Toward More Effective Acoustic Model Clustering by More Efficient Use of Data in Speech Recognition

Chaojun Liu
B.E., Univ. of Science & Technology of China, 1994
M.S., Institute of Acoustics, Chinese Academy of Science, 1997

A dissertation submitted to the faculty of the
OGI School of Science & Engineering
at Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

June 2002

The dissertation "Toward More Effective Acoustic Model Clustering by More Efficient Use of Data in Speech Recognition" by Chaojun Liu has been examined and approved by the following Examination Committee:

Dr. Yonghong Yan
Associate Professor
Thesis Research Advisor

Dr. Peter A. Heeman
Assistant Professor

Dr. Hynek Hermansky
Professor

Dr. Kevin S. Van Horn
Assistant Professor
North Dakota State University

Dedication

To my parents and my wife.

Acknowledgements

First and foremost, I would like to express my greatest gratitude to my advisor Dr. Yonghong Yan for his invaluable advices and support throughout the past nearly five years. It is he that led me into the real research area of speech recognition. Without his supervision and guidance, I would not have been able to complete this thesis. His insight, perspective, and vision as well as his diligence and perseverance are the continuous subject of my admiration.

Thanks are also due to Dr. Peter Heeman who gave me tremendous help, especially during the time when my advisor was at Intel China Research Center. Discussing with him was pleasant and fruitful. His advices helped me improve my research skills such as writing and presentation skills.

I also would like to thank my thesis committee members, Dr. Yonghong Yan, Dr. Peter Heeman, Dr. Hynek Hermansky, and Dr. Kevin van Horn, for reviewing my thesis and giving me precious feedbacks to improve my thesis.

My appreciation goes to the large vocabulary continuous speech recognition group members, former and present: Dr. Yonghong Yan, Dr. ChangHyuck Oh, DongHwa Kim, Dr. Xintian Wu and Chengyi Zheng. Without the collective hard work, I would not have a competitive baseline system to finish my experiments.

I also would like to thank the staff and students in the Center for Spoken Language Understanding (CSLU) for their help and support, which made my life much easier. It was a pleasant experience to study at CSLU.

Last but not least, I want to give special thanks to my wife for her continuous support through the past years. Without her encouragement, I might have already quit the program three years ago.

Contents

List of Tables

List of Figures

Abstract

Toward More Effective Acoustic Model Clustering by More Efficient Use of Data in Speech Recognition

Chaojun Liu

Ph.D., OGI School of Science & Engineering
at Oregon Health & Science University
June 2002

Thesis Advisor: Dr. Yonghong Yan

In current large vocabulary continuous speech recognition systems, multivariate Gaussian mixture distributions and context-dependent phones, typically triphones, are used to achieve high accuracy acoustic models. It is crucial to address the problem of how to estimate an extremely large number of model parameters from a limited amount of training data. The traditional approach uses phonetic decision tree based context clustering for reducing free parameters. However, this approach has several problems that might cause system performance degradation. All of these problems are due to the fact that the traditional approach does not efficiently use the limited training data and therefore fails to obtain effective acoustic models. Specifically, three problems are identified and addressed. The first problem is that all states clustered in a leaf node must share the same set of Gaussian components and mixture weights; no distinction is provided among those states. The second problem is due to the fact that triphones that are rarely seen in the training data might be poorly estimated and this causes an adverse effect on decision-tree clustering. The traditional approach lacks an effective mechanism to handle this. The third problem is that only single-Gaussian distributions are used to build decision trees whereas multiple-Gaussian mixture distributions are used in the final model set.

In this thesis, we propose to improve the quality of acoustic models by making use of training data more efficiently. We present a number of ways to address the problems in the traditional approach, namely, a two-level decision tree approach for the first problem, a two-stage decision tree based approach and a MAP-based approach for the second problem, and an approach using a new criterion and an effective clustering algorithm for the third problem. Each of these approaches has successfully reduced the word error rate (WER) of the traditional approach with a statistical significance. Finally the system combining all new approaches has achieved the best performance, which reduced the WER of the baseline system by 14% to 17% relative, with the sizes of acoustic models smaller than those of the baseline models by 8% to 11%.

