

estimation, Viterbi decoding, producing N-best lists or single recognition result. It can perform results analysis and editing HMM definitions, too. Especially editing the HMM's externally enables the user to control the acoustic models with a considerable flexibility.

The HTK also supports language model constructing. The language model is based on n-grams. The version used is restricted to bigrams.

General aspects of HTK are given in the following sections. The technical details and instructions how to use HTK can be found in HTK book for Version 3.1.

4.1 Tools and Modules

There are four main phases when constructing a speech recognition system: data preparation, training, testing and analysis. The commands to use HTK and auxiliary software modules are related to these main phases. They have an abbreviated form of typing. These forms represent the commands executed in operating system environment, while some command-like abbreviations represent the module programs that are used by these commands. The commands (tools) use the modules when performing a user defined task. In this respect, the modules are internally embedded. However, the user can control these modules either defining an option in the command line or defining the desired parameters in a text file.

Basically, HTK deals with two types of files: text files and speech data files. Text files can hold some editing commands, configuration parameters, transcription of speech files or the list of the files that will be used when performing the task.

The processing stages are demonstrated in Figure 4.1 [34]. In the figure, the abbreviations in boxes are commands used in HTK.

4.2 File Types

Except speech data files, HTK is completely based on text files. Although text files have their own extensions related to their functions in HTK, they can be edited in any text editor like Notepad, WordPad or Word for Windows.

A detailed discussion about the formats of the text and speech files can be found in HTK book. However, it is worth of mention here that the speech files should be in 'wav' format if the user prepares the speech data with the software CoolEdit and will run HTK in MS-DOS. There are different kinds of 'wav' options in CoolEdit, but HTK can load only A-law/μ-law 8-bit (CCITT standard) wav files. It is recommended by the author to save the speech file in this format by choosing this option. Other 'wav' options make HTK to report error and stop processing.

The tool HSLab enables the user to record the speech and to label that speech. Labeling means to appoint a transcription to the speech concerned.

4.2.1 Label Files

A label file holds the content of an utterance. Labeling is a must, because the feature vectors of the speech will be associated with a word, syllable or phoneme and by way of this, the HMM's will be constructed. One can use these label files either in training or in analyzing the recognition results. In the analysis, the exactly uttered sentence contained in a label file will be compared with the sentence that is output by the recognition system. Consequently word and sentence correct recognition rates (See Section 2.3) are calculated.

Labeling can be in four levels: word, phoneme, triphone or biphone and syllable. In a label file, the portion of the speech to which the label belongs can be defined. Each line of the label file contains actual label optionally preceded by start and end times and optionally followed by a match score.

[start end] name [score]

Start and end times are in 100 ns units. These aspects can be seen in Table 4.1 and Table 4.2.

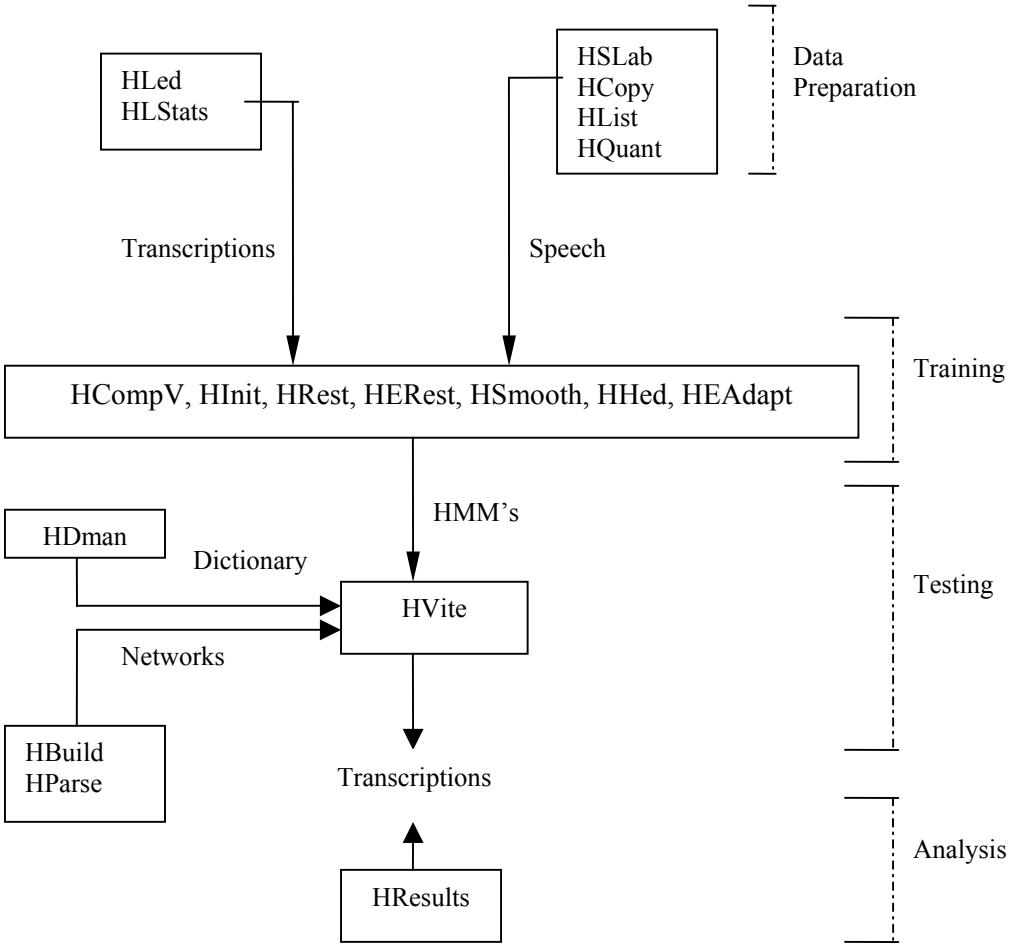


Fig. 4.1 HTK Processing Stages

Transcription level conversions are accomplished with HLed tool. It is a label file editor that performs inserting, deleting, replacing, merging and level converting with the help of a dictionary and of a text file that contains edit commands. A label file with a sentence on each line gives false results in converting a word level label into phone level transcription. But it does not cause error report. There has to be only one word on each line in a word level transcription.

Table 4.1. Triphone level label file content

120000 122000 sil-i+z -390.9
122000 125000 i-z+m -456.7
125000 127000 z-m+i -400.4
127000 129500 m-i+r -411.5

Table 4.2. Word level label file content

120000 129500 izmir
129500 139600 büyük

The use of numbers in labeling prevents the user from employing SAMPA phonetic descriptions. For example, if '2' is written to represent Turkish character 'ö', HTK reports error because it perceives '2' as a score or a time point on the speech signal and expects an alphabetical character. The phonetic description symbols used in this thesis can be seen in Appendix C.

Labels from different files can be written into a unique file, which is called 'master label file'.

4.2.2 Dictionary

Every line of a dictionary consists of a unique word and its pronunciation. A word may have more than one pronunciation and these pronunciations can be written one under the other. The general form of a line in the dictionary is as follows:

word [output symbol] p1 p2 p3 ….

For instance;

kaplan [kedigiller] k a p l a n

If the system recognizes the phone models “k-a-p-l-a-n” sequentially, it will output not “kaplan”, but “kedigiller”. If the angle brackets are left blank, nothing will be displayed as a result. If they are not used anyway, the word ‘kaplan’ will be displayed as recognition result.

The tool that deals with dictionaries is HDMan, which acts like HLed on dictionary files. It can merge different dictionaries, modify a dictionary, output a new dictionary and write statistical aspects of this dictionary into text file with the help of an edit command file. A dictionary is also an input to HLed tool when converting label levels.

For the use of HDMan, the input dictionary has to be sorted according to the ASCII values of the characters. The Turkish characters must be left as they are. If this is not the case, HTK reports error of not being sorted. On the other hand, HDMan will output these characters as a code; for example instead of “ş”, “\375” will appear.

The tool HDMan searches at first an edit command file called ‘global.ded’ by default. This name should be given to the edit command file.

4.2.3 HMM Definition Files

An HMM definition file contains the properties belonging to a particular acoustic model or it can be a master definition file (See Ch. 4.2.1).

HMM definitions in HTK have their own language. The rules of that language should be taken into account when writing a definition file. For example, a definition file begins with the prompt ‘~o’. This prompt states that the following arguments are common to all definitions, i.e. every HMM shares these arguments.

A definition file can be written manually as a text file and then modified using HHed tool. HHed is a text editor, too. It takes an HMM definition file, an edit command file and a list of models as input and modifies the input HMM definition file. HHed offers great flexibility in the process of acoustic training. The user can

add, delete, replace or insert transitions, states and mixtures. State tying is accomplished using HHed. Tied state definitions should take place at the beginning of the file with their own names, because they are shared.

In spite of dealing with less number of models in a sub-word based system compared to a word based system, HMM definition files occupy a lot of memory. For instance, approximately 6900 triphones are produced during acoustic training in this thesis and they have an amount of 20.5 MB. HTK gives the ability to save files in binary. With binary saving, the amount of storage needed for the example above is diminished to 5.86 MB.

4.2.4 Configuration Files

A configuration file contains user defined parameters that are used to control the modules. For example, extraction of the feature vectors of a speech signal needs a configuration file with content like the one in Figure 4.2 as input to the HCopy tool.

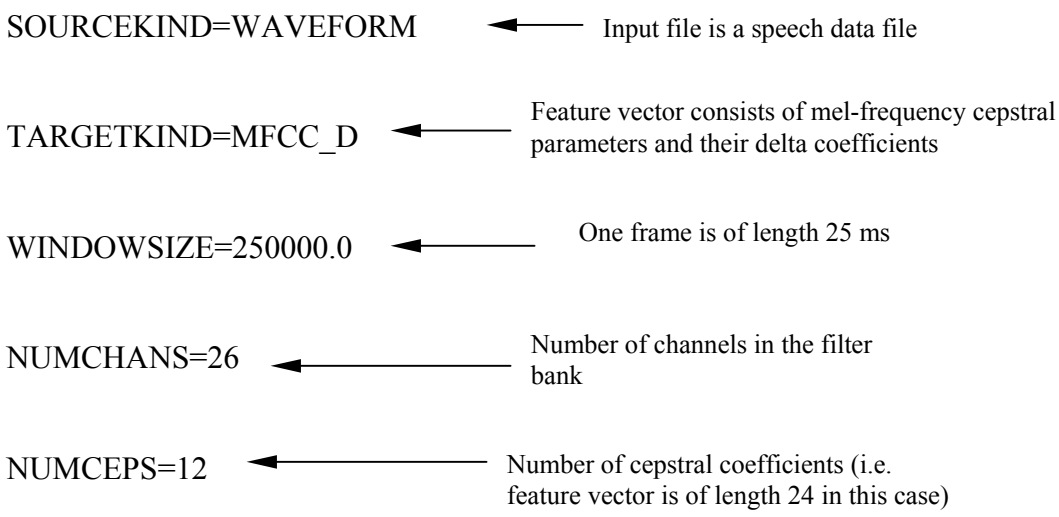


Figure 4.2. An example for the configuration file content

The tool HCopy is used to convert data files in other formats to the HTK format, to concatenate or segment data files and to parameterize the result.

4.2.5 Script Files

Tools which require a long list of files (e.g. wav files in acoustic training) allow the files to be specified in a script file. A tool can be invoked either by specifying the input files explicitly on the command line or by listing them in a text file. For the tool HCopy, the list of files has to be modified. Not only the files to be processed have to be specified. The name and extension of the file into which the converted parameters will be written have to be definitely written, too. This is shown below.

```
d:\wavs\ses1.wav     d:\feature\ses1.mfc
d:\wavs\ses2.wav     d:\features\ses2.mfc
.
.
```

In this example, the wav files in the folder “wavs” are converted into feature vectors containing parameters defined in a configuration file and then the resulting files are written into the folder “feature”.

4.2.6 Edit Command Files

These type of files are specific to HLed, HHed and HDMan tools. The commands written into these text files are in abbreviated forms. They have a special meaning corresponding to the tool. There exist common commands, too, e.g. “TC” command makes triphones from monophones in both HDMan and HLed tools.

4.3 Command Line

The general form of command line for invoking a tool is;

toolname [options] files …

The "toolname" variable consists of the commands seen in Figure 4.1. Only one command can be written each time. Options are denoted by either a capital letter or a small single letter. This letter is preceded by a dash and followed by a value, name, string or nothing. Capital letters denote the options that are common to all tools. They have the same meaning for every tool. However, small letters denote tool specific options. The options control the implicit modules. However, it is not necessary to write them. Every option has a default value and if not used, HTK assumes that it will be used with its default value. For instance, the command line

herest -s stats12 -M h12 -S trlist.txt -m 1 -H h11\hmmdefs.txt -B -L trilabs triphones.txt

estimates the HMM parameters. Meanings of the options in that command line are as follows:

-s: write the statistics of the training corpus into the file 'stats12'

-M: write the output files into the folder 'h12'

-S: take the files listed in the file 'trlist.txt' as input

-m: the minimum number of occurences needed for training a model is 1

-H: load the HMM definition file 'hmmdefs.txt' in folder 'h11'

-B: save the output in binary

-L: look for label files in the folder 'trilabs'

The last argument 'triphones.txt' is a script file that lists the triphones, parameters of which are to be estimated. If the '-m' option were not specified, it would be assumed to be 3 which is the default value for 'm'. If –S option were not

defined, the user would have to type every file name listed in “trlist.txt” on the command line.

4.4 Acoustic Model Training

Defining a topology required for each HMM by writing a prototype definition is an essential point of acoustic training. HTK allows HMMs to be built with any desired topology. Initially, a prototype HMM has to be defined. The purpose of the prototype definition is only to specify the characteristics and topology of the HMM. The mean or variance values encountered are not important in a prototype. The actual parameters will be computed by the training tools incrementally. The tools used in training can be seen in Figure 4.5.

The main topology offered in HTK is a 5-state HMM topology (See Figure 5.1). The non-emitting states don’t appear as a state definition in an HMM definition file. Their effect can only be seen in the transition probability matrix. The last row and the first column of that matrix are ‘0’.

The offered topology is violated in silence model design. There are two different silence models. One of them is the normal silence model. This model has the same topology as the main one, except the two transitions added from 2. to 4. and from 4. to 2. states. In addition, a short pause model is created. This model is assumed to occur between the words uttered. The silence model topologies offered are demonstrated in Figure 4.3, and Figure 4.4.

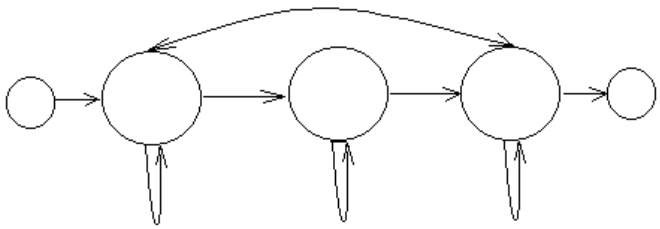


Figure 4.3. The silence model topology

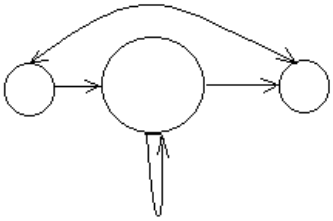


Figure 4.4. The short-pause model topology

Estimation of the parameters of these models progresses incrementally. At the beginning of the training process, the silence model is the same as the other phone models. Progressively the silence model is modified and the short pause model is inserted into the training process in later stages. The 2. state of the short pause model and the 3. state of the silence model are the same, i.e. they are tied states (See Section 2.1.3).

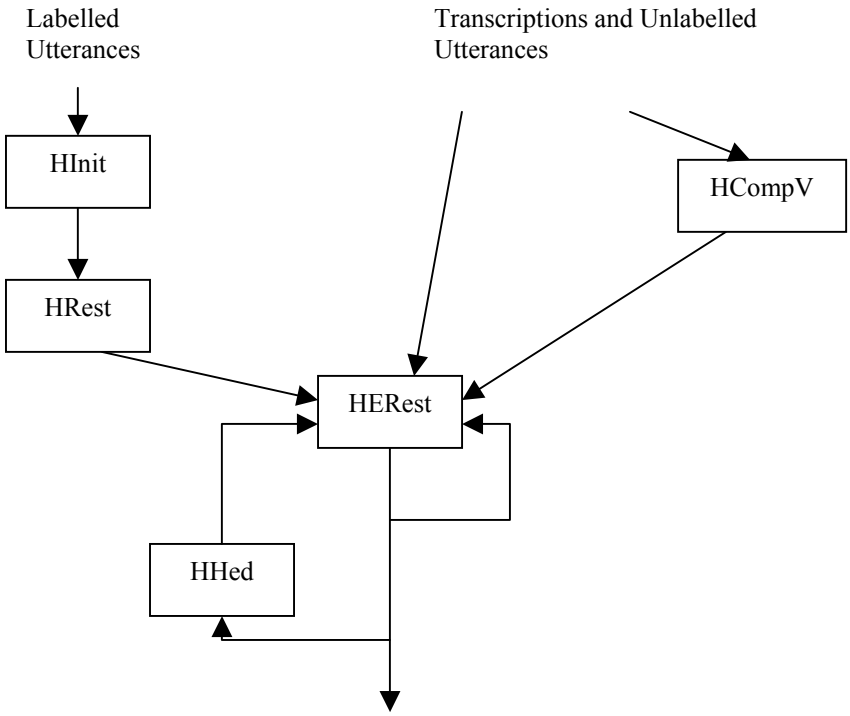


Figure 4.5. Training sub-word HMMs

This point of view should not be forgotten: training in HTK progresses incrementally. This strategy is the same when constructing multi mixture Gaussian context dependent triphones from single Gaussian context independent monophones. In the early stages of training, monophones are trained. Estimation of the parameters of the triphones takes place after all of the monophone, silence and short pause model's parameters are estimated. Number of mixtures in Gaussian distribution is incremented in the later stages of training. The stages of the acoustic training process are illustrated in Figure 4.5.

The tools HInit and HCompV are used to initialize the HMMs. If there is sufficient data with segmentation knowledge, i.e. if the location of sub-word boundaries on the signal is known, it is better to start with HInit. On the other hand, one might have no segmentation knowledge about the speech signal. Then, HCompV is used. This tool computes a global mean and variance using all of the utterances. In this case, all of the models are initialized to be identical and have state means and variances equal to the global speech mean and variance. However, HInit offers a better starting point than HCompV.

The initial parameter values computed by HInit or HCompV are then further re-estimated by HRest and HERest. The initial models created by HInit are trained by HRest isolatedly. This means that HRest will be run for each phoneme seperately. This tool is used in isolated speech recognition tasks, too. But HERest takes the entire non-labeled speech data as input. It updates all the models simultaneously. These two tools utilize Baum-Welch re-estimation procedure (See Section 2.1.2.3). In HERest, the corresponding phone models in each utterance are concatenated and then the forward-backward algorithm is used to accumulate the statistics of state occupation, means, variances etc. for each HMM in the sequence. When all of the training data has been processed, these statistics are used to compute re-estimates of the HMM parameters.

It is inevitable to have triphones in speech that do not occur in acoustic training corpus. In this case, artificial triphone models should be constructed from the triphones at hand. For this purpose, HTK supports decision tree clustering (See Section 2.1.3.1). The 'AU' command used in decision tree clustering process produces models for unseen triphones.

4.5 Language Model and Decoding Network

When testing a speech recognition system, the alignment between the test speech and the reference database is performed thorough a decoding network. As discussed in Chapter 3, this network constitutes the search space and is strictly related to the language model. HTK offers two kinds of language model. One of them is based on N-grams and is restricted to bigrams. Back-off bigram models including discount factors can be obtained using the command HLStats. The result of this command is a list of unigrams and bigrams followed by corresponding probability values. This list is the core of the decoding network. Referring to this list, a network definition file is obtained using the command HBuild. The arcs of the network that combine two words correspond to the bigrams in the text corpus.

The text corpus to obtain the language model probabilities should be in such a format that only one word be on each line. The start and end of the sentences should be denoted with a specific word or symbol in order to have a start and end point for the search. If this is not the case, HTK appoints default symbols “SENT-START” and “SENT-END”. These symbols should be included in the dictionary, too.

The other type of decoding network is constructed using the command HParse. This command takes a word level grammar definition as input and converts it to a network. The grammar is manually constructed by the user. This grammar has its own formulation consisted of slashes, brackets, square brackets and curly brackets and is called as Bacus-Naur form. These symbols may correspond to the rules of the language. The probabilities attached to the arcs of the output network are not N-gram probabilities. These probabilities are determined by the number of the followers of a word. For instance, if 10 words are defined to follow a word in the manually written grammar, the value ‘-log(10)’ is attached to each of the arcs emitting from that word. HParse enables the user to construct his own network.