Chapter 1

Introduction

Research on automatic recognition of speech by machines originated in the early 1950's and has been a continuous effort for the past fifty years due to its widespread use in a variety of applications including information retrieval, data entry, and general man-machine communication. Over the years, a number of general approaches have been proposed for speech recognition, among which the statistical pattern recognition approach has become the dominant one due to its simplicity of use, robustness and capability to model the variation in speech, and its proven high performance. Basically the approach has two steps, namely, training of speech patterns, and recognition of patterns via pattern comparison. Speech knowledge is brought into the system via the training procedure on the training data. Generally speaking, the more data we have, the more useful information the system will learn and therefore the better performance.

However, two factors prevent us from collecting "sufficient" training data. One is that collecting and transcribing data are expensive. The other is that although there are many existing different types of speech corpora available collected over the years, pooling data from these different sources may not help and sometimes may even hurt the system performance for a given task. This is because the variation in the pooled speech data will be dramatically increased and the mismatch between the training data and test data will be large. Current speech recognition algorithms are generally not able to effectively get rid of the irrelevant information and extract only the information useful for recognition. So usually for a specific task in a given domain and environment, we need to collect training data in the same or similar domain and environment as in the test environment. This limitation also prevents us from collecting enough training data. So, given limited training data, a critical issue is how to effectively use the data, that is, how can we improve the training algorithm to extract as much useful information as possible from the training data. This is the theme that we explore in this thesis.

1.1 Speech Recognition System Overview

Most current speech recognition systems are firmly based on the principles of statistical pattern recognition. Figure 1.1 illustrates the main components of a typical speech recognition system.

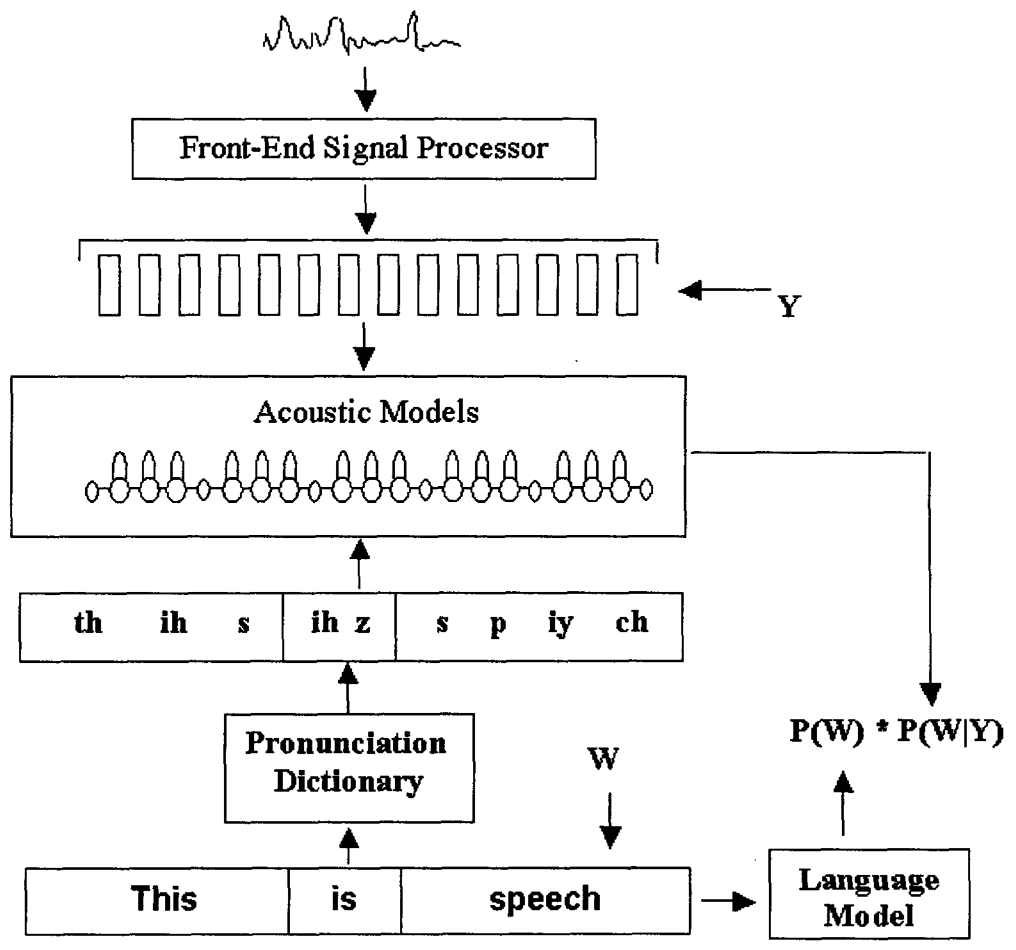


Figure 1.1: Overview of statistical speech recognition (after [59]).