A decoding network based on a grammar or including bigrams is an input to the recognition tool HVite. The two final stages and corresponding tools are discussed in the next section.

4.6 Testing and Analysis

Testing is the actual recognition phase and is accomplished with HVite tool. The speech signal to be recognized can be input directly on the fly or can be recorded before recognition. It is better to keep the recorded speech signal in terms of feature vectors, i.e. in parametrized form. This will help save time in testing phase.

HVite is a general-purpose Viterbi word recognizer. It will match a speech file against a network of HMM's. When performing N-best recognition a word level lattice containing multiple hypotheses can also be produced. The type of network expansion (context dependent triphones, biphones or context independent monophones) is inserted into HVite with a configuration file.

HVite performs Viterbi search with token passing algorithm (See Section 3.1.1). For N-best recognition, N tokens are hold in every state.

The recognition results can be output to the screen or to a text file. After obtaining the results, the performance of the system can be measured via HResults tool. It reads in a set of label files and compares them with the corresponding reference transcription files. The comparison is based on a dynamic programming-based string alignment procedure.

CHAPTER 5

EVALUATION RESULTS

In this chapter, the work actually performed in this thesis is given. The main goal was to test the suitability of a new language model that is expected to fit Turkish as an agglutinative language. Information about Turkish language and the training corpora is given. First, the words in the text used for language model training are parsed into two sub-word units; the stem and the ending. To make comparison, a language model based on bigrams obtained using stems and endings is tested. Then, the proposed language model based on bigrams obtained only using stems is tested. In addition, an IWR (Isolated Word Recognition) and a CWR (Connected Word Recognition) system is implemented.

5.1 Acoustic Model

To build an acoustic model, the monophones have to be defined at first. In this thesis, 34 monophone models are used. One of them models the silence and is denoted as ‘ccc’. The other silence model corresponds to the short pause that can occur between two words and is denoted as ‘ppp’. The second state of the short pause model and the third state of the silence model are identical. They are tied and trained together. The HMM topology of the monophone models is demonstrated in Figure 5.1. HMM topology for short pause and silence models can be seen in Figure

4.3 and Figure 4.4. The phone list including observation counts of phones is given in Appendix C.

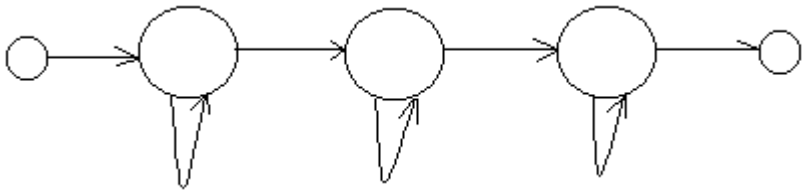


Fig. 5.1. The HMM topology for phone models.

5.1.1 Data Preparation

The speech data was recorded at METU consisting of 193 people's utterances. This data were first segmented by Ö. Salor [42]. The age range is from 19 to 50. Most of the speakers were university students. The meaning of this situation is that the corpus is heavily weighted on young voices and the accent variation is very small.

We had two groups of recorded speech data; one of them including segmented data and the other one being larger and including unsegmented data. The data to be segmented was selected from the unsegmented data arbitrarily and then segmented into phones. This segmented data is used to initialize the monophone models. As mentioned in Section 4.4, segmented data offers a better starting point for training than flat start technique does. The statistics of the acoustic training corpora are given in Section 5.1.2.

The main issue that consumes much effort in speech recognition is the preparation of the training data, either for acoustic or language model. Although we had the recorded data at hand, we had to document the contents of the speech records in terms of transcription files by listening to every record carefully in order to determine what is really uttered in the basis of monophones. The transcription files are not written according to well-defined writing rules of Turkish. For example, one or more pronunciations took place for some words, such as 'sarmısak', 'yapmayacak'

etc. If the speaker had pronounced ‘sarmsak’ or ‘sarımsak’, ’yapmıycak’ or ‘yapmıcak’, we wrote it as it is uttered, instead of the correct writing. Because, we have to correctly define the monophones at first. The pronunciation variations are also included in the dictionary that gives the monophone expansions of the words.

The phone boundaries in segmented data were in terms of milliseconds. We converted them into **100** nanoseconds, so that HTK can process them.

The feature vector that represents the distinctive properties of the speech is designed to be of length **39**, consisting of **12** mel-cepstrum coefficients and an energy component, and additionally their delta and acceleration coefficients. The structure of the feature vector is shown in Figure 5.2.

The configuration of the front-end processor is defined in a text file, the content of which is given in Table 5.1. Since we used recorded speech data in the test experiments, we could normalize the energy of the frames. Instead of magnitudes, we utilized zero-mean power mel-cepstrum coefficients in the configuration. The frames of length **25** ms are taken every **10** ms from the speech signal. The filterbank consists of **26** channels.

Table 5.1. The configuration parameters for the front-end processor.

SOURCEKIND=WAVEFORM
SOURCEFORMAT=WAV
TARGETKIND=MFCC_E_D_A_Z
USEHAMMING=T
SAVEWITHCRC=T
SAVECOMPRESSED=T
PREEMCOEF=0.97
ENORMALISE=T
TARGETRATE=100000.0
WINDOWSIZE=250000.0
NUMCHANS=26
NUMCEPS=12
USEPOWER=T

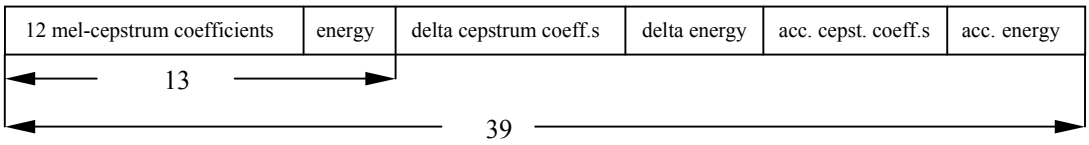


Fig. 5.2. The structure of the feature vector.

5.1.2 Training

In this section, construction of the acoustic model used in this thesis will be given. The flowchart that explains this procedure can be found in Appendix A.1.

We first constructed a prototype HMM definition based on the topology shown in Figure 5.1. The quantities in the prototype definition are not important. They are updated during training.

We assumed that the observation probability density function is continuous Gaussian and the covariance matrix of this density is diagonal, which means that the feature vector elements are de-correlated. At the beginning, the short pause model is not included in the training process and the silence model is the same with other phone models. At this step, Gaussian distribution consists of only one mixture.

We had 55 segmented utterances of 32 male and 23 female speakers. The monophone models are built one-by-one using these utterances with the tool HInit. HInit cuts the segments belonging to a particular monophone and uses these segments to train that monophone. After having obtained the initial models of monophones, we used the tool HRest to further train these models. In fact, HRest is designed to train word models separately for IWR tasks, but it can also be used to train phone models.

Before going on with the embedded training (See Section 2.1.2.4), we gathered the monophone models into a single HMM definition file. At this step, no component is shared between these models except the variance floor values and the structure of

the feature vector. Variance floor vector is obtained from the second group of the speech data that is unsegmented using the tool HCompV. HCompV computes the global variance vector and multiplies it by a value entered by the user such as 0.01. The resulting vector is used to determine the minimum variance value that is allowed for models during training.

The unsegmented data, that is used to train phone HMM's in parallel, consisted of **7650** utterances spoken by **104** male and **89** female speakers. The corpus contained **52155** words with a vocabulary of size **9360** words.

We ran the tool HERest twice on the initial HMM's, in order to perform embedded training. This tool is invoked at least twice after any modification to models is applied.

Then we added two transitions between the second state and the fourth state of the current HMM topology of the silence model to improve the model. The idea is to allow individual states to absorb the impulsive noises. The backward transition from state 4 to state 2 makes this possible without committing the model to transit to the following phone model. Its topology has become the one shown in the Figure 4.3. We inserted a short pause model with topology shown in Figure 4.4 into HMM definitions. The transcriptions are modified as to include this short pause model at the end of every word, too. The second state of the short pause model and the third state of the silence model are tied.

After two iterations of HERest, the transcriptions are modified so as to include triphones. Monophone models represent the features of the phones without including the effect of the context, whereas the triphone models include the effect of the context. The modification of the transcriptions can be seen in Figure 5.3. modification is achieved by using HLed. The HMM definitions are also modified; they are converted from monophone into triphone models. The number of triphones was **6689**. At this step, the transition matrices of these triphone models are tied.

Again, two iterations of HERest are performed. Then, using the tool HVite, alignment had been made. Alignment means that the training speech data is segmented according to its transcription; i.e. the boundaries of the models included in the transcription are found. Here, the boundaries of the triphones are determined. The training will go on with this segmented data.

120000 122000 i

122000 125000 z

125000 127000 m

127000 129500 i

129500 132500 r

(a)

120000 122000 sil-i+z

122000 125000 i-z+m

125000 127000 z-m+i

127000 129500 m-i+r

129500 132500 i-r+sil

(b)

Fig.5.3. (a) Monophone transcriptions, (b) triphone transcriptions.

After the parameters of triphones are re-estimated, state tying is performed. State tying is accomplished with DTC method (See Section 2.1.3.1) that needs the statistics file output by the tool HERest in the previous step. The statistics file includes the occupation counts for all states of the HMM's and observation counts of the models. The occupancy count is the number of frames that are allocated to a particular state.

We chose the outlier threshold for DTC **200**; i.e. the minimum number of frames in a cluster is limited to 200. The questions to use in clustering in this thesis are given in Appendix B.

One important point in DTC is the construction of the unseen triphones artificially. For this purpose, the unseen triphones should be included in the list of triphones occurred in acoustic training. One way is to add the triphones that are in the text corpus used for language model training but not occurred in acoustic training. Although we listed the triphones in the text corpus, we were not contented with this list. The theoretical number of triphones for a 34-monophone list is 34^3 = 39304. Excluding some unprobable triphones, such as 't-t+t' this number should

diminish. We used **39104** triphones in order to build unseen triphones. This huge amount of triphones does not imply additional effort for the search process. Because, the decoder looks up the pronunciation rules defined in the lexicon and determines which models should be used during search. The statistics about triphones are given in Table 5.2.