The front-end signal processor converts an unknown speech waveform into a sequence of acoustic vectors, $O = o_1, o_2, \ldots, o_T$. Each of these vectors is a compact representation of the short-time speech spectrum covering a period of typically 10 milliseconds. Let W be a sequence of words $w_1, w_2, \ldots, w_n$. The speech recognition system will determine the most likely word sequence given the observed acoustic signal Y. According to Bayes' rule, we get

$$\hat{W} = \arg\max_W P(W|O) = \arg\max_W \frac{P(W)P(O|W)}{P(O)} = \arg\max_W P(W)P(O|W)$$

This equation indicates that the most likely word sequence $\hat{W}$ is the one that maximizes the product of $P(W)$ and $P(O|W)$. The first term represents the *a priori* probability of observing W independent of the observed signal, and this probability is determined by a language model. The second term represents the probability of observing the vector sequence O given some specified word sequence W, and this probability is determined by an acoustic model. As shown in Figure 1.1, a word sequence W = "*This is speech*" is hypothesized and the language model computes its probability $P(W)$; each word is then converted into a sequence of phones using a pronunciation dictionary. For each phone there is a corresponding statistical model called a *hidden Markov model* (HMM). The HMMs corresponding to the phones in the utterance are concatenated to form a single composite model, and the probability of that model generating the observed sequence O is calculated. This is the required probability $P(O|W)$. In principle, this process can be repeated for all possible word sequences, and the most likely sequence is selected as the system output. Since computing all possible word sequences is too computationally expensive to be practical, much more efficient algorithms are used to find the optimal word sequence.

1.2 Acoustic Modeling

The purpose of the acoustic model is to calculate the likelihood of any vector sequence O given a word w. In principle, the required probability distribution could be determined by finding many examples of each word w and collecting the statistics of the corresponding vector sequences. However, when the vocabulary size becomes large, it is impractical to collect sufficient training data. So words are chosen to be the basic model unit only in applications with very small vocabulary size, such as command-and-control or digit recognition. For systems with medium to large size vocabulary, sub-word units such as syllables, demi-syllables and phones are used. By concatenating models of sub-word units, word models can be constructed. Two criteria need to be considered in choosing model units: (1) consistency – different instances of a unit have similar characteristics, and (2) trainability – sufficient training samples exist to create a robust model. Ever since phone-based models were introduced, they have become the most popular choice of model units, especially in large vocabulary continuous speech recognition systems. Each phone is represented by a HMM with Gaussian mixture output distributions (see Chapter 2 for details). There are about 50 phones in English.

Because speech is produced by physical articulators, which do not undergo drastic or sudden movement, the acoustic realization of a particular phone is heavily influenced by the preceding and following positions of the articulators. This effect is called *coarticulation*. Coarticulation means that the acoustic realization of a phone in a particular phonetic context is more consistent than the phone occurring in a variety of contexts. Hence, to achieve good phonetic discrimination, different HMMs have to be trained for each different context. This is called context-dependent phone modeling. The simplest and most commonly used context-dependent phone is triphone [49]. A triphone is a single phone but in the context of a unique pair of left and right neighboring phones. In contrast, the context-independent phone is called a monophone. For example, suppose that the notation x-y+z represents the phone /y/ occurring after an /x/ and before a /z/. The sentence, "this is speech", as shown in Figure 1.1, would be represented by the phone sequence "sil th ih s ih z s p iy ch sil". The corresponding triphone[1] sequence would be "sil sil-th+ih th-ih+s ih-s+ih s-ih+z ih-z+s z-s+p s-p+iy p-iy+ch iy-ch+sil sil". Note that the two instances of phone /ih/ are represented by different triphones and therefore different models.

[1]Actually, cross-word triphone sequences are used since the triphone contexts span word boundaries. /sil/ is a context-independent model for silence.