Table 5.2. Triphone statistics

Theoretical triphones	39304
Exclusive triphones	39104
Triphones from text corpus	9417
Triphones from acoustic training corpus	6689
Triphones after DTC	2318

Referring to the Table 5.2, the actual number of HMM's used in recognition is **2318**. They are referred to as *physical* HMM's. However, because of state tying, they can correspond to **39104** HMM's and these are referred to as *logical* HMM's.

The last step of acoustical model training consists of mixture incrementing. We utilized the statistics file again, which is output by HERest tool. According to the observation counts, we incremented the number of mixtures in the Gaussian distribution up to 4-8. In consequence of incrementing the number of mixtures by 2, we ran HERest twice. The improvement in average log likelihood per frame achieved in training can be observed in Figure 5.4.

5.2 Language Model

The text corpus used to obtain bigram probabilities consists of sports news downloaded from internet. Firstly, we eliminated the foreign names that include

foreign letters x, w and q and the pronunciations of which do not obey the rules of Turkish. However, very popular foreign names such as ‘Okocha’ or ‘Ajax’ are left.

To define the start and end of the decoding network, the beginnings and endings of the sentences are denoted as special words ‘bbaaSS’ and ‘ssoonn’. In the recognition phase, these words are associated with the silence model and they are included in the dictionary.

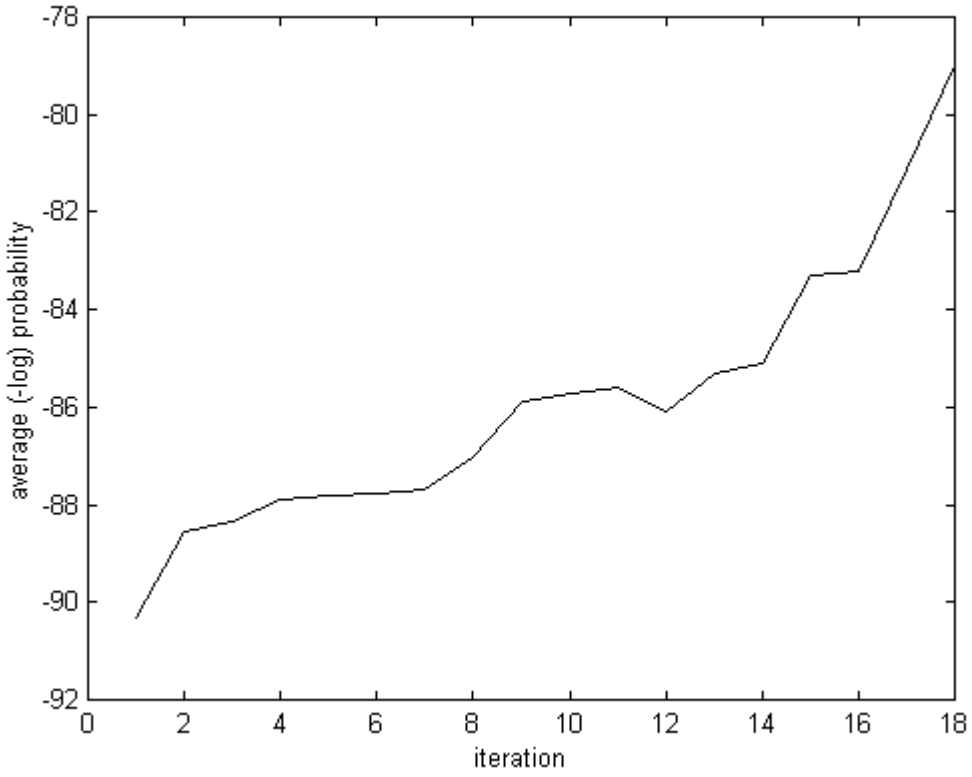


Fig. 5.4. Improvement in average (-log) likelihood per frame achieved in acoustic model training

Correction of the writing errors took most of our effort. There are two basic causes; first, the editors do not even look up a Turkish dictionary when editing. Second, they admire foreign languages so much that they try to use foreign terms even their command of English, German, Italian etc. is not well and even there exists a Turkish term instead of it.

Before explaining the language model training work, it would be better to give the basic morphology of Turkish.

5.2.1 Basic Turkish Morphology

Turkish is an agglutinative language being highly productive in terms of words. New words are constructed by appending suffixes to stems or words to words. However, the word-to-word production is not as much functional as it is in German. Especially, the proper nouns can be produced with this type of production, such as 'Beylerbeyi' or 'Kadınhanı'. Unfortunately, there emerged a trend to use this word production more frequently; for example, the doubling phrase 'el ele' has begun to be written as 'elele' where the first form is the right one.

Suffixes are either inflectional or derivational suffixes. They can be affixed one after another. They can convert the word from a nominal to a verbal structure or vice-versa. In his popular paper [28], Oflazer proposes a two-level description of Turkish morphology. According to the two-level description, a word can be represented in two forms; one, the *lexical form* and two, the *surface form*. We used the surface form in this thesis. Surface form corresponds to the actual spelling of the word. Lexical form consists of a sequence of morphemes that are the smallest meaningful components of a language. Transition from lexical form into surface form is achieved according to phonetic rules of the language.

To illustrate the productivity of Turkish, we can give the example that we ran into in the text corpus;

Gör + ev + le + n + dir + il + me + me + si + ne

where '+' denotes the morpheme conjunctions. The last morpheme is actually '+e', but 'n' is inserted according to the phonetic rules of Turkish. Translation of the above word into English is; 'to his not being charged'.

The most important phonetic rule of Turkish is the *vowel harmony* rule. Vowels in the suffix have to agree with the ones in the stem. Sure, there are exceptions, but most of them are assimilated from foreign languages like Arabic and French. In some cases, vowels in the stem or suffixes are deleted; e.g.

karın ⟶ karn + ı + nda

where the letter ‘ı’ in the stem is deleted. Like that, consonant in the stems or in the concatenated morphemes are modified or deleted such as

birçok → bırçoğ + u + nda

where the letter ‘k’ is modified to ‘ğ’. If the continuous tense suffix is affixed to a stem ending with a vowel, this vowel is deleted:

de + iyor → diyor

kapa + ıyor → kapıyor

This suffix also violates the vowel harmony rules. The stems ending with a vowel exhibit different affixations in the genitive case and in the inflection of the verb ‘imek’ (to be). This verb is affixed to the stem that represents the object of the sentence. For example;

paşa + ım (to be) → paşayım (y is inserted)

paşa + ım (genitive –my-) → paşam (ı is deleted)

These simple examples show that the morphotactics of Turkish are very complex. In addition, not all of them are given here. We tried to give an idea about the work done in this thesis: We used three text corpora in order to perform experiments. First, the text included words with their suffixes concatenated. In this case, every inflectional or derivational version of a stem corresponds to a different word. Second, the text included the stems and their endings, consisting of one or more morphemes, separated. Every stem and ending is treated as a different word. Third, the text included only stems.

We parsed the words into their stems and endings with a vocabulary-specific software that we wrote in C++ programming language. This is done with a list file, which contains the words to be parsed and their expansions to stems and endings. Our program looks up this file to determine how to parse a word in the text corpus. Then it writes the parsed text corpus into another text file.

5.2.2 Language Model Training and Decoding Network

As mentioned in Section 5.2.1, at the beginning, we had a text consisting of words that are not parsed into their morphemes; i.e. the derivational and inflectional versions of a stem are treated as a separate word. We constructed a decoding network, say Network A, using this text (See Section 4.5 and Chapter 3). Then we parsed the words in this text into their stems and endings. Endings consist of one or more morphemes. It is shown in [30] and in [9] that a language model based on stems and endings for an agglutinative language performs the best among word-based and morpheme-based models. Hence, we chose to build our model on stem-ending based bigrams.

The vocabulary of the stem-ending based model was built according to the aspects given below.

1. The modified versions of the stems ending with stop consonants ‘ç, k, p, t’ are included in the vocabulary as separate words, such as ‘git’ and ‘gid-’, ‘birçok’ and ‘birçoğ-’.
2. The stems, especially Arabic oriented ones, having the last consonant doubled in suffixation, are included as a separate word, too. Like ‘hak’ and ‘hakk-’.
3. At first, we had parsed the words strictly according to the affixation rules. Then we had a vocabulary of size about **23,000** words. But some suffixes are not very functional, such as ‘-ıt’, ‘-kı’ and ‘-ik’ for example in the words ‘yap+ıt’, ‘kat+kı’ and ‘geç+ik+mek (gecikmek)’. That is why we decided to affix these suffixes to the stems and obtain new stems. This idea complies with the information theory, too, which says that very frequently seen events cannot be regarded as *information*, e.g. in English, ‘q’ is always followed by ‘u’, which does not have a meaning in the sense of information. So, extra affixation of these suffixes reduced the vocabulary size to **18,326**.
4. The words including one-letter morphemes are not parsed. This means that the accusative, dative and the genitive forms of the stems are left. For

example, ‘kedi’ and ‘kedi+m’ (gen.), ‘cam’ and ‘cam+ı’ (acc.). The aim is to prevent acoustic confusion during recognition.

5. The stems having one vowel deleted are included separately, such as ‘burun’ and ‘burn+u+n+da’. But, if the word was in its accusative or dative form, it is left without being parsed. For the example ‘burun’, the word ‘burnu’ (acc.) is left unchanged and included in the vocabulary.
6. The proper nouns were left in their nominal forms. For example, ‘Bilgili’ can be parsed as ‘bil+gi+li’, but in our text, it was mostly used as the name of the president of Beşiktaş Football Club.
7. We made a last control to differ between homonyms (the words that are written the same but having a different meaning), such as ‘hata+ya’ and ‘hatay+a’, ‘bas+ın’ and ‘basın’, ‘iz+i+n+de’ and ‘izin+de’. But there are some morphemes that cannot be differed in parsing. For example, ‘gönder’ is a stem and cannot be parsed furthermore. It has two meanings. To make a difference between these, they can be labeled as ‘gönder1’ and ‘gönder2’ in a future work.

After parsing the words, we obtained a network from this text using the tools HLStats and HBuild, say Network 1.

In the third step, we obtained bigram probabilities only over the stems. The underlying idea is that the actual informative part of the word is the stem. However, the stems obtained in parsing had to be modified as to become pure stems. We corrected the stems again before we obtained stem-based bigrams, such that the stem ‘burn-’ became ‘burun’, ‘camı’ became ‘cam’ etc. Then we converted these probabilities into a network, say Network B. We transferred the transition probabilities of this Network B to Network A and obtained a new hybrid network, say Network 2. The experiment 4 is made with Network 1 and the experiment 5 is made with Network 2. Then their results are compared.