The use of triphones and Gaussian mixture output distributions greatly improves the accuracy of acoustic models. However, it results in an extremely large number of parameters in a system. For a typical set of 50 phones, there are $50 * 50 * 50$ possible triphones although some of them cannot occur due to phonological reasons. In the DARPA Wall Street Journal (WSJ) 20K vocabulary task, a standard continuous speech recognition task, the training data set SI284 has about 28,000 cross-word triphones (the other triphones, which are not seen in the training data, are called *unseen* triphones). Assuming that the feature vector has 39 elements and each HMM state has 10 Gaussian components, which are typical settings for a system to obtain reasonable good recognition accuracy, a typical 3-state HMM phone model will have about 2370 parameters[2]. Hence, 28,000 triphones would have a total of 66 million parameters! Obviously large amounts of training data are needed to estimate so many parameters.

It is crucial to address this problem of too many parameters and too little available training data in the design of a statistical speech recognition system. Over the years, many approaches have been proposed, such as tied-mixture [8] or semi-continuous HMM [23], generalized triphone [34], state-based tying [39] [61] and phone-based component tying [16]. Among these studies, the approach of state-tying using phonetic decision trees [61], proposed by the HTK group in Cambridge University, is probably the most popular one, partially due to their success in a series of DARPA evaluations and partially due to the simplicity and efficiency in its implementation.

1.3 HMM State Clustering

State clustering is the practice of clustering states that are acoustically very similar. This allows all the data associated with similar states to be pooled, and thereby gives more robust estimates for the parameters of the clustered state. There are two major categories of approaches to clustering: bottom-up approaches and top-down approaches. The top-down approach using phonetic decision trees has the advantage of predicting unseen

[2]Every Gaussian component has 39 means, 39 covariances (assuming that its covariance matrix is diagonal) plus 1 mixture weight. So a 3-state HMM model has $3 * 10 * (39 + 39 + 1) = 2370$ parameters.

triphones, and hence has become more popular than bottom-up approaches, especially in large vocabulary continuous speech recognition tasks where cross-word triphones are used and therefore a significant portion of triphones are unseen in the training data.

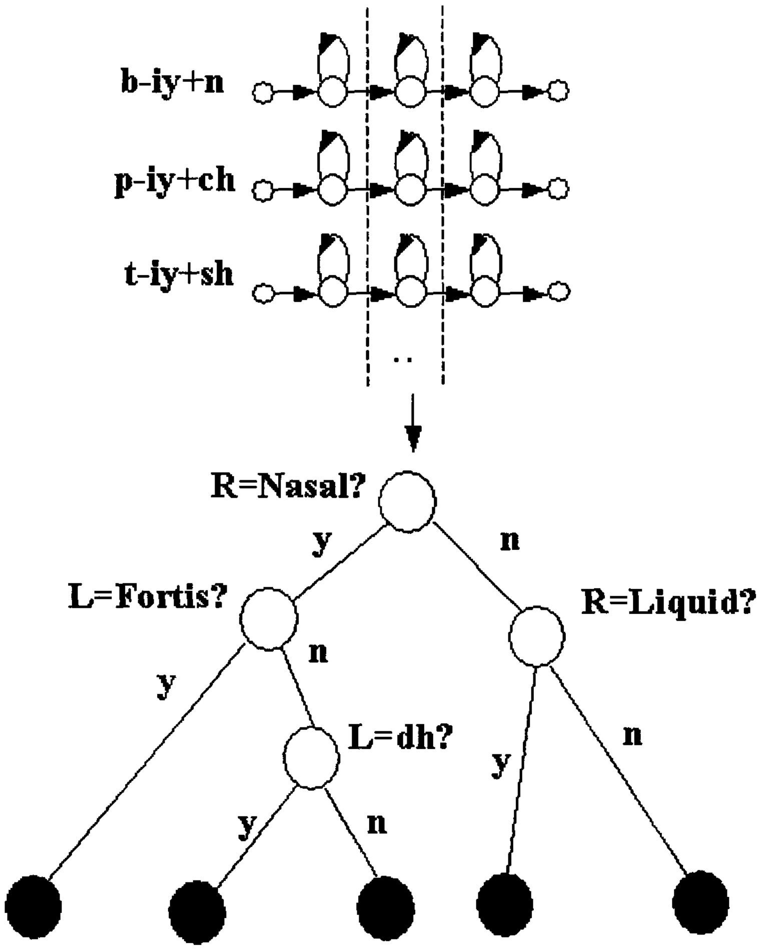


Figure 1.2: Phonetic decision-tree based clustering.