It should be noted here that all of the networks mentioned is based on **back-off bigrams** without smoothing. The formulae of obtaining back-off bigrams were given in Section 2.2.

The process of building decoding networks can be seen in Appendix A2. The statistics of these three text corpora are given in Table 5.3.

Table 5.3. Text corpora statistics

	Number of words	Vocabulary size
Text 1 (unparsed)	434,601	49,168
Text 2 (parsed into stems&endings)	650,738	18,326
Text 3 (only stems)	434,601	7,974

The item ‘number of words’ in Table 5.3 is the same for Text 1 and Text 3, because Text 3 consists of the stems of the words included in Text 1. When we look up to the Table 5.3, it is immediately seen that parsing reduces the size of the vocabulary. In fact, the vocabulary sizes might be much smaller in a text with a different topic, because there are lots of foreign proper nouns in the corpora consisted of sports news. Not to forget, the endings are regarded as separate words. They can construct a meaningful word if only they are affixed to a stem.

On the other hand, the decoding network is expected to reduce in size, too. The number of nodes and arcs in Network 1 and 2 that are explained above are given in Table 5.4.

The bigram probabilities obtained only over stems are expected to be safer than the ones obtained without any parsing. One way to check this could be to look up at the number of words, observation counts of which fall below a given threshold. The cumulative percentages of the number of words that are seen **10** times or less are given in Figure 5.5.

Table 5.4. Network sizes for Experiment 4 and 5

	Nodes	Arcs
Network 1	18,326	248,113
Network 2	49,168	363,007

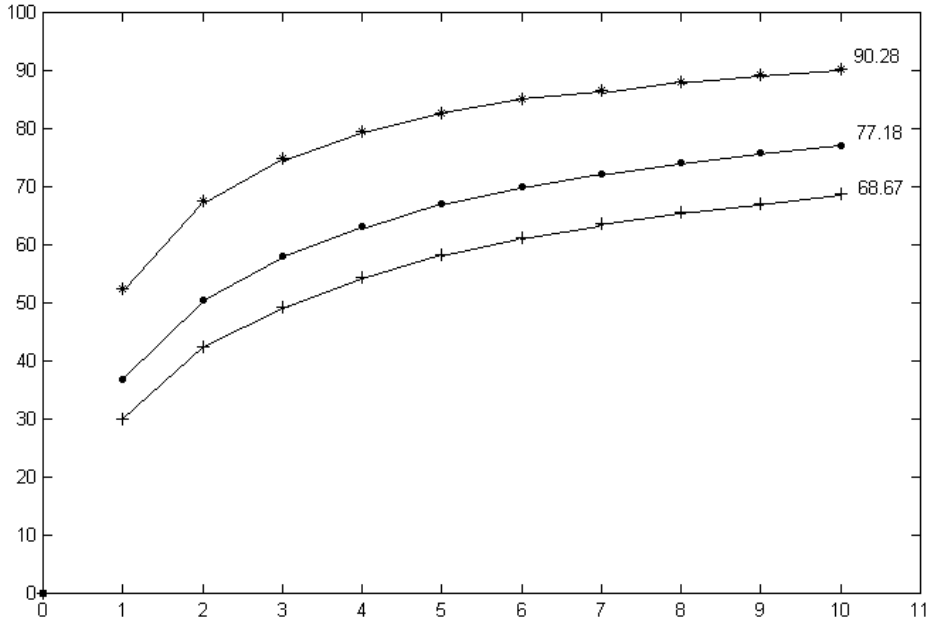


Fig. 5.5. Cumulative percentages of words observed 10 times or less.

Explanation of Figure 5.5: The graph punctuated with '*' represents the words that are observed 10 times or less in Text 1. This text consisted of words that are not parsed into stems and endings. The graph punctuated with '**.**' corresponds to words in Text 2. This text consisted of stems and endings that are treated as separate words. Finally, the graph punctuated with '+' corresponds to words in Text 3. This text consisted only of stems. When we analyze the figure, we see that **90.28**% of Text 1 consists of words that are seen 10 times or less. This means that the text is not appropriate for obtaining robust bigram probabilities. If Text 1 is parsed into stems and endings, the percentage of words that are seen 10 times or less drops to **77.18**%. The percentage subject to mention is **68.67**% for Text 3; i.e. a text corpus consisting only of stems provides a better condition for obtaining bigrams and we can say that smoothing operation is automatically performed.

The complexities of these language models are to be compared, too. The complexity measures perplexity and entrophy related to these language models are given in Table 5.5.

Table 5.5. Complexity measures of language models

	Entrophy	Perplexity
LM 1 (Text 1)	7.739	213.59
LM 2 (Text 2)	6.483	89.48
LM 3 (Text 3)	7.037	131.24

5.3 Test Results

We performed 5 experiments to test the ASR system built. The first experiment is an IWR task, the second and third ones are CWR tasks. In Experiment 2, we applied a network with no grammar; i.e. words can follow each other with no rule-based constraint. In Experiment 3, we applied a network based on a grammar that is structured according to test utterances. In Experiment 4, we tested Network 1 with the language model obtained from Text 2; i.e. bigrams based on stems and endings. In Experiment 5, we tested Network 2 with the language model obtained from Text 3; i.e. bigrams based only on the stems (See Appendix A.2).

The test utterances include **220** sentences that are arbitrarily selected from the text corpus that is used to extract bigram probabilities. They do not overlap with transcriptions of the speech data used in acoustic model training. These sentences were continuously spoken by 6 speakers (4 male, 2 female), who contributed in acoustic model training, too. These 220 utterances have a vocabulary of **1168** words.

Experiment 1: Isolated Word Recognition

To perform an IWR task, we cut **40** speech segments from test data, which contained only one word. The test vocabulary consists of **1168** words, not being parsed into stems and endings. The task can be defined in extended Backus-Naur form as

START $word END

where the variable $word denotes a word in the vocabulary and START and END corresponds to start and end of the utterance. This means that only one word is uttered each time in isolation.

Extended Backus-Naur form is a high level grammar notation where

| denotes alternatives

[ ] encloses options

{ } denotes zero or more repetitions, and

< > denotes one or more repetitions.

The recognition is based only on the acoustic score. But instead of word models, we used triphone models constructed in acoustic model training.

At the end of the test, we achieved a correct word recognition rate **55.17** %.

Experiment 2: Connected Word Recognition (No Grammar)

In this experiment we performed a CWR task based on a network with no grammar; i.e. every word can follow another one employing no rule. The task can be defined as

START <$word> END

where '< >' states that the variable $word may be repeated one or more times. In this case, the sentence uttered may consist of all words in the vocabulary or of only one word. A network with **1170** nodes and **3492** arcs allowing cross-word expansion is built. There is not a language model probability applied.

The test utterances are the same as the ones mentioned in Section 5.3, paragraph

Table 5.6. Recognition results in Experiment 2.

	%
Correct Sentence Recognition Rate (CSRR)	0.45
Correct Word Recognition Rate (CWRR)	41.28
Accuracy (Ac)	-111.87

2. The vocabulary size is again **1168**. This vocabulary is extracted from the test utterance transcriptions.

The recognition results achieved at the end of the test are given in Table 5.6.

The term accuracy is associated with the correct word recognition rate. The correct word recognition rate is measured with the formula;

$$100\frac{\mathrm{N - D - S}}{\mathrm{N}}$$

where N denotes the total number of words in recognized sentences, D denotes deletions and S denotes substitutions. This formula does not include insertions. Accuracy is given as;

$$100\frac{N - D - S - I}{N}$$

where I denotes insertions.

Experiment 3: Connected Word Recognition (With Grammar)

In Experiment 3, a CWR task based on a network with a simple grammar is performed; i.e. follower words are determined according to this grammar (See Section 3.1). The task can be defined as

START $w1 $w2 ($w3 | ($w3 $w4) | ($w3 $w4 $w5) | …….| ($w3 $w4 $w5 $w6 $w7 $w8 $w9 w$10 w$11 $w12 $w13 $w14)) END

where '|' correspons to the logical 'OR' operator and the variable $wi denotes the group of words that are placed in the i'th position in the test utterance transcriptions. In this case, the sentence uttered may be of length at least 2-words or at most 14-words. The network built consists of **12763** node and **25371** arcs, allowing cross-word expansion. There is not a language model probability applied. The general structure of this network is demonstrated in Figure 5.6.

The test utterances are the same as the ones in Experiment 2. The vocabulary size is again **1168**. This vocabulary is extracted from the test utterance transcriptions.

The recognition results achieved at the end of the test are given in Table 5.7.

When we look up at the Table 5.7, the effect of applying a grammar to the CWR task is obvious. It provides improvement in CSRR, CWRR and Ac compared to the results of Experiment 2.

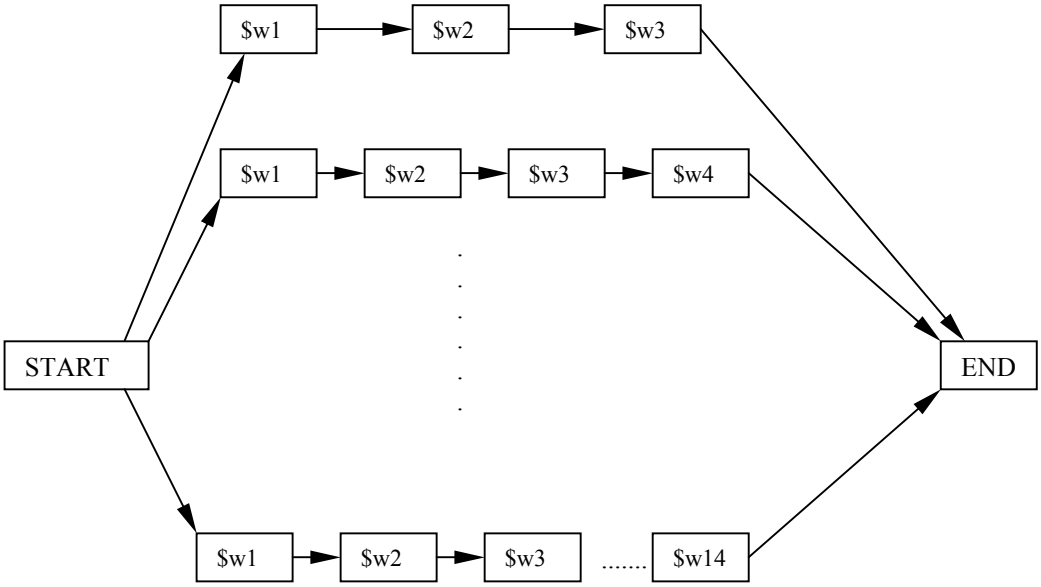


Fig.5.6. The general structure of the decoding network in Experiment 3.

Table 5.7. Recognition results in Experiment 3.

	%
CSRR (Correct Sentence Recognition Rate)	19.55
CWRR (Correct Word Recognition Rate)	47.17
Ac (Accuracy)	32.76

Experiment 4: Continuous Speech Recognition (Stem-Ending Based Bigrams)

The detailed discussion about the language models and networks used in Experiment 4 and Experiment 5 was made in Section 5.2.2. The statistics of networks and language models are given in Table 5.3-5.5. However, the language

model used in this experiment can be visualized with an example as follows. Assume, we have the sentence;

Havalar sıcak.

After parsing, it takes the form

Hava lar sıcak.

All parts in the sentence right above are treated as a separate word, be it stem or ending. We have the bigram probabilities P(lar\hava) and P(sıcak\lar). The cross-word expanded network contains these probabilities and their back-off probabilities implicitly.

Table 5.8. Results with different s and p values for Experiment 4. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

s	**p**	**CSRR**	**CWRR**	**Ac**
10	10	8.27%	57.35%	33.64%
10	20	7.82%	56.38%	16.64%
10	30	6.45%	54.12%	12.29%
20	10	17.36%	61.19%	50.50%
20	15	17.36%	61.87%	49.57%
20	20	16.45%	61.33%	47.45%
20	30	15.55%	62.51%	41.42%
30	10	18.27%	57.21%	51.86%
30	20	18.27%	58.75%	52.00%
30	30	17.36%	60.36%	51.25%
40	20	17.36%	50.43%	46.59%

In this experiment, we first tried to find near-optimal values for *language model probability scaling factor* s and *fixed penalty* p. These variables are used to control word insertion and deletion levels. Every language model probability value is multiplied by s and p is subtracted from the result. For example, if p=10 and s=20,

the probability x becomes 20x-10. If p gets higher, more short words are inserted into the recognized sentence. This causes the insertion errors get high.

The recognition results achieved by changing the values of s and p in the CSR task with a vocabulary of **18326** words and based on a network with **18326** nodes and **248,113** arcs are given in Table 5.8. Histogram pruning threshold applied in this experiment is **2000**.

According to the table, the optimal values for s and p are **30** and **20** respectively. There is a trade off between the CSRR, CWRR and Ac values. For example, in the seventh row of Table 5.8, the CWRR value reaches its vertex. But the Ac and CSRR values in the ninth row tell us to choose the configuration in this row. On the other hand, one has to choose the row with the higher value of Ac if there is an equality between CSRR values of two different rows. In a configuration with a higher CWRR value than the Ac value, there are redundant words in the recognized sentence (See Section 2.3.). The configuration with higher Ac value offers more robust recognition result.

Then, we tested the effect of different pruning threshold values denoted by u, applying the optimal s and p values. The recognition results for different pruning threshold values are given in Table 5.9.

Table 5.9. Results for different pruning thresholds in Experiment 4. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

u	**CSRR**	**CWRR**	**Ac**
1000	13.73%	51.22%	44.54%
3000	19.64%	61.19%	55.45%
4000	19.64%	61.33%	55.48%

The threshold 1000 is obvious to provide a suboptimal beam search. Although the result of 4000 seems to be a bit better, we chose **3000** to apply in the next stage because it offers lower computational load. In the next stage, we decreased the back-

off transition probabilities in the network, as to have a lower effect in the search. The comparison of the results can be seen in Table 5.10.

Table 5.10. Comparison of the results of the networks with back-off mechanism changed and unchanged in Experiment 4. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

	s	**p**	**u**	**CSRR**	**CWRR**	**Ac**
Back-off not changed	30	20	3000	19.64%	61.19%	55.45%
Back-off changed	30	20	3000	20.09%	61.22%	54.30%

During these tests, the average recognition process time per sentence was **1^{m} 34^{s}** .

Experiment 5: Continuous Speech Recognition (Stem Based Bigrams)

Refer to the example sentence given in Experiment 4;

Havalar sıcak.

Table 5.11. Results for different values of s and p for Experiment 5. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

s	**p**	**CSRR**	**CWRR**	**Ac**
10	10	5.55%	28.24%	-15.59%
10	20	4.18%	28.08%	-30.62%
20	10	8.73%	28.24%	15.32%
20	20	8.73%	29.38%	11.92%
20	30	8.73%	29.81%	5.70%
30	10	8.73%	24.89%	20.46%
30	20	8.73%	25.86%	19.22%
30	30	8.73%	27.65%	17.86%

The structure of the network used in this experiment is based on the words that are not parsed; i.e. the word ‘havalar’ is left in the vocabulary as it is. But the probability P(sıcak\havalar) is made equal to the probability P(sıcak\hava) (See Section 5.2). The testing procedure in this experiment is the same as the procedure in the Experiment 4. First, we test the effect of the values of s and p. The recognition results for different values of s and p with histogram pruning threshold **3000** are given in Table 5.11.

Taking into account the CSRR values, it is understood that the threshold value 3000 offers an over pruning. The choice s=**30** and p=**20** seems to be most promising among the alternatives. Then we applied these values chosen to different threshold values. The results can be seen in Table 5.12.

The threshold value **7000** is the most promising one among others. But there is need for some fine tuning. To achieve this, we tested this value with different s and p values again. The best result obtained then will be used in the back-off mechanism test. Results for different values of s and p with a pruning threshold 7000 can be seen in Table 5.13 .

Table 5.12. Results for different pruning thresholds in Experiment 5. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

u	**CSRR**	**CWRR**	**Ac**
4000	13.27%	32.35%	26.78%
5000	13.73%	33.97%	28.35%
6000	15.55%	36.95%	31.59%
7000	15.55%	39.32%	34.30%

From Table 5.13, we understand that the best choice is s=**30** and p=**25**. As it is done in Experiment 4, we decreased the back-off transition probabilities, as to have a lower effect in the search. The comparison of the results can be seen in Table 5.14.

Table 5.13. Results for different values of s and p with pruning threshold 7000 in Experiment 5. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

s	p	CSRR	CWRR	Ac
40	20	12.36%	31.49%	29.11%
30	25	16.00%	40.35%	34.08%
35	20	14.18%	37.11%	33.86%

During these tests, the average recognition process time per sentence was **1^{m} 57^{s}**. The main reason of longer process time in Experiment 5 than the one in Experiment 4 is the size of the network constructed.

Table 5.14. Comparison of the results of the networks with back-off mechanism changed and unchanged in Experiment 5. (CSRR: Correct Sentence Recognition Rate, CWRR: Correct Word Recognition Rate, Ac: Accuracy)

	s	p	u	CSRR	CWRR	Ac
Back-off not changed	30	25	7000	16.00%	40.35%	34.08%
Back-off changed	30	25	7000	30.90%	67.86%	52.30%

In the next chapter, we conclude the thesis analyzing these results.

CHAPTER 6

CONCLUSION

In this thesis, we made five experiments. The first one was the IWR (Isolated Word Recognition) task. In the second one, we tested a CWR (Connected Word Recognition) system with no grammar, whereas in the third one we tested a CWR system with a simple grammar that we designed. The fourth experiment was related to a CSR (Continuous Speech Recognition) system, in which the cross-word expanded network is based on bigrams that include stems and endings. The fifth experiment was performed in order to test the language model that is actually proposed in this thesis. In this experiment, the cross-word expanded network was based on bigrams including the words that were not parsed into their stems and endings. However, the bigram probabilities associated with these bigrams had been obtained only using the stems of the words that construct the bigram.

The difference between two language models applied in Experiment 4 and Experiment 5 is the way of applying the bigram probabilities. In Experiment 4, stem and ending of a word are treated as separate words. Remember the known example in this thesis;

Hava lar sıcak

The consequent bigram probabilities are P(lar\hava) and P(sıcak\lar). But in Experiment 5, the sentence takes the form;

Havalar sıcak

and the bigram probability should be P(sıcak\havalar). However, we modify this probability as to be P(sıcak\hava) because of that we obtained the bigram probability

by using only the stems. But the structure of the sentence does not change: Havalar sıcak.

We applied several parameters in Experiment 4 and 5 to find the optimal configuration. The CSR systems built exhibited different degrees of performance. For example, when we modified the back-off probabilities, the system built in Experiment 5 out performed the former one. But, when we left the back-off probabilities unchanged, we saw that the system built in Experiment 4 gave better results. We gave the results of all experiments in tables, too.

First of all, if one inspects the results, it can be said that the CSR systems built in Experiment 4 and 5 are not suitable for real time implementations regarding either the average process time per sentence or the recognition results.

The reasons for requiring rather long time to recognize an utterence are the structure of the network and the size of the vocabulary. We utilized linear lexical search which consumes much effort in a CSR system with a large vocabulary. It is a must to apply a lexical tree search in order to reduce the search effort especially for a vocabulary of size more than 20,000 words.

The acoustic model built is also arguable, although a common known structure is used. Different type of front-end processor parameters can give better results. On the other hand, the poorly balanced phone counts (See Appendix C) may have caused unrobust model parameters, respectively. If the triphone counts are taken into account, the sparsity of the acoustic training data becomes clearer. But as we inspected whether the erroneous results are speaker specific, we found that the case is not so; in this respect, we can say that the speaker independency is achieved.

We applied no smoothing algorithm to the language model built. This can be a handicap to obtain a robust language model. It would be better to apply a smoothing technique such as Katz or Ney-Kneser. But it is worth noting that we tried to compare only the language models, not the smoothing techniques.

In a language model based on only stems, I think that Turkish does not require huge training text corpus, if the subject of the text does not include large amount of proper nouns in its nature. Because, as we can see in Table 5.3, a text of size 434,601 words can reduce to a size 7,974 words if the stems are basic units.

However, the cumulative percentages plotted in Figure 5.5 show that our text corpora used for obtaining the language model suffer from sparsity. Even the smallest percentage value 68.67% is not sufficient to say that the text corpus is balanced.