The decision-tree based clustering approach has been widely used in state-of-the-art systems, such as HTK [54] from Cambridge University and IBM's system [45], which gave the best performance in several DARPA evaluations in the past few years. Figure 1.2 illustrates an example of clustering the center states of all triphones that are derived from monophone /iy/, using a phonetic decision tree.

Decision tree clustering involves building a binary tree for each state of each phone.

Each node (except leaf node) of the tree has a yes-no phonetic question such as "R=Nasal?", which means "Is the right phone to the current phone a nasal?". Initially all states for a given phone state position are placed at the root node of a tree. Depending on the answer to the question, the pool of states is split into two subsets and this procedure repeats until the states trickle down to leaf nodes. All states in the same leaf node are then clustered. The question at each node is chosen to maximize the likelihood of the training data given the resultant sets of clustered states. To ensure that each state can be robustly estimated, each leaf node in a decision tree is required to have some minimum amount of training data. All the data in a leaf node are used to estimate a multiple-Gaussian mixture distribution, and all the states clustered in that leaf node share the same distribution.

Note that with this approach, the unclustered models used for building decision trees are single-mixture models, and after the trees are built, the number of Gaussian components in each state cluster (leaf node) is increased until a desired number is reached. Section 2.3 gives more details on building acoustic models using phonetic decision trees.

The decision trees give a clustering of the triphone states and this clustering will heavily influence all the subsequent training steps, and thereby the quality of the final acoustic model. During decoding, whether or not a particular triphone has appeared in the training data, a model for each state of the triphone can be found by traversing the corresponding decision tree to a leaf node by answering the questions along the path.

1.4 Problems in Traditional Decision-Tree Based Acoustic Modeling

Although the decision-tree based acoustic modeling is successful and widely used, it fail to take full advantage of the limited training data, and this might cause degradation of system performance. Some examples are as follows.

First, in the traditional approach, all the data in each leaf node are used to estimate a Gaussian mixture distribution, and all the member states share the same set of Gaussian components and their mixture weights, that is, there is no distinction among those member states. To ensure that those Gaussian components can be robustly estimated, a relatively high threshold of data count must be used for each leaf node. This constraint might cause an accuracy problem, especially for those triphones that do not have much training data. They might be grouped together not because they are acoustically similar, but because they do not have enough training data. As a result, this constraint can be a cause of performance degradation.

Second, the training data are usually distributed unevenly in terms of the occurrence of triphones. For example, in the SI284 training set, 10% of the number of distinct triphones only occur once. For triphones that are rarely seen in the training data, the estimated Gaussian distributions for them might not be robust. When they directly participate in the tree construction, the errors in estimating the statistics of these rarely seen triphones can cause a long-term adverse effect on the quality of the decision-tree based state clustering. How to better cluster these rarely seen triphones and how to minimize the negative effects of the non-robustly estimated statistics of these triphones for decision tree building is very important. The traditional approach uses a simple back-off mechanism to prevent the non-robustness in the estimation of rarely seen triphones. If the amount of training data in a triphone state is less than a certain threshold, it will be backed-off to the corresponding HMM state of the monophone model. This means that the data for the rarely seen triphones are wasted and the information is lost. This might not be an efficient way to use the limited training data.

Third, in the traditional approach, the parametric form of the initial unclustered triphone states must be based on only single Gaussian distributions, although more accurate multiple-Gaussian mixture distributions are used in the final model set. This disparity is due in part to the computational complexity in the tree-building process. The multiple-Gaussian mixture distribution for each tree node would need to be re-estimated from the training data, whereas the statistics for the single-Gaussian case can be calculated efficiently from the cluster-member statistics without re-accessing the original training data. However, this constraint prevents the decision tree building from using the data effectively. Instead, it uses the data in a wasteful manner, throwing away a large amount of information in the data, particularly for the triphones that have many training samples. The single Gaussian distribution is a very crude representation of the acoustic space of triphone states, and decision trees based on such initial models might not give good clustering of triphone states.