Comparing the results given in Table 5.10 and Table 5.14, one can see that the ASR system implemented in Experiment 4 performs better than the one implemented in Experiment 5, if the back-off node probabilities are not changed. But if the back-off node probabilities are changed in order to reduce the effect of back-off mechanism, the system implemented in Experiment 5 outperforms the one in Experiment 4. The maximum CSRR (Correct Sentence Recognition Rate) achieved in Experiment 5 is 30.90% whereas the one achieved in Experiment 4 is 20.09%. The situation shows that the back-off mechanism has a greater effect in Experiment 5.

The language model proposed and tested in Experiment 5 is not able to model the endings, allowing only applying the bigram probability over stems. Thus, determination of the endings remains as a task for the acoustic model. This may lead to confusion between words that are inflectional and derivational forms of a stem. To visualize this case, remember the example we gave,

havalar sıcak

The bigram probability P(sıcak\havalar) is made equal to the probability P(sıcak\hava) in our proposition. Expanding this to other examples requires P(sıcak\hava) = P(sıcak\havada) = P(sıcak\havamız), etc. So, we can deduce that better results can be obtained if the proposed language model expanded so that it takes into account the endings. One solution may be building a decoding algorithm that keeps the stem ‘hava’ in mind after it had determined this word as a partial solution, applies the bigram probability P(lar\hava) and then applies the stem-based probability P(sıcak\hava) when entering the word ‘sıcak’. Hence, the accumulated score at the end of the word ‘sıcak’ would include,

P(lar\hava)P(sıcak\hava)

However, determination of the probability P(lar\hava) would require a large text corpus, which is contrary to my idea written in the fifth paragraph of this chapter.

REFERENCES

1. S. E. Levinson. Structural Methods in Automatic Speech Recognition. *Proceedings of the IEEE.* pp. 1625-1649. 1985.
2. F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1998.
3. H. Sakoe. Two-level DP Matching ─A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* pp. 588-595, 1979.
4. C. Myers, L.R. Rabiner. A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* pp. 284-297, 1981.
5. T. K. Vintsyuk. Element-wise Recognition of Continuous Speech Composed of Words From a Specified Dictionary. *Kibernetika*, pp. 133-143. 1971.
6. L.R. Rabiner, B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine.* pp.4-15. 1986.
7. L. R. Rabiner, J. G. Wilpon, F. K. Soong. High Performance Connected Digit Recognition Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech and Signal Processing.* pp. 1214-1225. 1989.
8. K. Çarkı, P. Geutner, T. Schultz. Turkish LVCSR: Towards Better Speech Recognition for Agglutinative Languages. *ICASSP*.pp. 1563-1566. 2000.
9. E. Mengüşoğlu, O. Deroo. Turkish LVCSR: Database Preparation and Language Modeling for an Agglutinative Language. *ICASSP Student Forum.* 2001.
10. C. Yılmaz. *A Large Vocabulary Speech Recognition System forTurkish.* M. Sc. Thesis, Bilkent University, 1999.
11. L. R. Rabiner, B. H. Huang. *Fundamentals of Speech Recognition.* Prentice Hall. 1993.

12. M. K. Ravishankar. *Efficient Algorithms for Speech Recognition.* Ph. D. Thesis. Carnegie Mellon University. 1996.
13. S. Ortmanns. *Effiziente Suchverfahren zur Erkennung Kontinuierlich Gesprochener Sprache.* Ph. D. Thesis. Rheinisch-Westfälischen Hochschule Aachen. 1998.
14. M. Woszczyna. *Fast Speaker Independent Large Vocabulary Continuous Speech Recognition.* Ph. D. Thesis. Universität Karlsruhe. 1998.
15. H. Purnhagen. *N-Best Search Methods Applied to Speech Recognition.* Diploma Thesis. Universitet i Trondheim. 1994.
16. A. Sixtus, H. Ney. From Within-Word Model Search to Accross-Word Model Search in Large Vocabulary Continuous Speech Recognition. *Computer, Speech and Language.* pp. 245-271. 2002.
17. X. Huang, R. Reddy. *Spoken Language Processing.* Pearson Education. 2001
18. S. Ortmanns, H. Ney. The Time-Conditioned Approach in Dynamic Programming Search for LVCSR. *IEEE Transactions on Speech and Audio Processing.* pp. 676-687. 2000.
19. S. Ortmanns, H. Ney, A. Eiden. Language Model Look-Ahead for Large Vocabulary Speech Recognition. *Proceedings of the International Conference on Spoken Language Processing.* pp. 2095-2098. 1996.
20. H. Ney, D. Mergel, A. Noll, A. Päseler. Data Driven Search Organization for Continuous Speech Recognition. *IEEE Transactions on Signal Processing.* pp. 272-281. 1992.
21. S. Renals, M. M. Hochberg. Start-Synchronous Search for Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing.* pp. 542-553. 1999.
22. S. Ortmanns, L. Welling, K. Beulen, F. Wessel, H. Ney. Architecture and Search Organization for Large Vocabulary Continuous Speech Recognition. *27. Jahrestagung der Gesellschaft für Informatik.* 1997.
23. X. L. Aubert. An Overview of Decoding Techniques for Large Vocabulary Continuous Speech Recognition. *Computer, Speech and Language.* pp. 89-114. 2002.

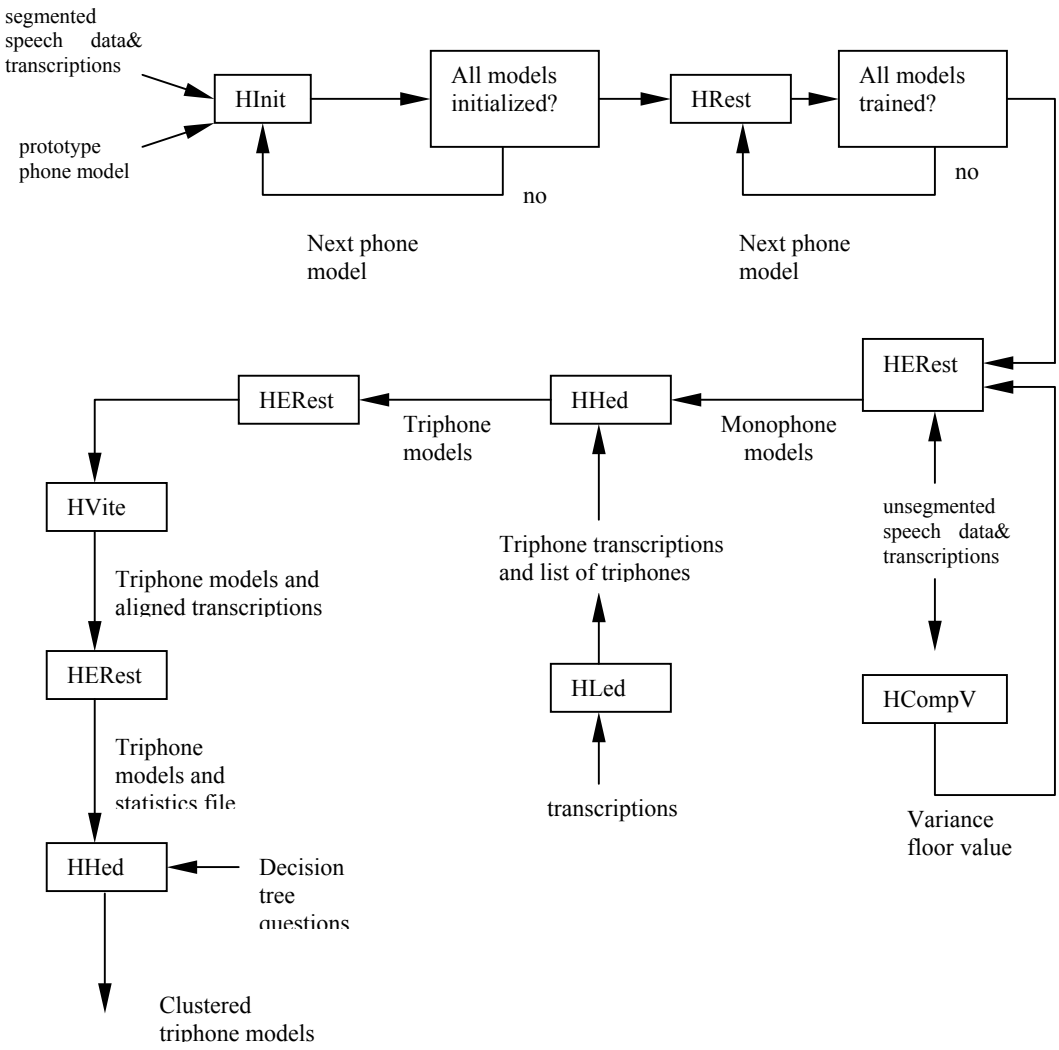
24. J. Gauvain, L. Lamel. Large-Vocabulary Continuous Speech Recognition: Advances and Applications. *Proceedings of the IEEE.* pp. 1181-1199. 2000.
25. H. Ney, S. Ortmanns. Progress in Dynamic Programming Search for LVCSR. *Proceedings of the IEEE.* pp. 1224-1239. 2000.
26. R. Rosenfeld. Two Decades of Statistical Language Modeling: Where Do We Go from Here?. *Proceedings of the IEEE.* pp. 1270-1277. 2000.
27. *Milliyet.* 15 March 2003.
28. K. Oflazer. Two-level Description of Turkish Morphology. *Literary and Linguistic Computing.*pp. 137-148. 1994.
29. Ö. Salor, T. Çiloğlu, M. Demirekler, D. Uluşen, A. Susar. New Corpora and Tools for Turkish Speech Research. *ICSLP.* 2002.
30. H. Dutağacı. *Statistical Language Models for Large Vocabulary Turkish Speech Recognition.* M. Sc. Thesis. Boğaziçi University. 1999.
31. O. Çilingir. *Large Vocabulary Speech Recognition for Turkish.* M. Sc. Thesis. Middle East Technical University. 2003.
32. D. Acar. *Triphone Based Turkish Word Spotting System.* M. Sc. Thesis. Middle East Technical University. 2001.
33. X. Liu, Y. Zhao, X. Pi, L. Liang. Audio-Visual Continuous Speech Recognition Using a Coupled Hidden Markov Model. *ICSLP.* 2002.
34. S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, P. Woodland. *The HTK Book (for HTK Version 3.1).* Cambridge University Engineering Department. 2002.
35. S. Young. Large Vocabulary Continuous Speech Recognition: a Review. *Technical Report.* Cambridge University Engineering Department. 1996.
36. P.C. Woodland, J.J. Odell, V. Valtchev, S.J. Young. Large Vocabulary Continuous Speech Recognition Using HTK. *Proceedings of ICASSP.* 1994
37. L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE.* pp. 257-285. 1989.
38. W. H. Abdulla, N. K. Kasabov. The Concepts of Hidden Markov Model in Speech Recognition. Technical Report. University of Otago. 1999.