1.5 This Thesis in Perspective

All these problems are due to the fact that the traditional approach does not make use of the limited training data in an efficient manner, and that this is one of the major causes of performance degradation. In this thesis, we propose a number of ways to improve the quality of acoustic models by making use of training data more efficiently.

For the first problem of using the same set of data to estimate both Gaussians and their mixture weights, we propose an approach using what we called two-level decision trees [37]. The key idea is to de-couple the Gaussian components and their mixture weights. As estimating a large multivariate Gaussian distribution needs much more data than estimating a mixture weight (given the Gaussian set), we can use less data to estimate the mixture weights for a multiple-Gaussian mixture model. In other words, given the same amount of data (in a cluster of states), we might be able to distinguish those states by estimating different weights for them given the same set of Gaussian distribution. In this way, we can make more efficient use of the limited data and obtain more accurate acoustic models. To apply this idea, we change the structure of conventional trees into two levels, as shown in Figure 1.3. In every first-level leaf node, a Gaussian mixture distribution is estimated. In every second-level leaf node, only a set of mixture weights (corresponding to the set of Gaussian components estimated in its first-level ancestor nodes) are estimated.

For the second problem of waste of data for the rarely seen triphones, we propose two approaches. They fall into two categories: parameter sharing and parameter smoothing, which are two common principles of parameter estimation to handle the problem of data sparseness in statistical speech recognition.

1. Two-stage decision tree building approach. This approach involves building decision trees in two stages. In the first stage, we build very big decision trees to cluster the rarely seen triphone states with their most similar peers. In the second stage, the statistics for a rarely seen triphone state is smoothed with the statistics for other

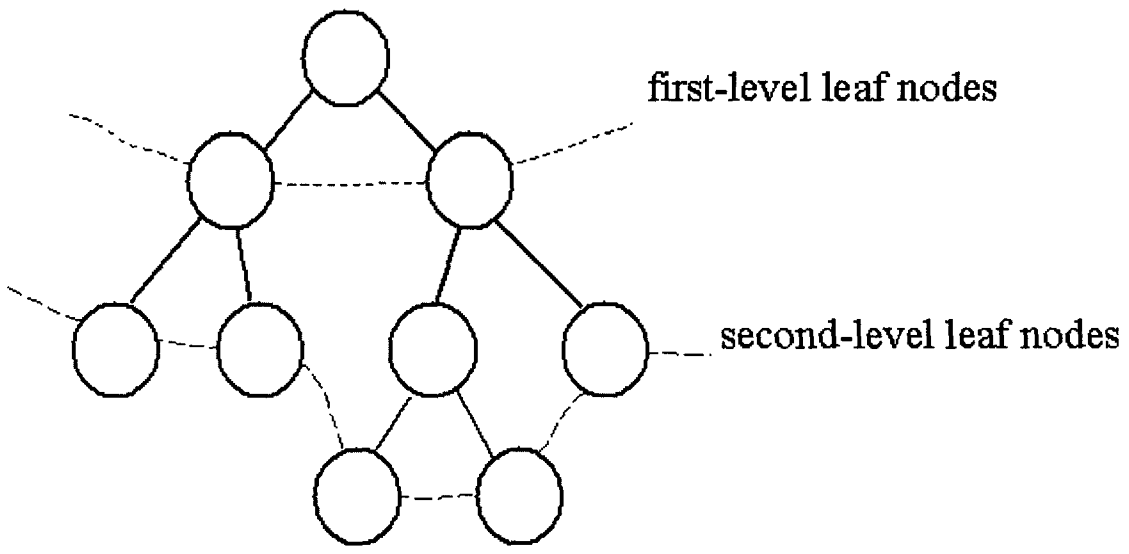


Figure 1.3: A two-level phonetic decision tree. Gaussian mixture distributions are estimated in the first-level leaf nodes; Mixture weights corresponding to the Gaussian components in their first-level ancestor nodes are estimated in the second-level leaf nodes.

triphone states that are in the same cluster. Therefore, the new statistics for the rarely seen triphone state is more robust and the information in its limited training data is also used. But the statistics for the frequently-seen triphone states are kept unchanged because the amount of training data associated with them warrants robust estimates of their statistics. Then, final phonetic decision trees are built, using these statistics, to determine the final clustering of all triphone states. Compared to other approaches like the one in [47], our approach uses both phonetic knowledge and underlying data information in the initial clustering of rarely seen triphone states. It may be more accurate and flexible.