39. S. J. Young, N. H. Russell, J. H. S. Thornton. Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems. *Technical Report.* Cambridge University Engineering Department. 1989.
40. C. S. Myers, L. R. Rabiner. Connected Digit Recognition Using a Level Building DTW Algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing.* pp. 351-363. 1981.
41. H. Ney. The Use of One-Stage Dynamic Programming Algorithm for Connected Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing.* pp. 263-271. 1984.
42. Ö. Salor, B. Pellom, T. Çiloğlu, K. Hacioğlu, M. Demirekler. On Developing New Text and Audio Corpora and Speech Recognition Tools for the Turkish Language, ICSLP 2002, Denver Colorado

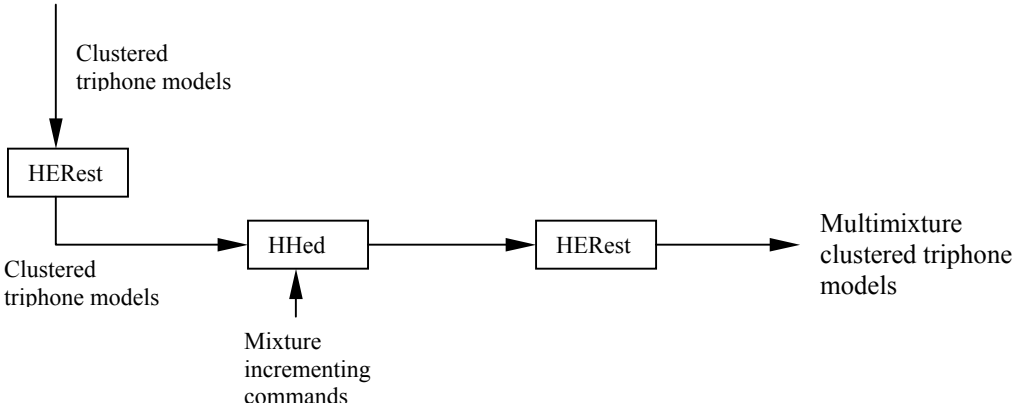
APPENDIX A

TRAINING AND TESTING PHASES WITH HTK

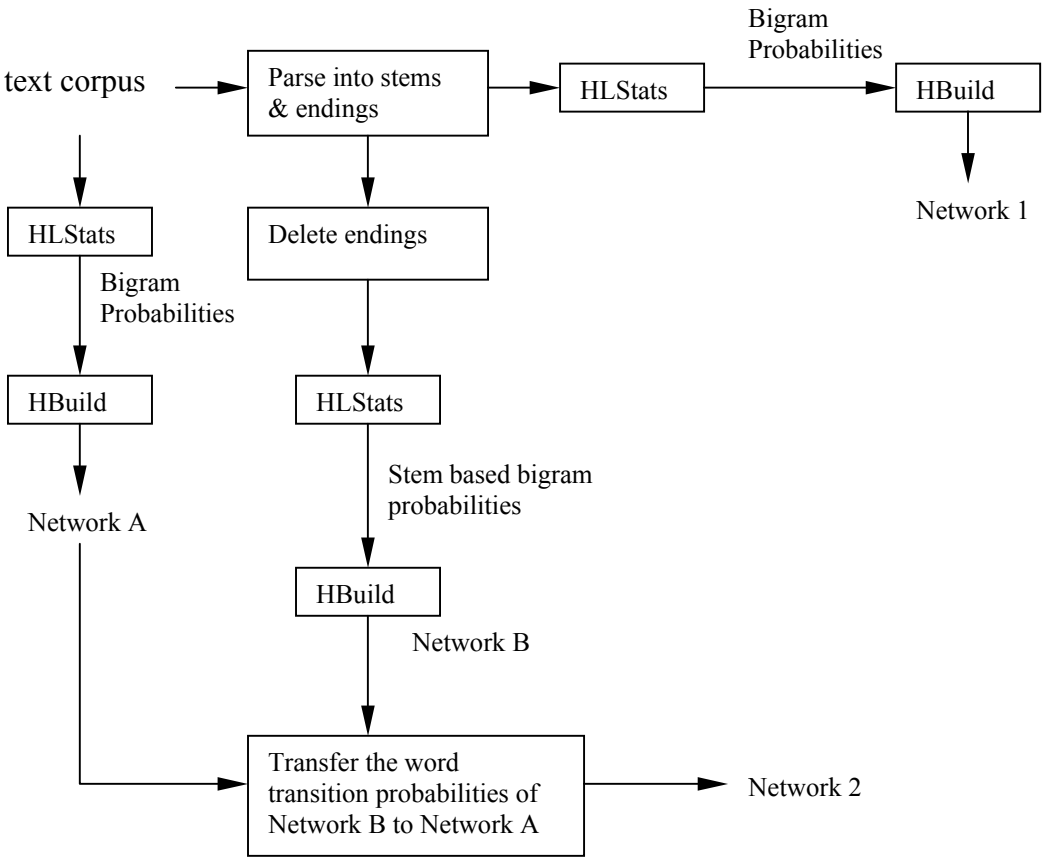
A.1.1 Acoustic Model Training



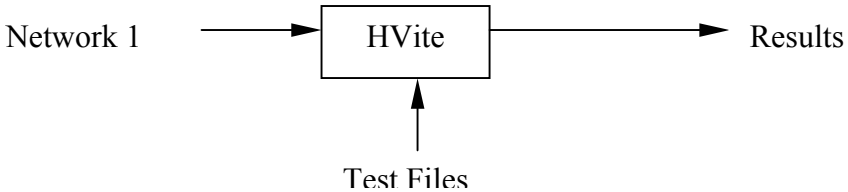
A.1.2 Acoustic Model Training (Continuing)

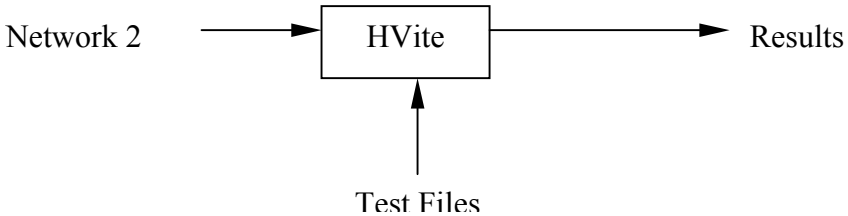


A.2 Language Model Training



A.3 Test


Network 1
HVite
Results
Test Files


Network 2
HVite
Results
Test Files

APPENDIX B

DECISION TREE QUESTIONS

B.1

QS 'L_v_ince_dar' {"i-*","e-*"}
QS 'R_v_ince_dar' {"*+i","*+e"}
QS 'L_v_ince_yuv' {"to-*","tu-*"}
QS 'R_v_ince_yuv' {"*+to","*+tu"}
QS 'L_v_kalin_dar' {"a-*","ti-*"}
QS 'R_v_kalin_dar' {"*+a","*+ti"}
QS 'L_v_kalin_yuv' {"o-*","u-*"}
QS 'R_v_kalin_yuv' {"*+o","*+u"}
QS 'L_nasal' {"n-*","m-*"}
QS 'R_nasal' {"*+n","*+m"}
QS 'L_v_stop' {"c-*","d-*","b-*"}
QS 'R_v_stop' {"*+c","*+d","*+b"}
QS 'L_v_stop1' {"kg-*"}
QS 'R_v_stop1' {"*+kg"}
QS 'L_v_stop2' {"g-*"}
QS 'R_v_stop2' {"*+g"}
QS 'L_unv_stop' {"t-*","p-*","tc-*"}
QS 'R_unv_stop' {"*+t","*+p","*+tc"}
QS 'L_unv_stop1' {"kk-*"}

B.2 (Continuing)

```
QS 'R_unv_stop1' {"*+kk"}
QS 'L_unv_stop2' {"k-*"}
QS 'R_unv_stop2' {"*+k"}
QS 'L_v_fric' {"z-*","v-*","j-*"}
QS 'R_v_fric' {"*+z","*+v","*+j"}
QS 'L_unv_fric' {"f-*","s-*","ts-*"}
QS 'R_unv_fric' {"*+f","*+s","*+ts"}
QS 'L_fisil' {"h-*"}
QS 'R_fisil' {"*+h"}
QS 'L_sessizlik' {"ccc-*","ppp-*"}
QS 'R_sessizlik' {"*+ccc","*+ppp"}
QS 'L_yum_g' {"tg-*"}
QS 'R_yum_g' {"*+tg"}
QS 'L_y' {"y-*"}
QS 'R_y' {"*+y"}
QS 'L_diger0' {"l-*"}
QS 'R_diger0' {"*+l"}
QS 'L_diger2' {"kl-*"}
QS 'R_diger2' {"*+kl"}
QS 'L_diger1' {"r-*"}
QS 'R_diger1' {"*+r"}
```

APPENDIX C

PHONE OBSERVATION COUNTS IN ACOUSTIC MODEL TRAINING

C.1

	Phone model	SAMPA definition	count
1	a	a	37657
2	b	b	8245
3	c	dZ	3263
4	d	d	13974
5	e	e	31016
6	f	f	1837
7	g	gj	3616
8	h	h	3620
9	i	i	28747
10	j	Z	235
11	k	c	7003
12	l	l	10352
13	m	m	10906
14	n	N	21963
15	o	o	9181

C.2 (Continuing)

16	p	p	3036
17	r	r	24362
18	s	s	9737
19	t	t	11930
20	u	u	10177
21	ppp	Short pause	52024
22	v	v	3866
23	y	j	11099
24	z	z	5422
25	kg	g	746
26	kk	k	9003
27	kl	5	11282
28	tc	tS	4515
29	tg	:	3404
30	ti	1	16074
31	to	2	2675
32	ts	S	5609
33	tu	y	6567
34	ccc	silence	7718