2. Maximum A Posteriori (MAP) based parameter smoothing. MAP [19] is a widely known and deployed technique to estimate HMM parameters. In contrast to the commonly used Maximum Likelihood (ML) estimate, MAP incorporates prior knowledge of parameter values in estimating them. We propose to use MAP to obtain robust statistics of context-dependent models. It can be interpreted as a weighted average

of context-independent model parameters, which is more robust and less accurate, and unsmoothed context-dependent model parameters, which is more accurate and less robust.

Both approaches yield a small, but consistent and statistically significant improvement over the traditional approach.

For the third problem of using only single Gaussian models to build decision trees and therefore wasting information in the data for those frequently-seen triphones, we propose a new criterion to approximate the total likelihood of training data given a set of multiple-Gaussian mixture models. When applied to the decision tree building process, an effective algorithm is needed to estimate a multiple-Gaussian mixture distribution from its member states, each of which is represented by a multiple-Gaussian mixture distribution. Theoretically, original training data need to be re-accessed because sufficient statistics are not available. However, we circumvent this difficulty by using an efficient clustering algorithm to directly approximate the true distribution of each node, when an appropriate assumption is made. Experimental results confirm the effectiveness of this approach.

Finally, we combine all our approaches into one system and obtain the best performance, which reduces the word error rate (WER) of the baseline system by 14% to 17% relative[3], on a variety of test sets. It shows that each of these methods targets a different aspect of the traditional decision tree based acoustic model. Although their contributions overlap to some degree, in general, the combined system is better than each separate one.

This thesis will be organized as follows.

- Chapter 2 will give some background knowledge in speech recognition. These include HMM technology, definition of phonetic decision trees and some details in using phonetic decision trees to build acoustic models.
- Chapter 3 will give the experimental environment including speech databases and evaluation tasks, in which all our proposed approaches will be evaluated. We will also briefly explain our large vocabulary continuous speech recognition system. A

[3]When we say we have reduced the WER by N%, we mean N% relative, unless stated otherwise.

new training strategy will be introduced and its impact to our system performance will be demonstrated. The improved system will serve as our baseline to compare with other systems that built using our new approaches in the following chapters.

- In Chapter 4, 5 and 6, we will present in depths our new approaches to address each of the three problems in the traditional approach. Specifically, Chapter 4 will present the two-level phonetic decision tree approach to address the first problem. Chapter 5 will cover the two-stage decision tree building approach and MAP-based smoothing approach to address the second problem in the traditional approach. Chapter 6 will give the details of building phonetic decision trees using multiple-Gaussian mixture models. Experimental results will also be given in each chapter. Readers can directly jump to a particular chapter that is of interest to them. At the end of Chapter 6, systems combining two or more approaches are built and the best results are given.

- Chapter 7 will summarize this thesis work and point out our future research directions.

Chapter 2

Background Technology and Related Work

This chapter gives the background technology used in this thesis. Section 2.1 and 2.2 introduces the concept of hidden Markov model and its mathematical solutions for application in speech recognition. Section 2.3 describes the phonetic decision trees and the traditional acoustic modeling approach using phonetic decision tree based state clustering. The goodness-of-split criterion used in the traditional approach is given. Some related work is also briefly explained.

2.1 Hidden Markov Models (HMM)

Hidden Markov models are probably the most widely used and successful statistical method in speech recognition. The underlying assumption of HMM based speech recognition is that the speech signal can be well characterized as a parametric random process, and the parameters of the stochastic process can be determined in a precise and well-defined manner. The success of HMM for speech recognition is mostly due to the existence of computationally efficient algorithms for estimating the model parameters given example observation sequences of known class, which is called training, and for choosing which model best matches an observation sequence of unknown class, which is called decoding. The basic theory of hidden Markov models was proposed by Baum and his colleagues [7] [6] in late 1960s and early 1970s. It was applied to speech processing by Baker [5] at CMU, and by Jelinek and his colleagues [25] [4] at IBM in the 1970s.

An HMM is a stochastic process model that has a number of states connected by arcs. It can be viewed as a finite state machine, which changes state once every time unit. At each time t that a state j is entered, an acoustic speech vector o_t is generated with probability density $b_j(o_t)$. The transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}. The joint probability of an acoustic vector sequence $O = o_1, o_2, \ldots, o_T$ and some state sequence $S = s_1, s_2, \ldots, s_T$, given some model λ, is

$$P(O, S|\lambda) = a_{s_0 s_1} \prod_{t=1}^{T} b_{s_t}(o_t) a_{s_t s_{t+1}} \tag{2.1}$$

where s_0 is constrained to be the entry state and s_{T+1} is constrained to be the exit state and $a_{s_T s_{T+1}} = 1$. In practice, only the observation sequence O and model λ are known and the underlying state sequence S is unknown, or "hidden". This is why it is called a *hidden* Markov model.

Figure 2.1 illustrates an HMM-based phone model. Typically, a phone model has three emitting states and a simple left-to-right topology. The entry and exit states, which are non-emitting states, are provided to make it easy to join models together. The self-transition probabilities model the duration variability in real speech and the output probability models the spectral variability.

The choice of the parametric form for the output probability distribution is crucial since it must model all of the intrinsic spectral variability in speech, both within and across speakers. Early systems used discrete HMM models, in which each state has a discrete output probability function, in conjunction with a vector quantizer. Each input acoustic vector was replaced by the index of the closest vector in a pre-computed codebook, and the output probability functions were just look-up tables containing the probabilities of each possible vector quantization (VQ) index. This approach is computationally very efficient, but the VQ introduces distortion, which limits the precision of acoustic models. Hence, modern systems use *continuous density* HMMs (CDHMM), which have a parametric continuous density output distribution for each state to model the acoustic vectors directly. The most common choice of distribution is the multivariate Gaussian mixture:

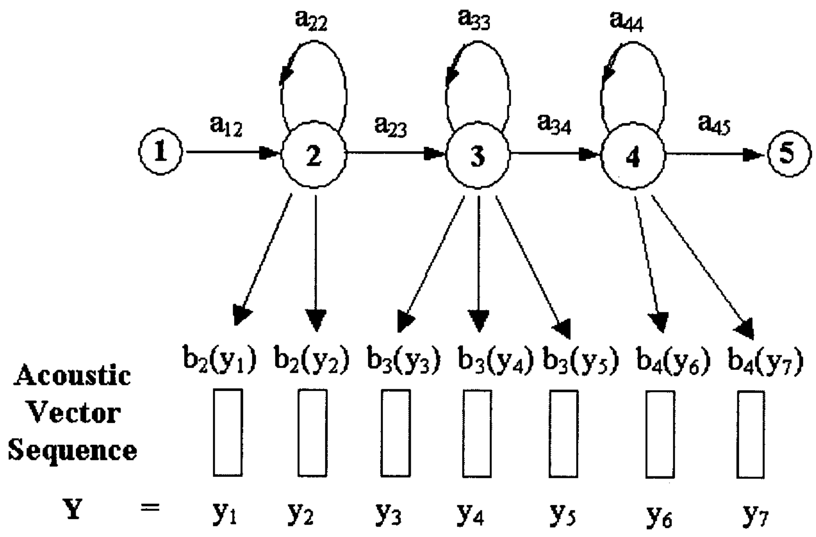


Figure 2.1: An HMM-based phone model.

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(o_t; \mu_{jm}, \Sigma_{jm})$$

where c_{jm} is the weight of mixture component m in state j and N denotes a multivariate Gaussian of mean μ and covariance Σ.

2.2 Three Problems with HMM

To make HMM useful for speech recognition, there are three problems to be solved [46]:

1. Given an observation sequence $O = (o_1 o_2 \ldots o_T)$ and a model λ, how to efficiently compute the probability of the observation sequence, $P(O|\lambda)$, given the model?

2. Given an observation sequence $O(o_1 o_2 \ldots o_T)$ and a model λ, how to determine a corresponding state sequence that can best explain the observation?

3. Given an observation sequence $O(o_1 o_2 \ldots o_T)$, how to estimate the parameters of a HMM model λ to maximize $P(O|\lambda)$?

2.2.1 Evaluation: the forward or backward algorithm

The first problem is an evaluation problem. Given an HMM represented by λ and an observation sequence $O = (o_1 o_2 \ldots o_T)$, the occurrence probability of the observation $P(O|\lambda)$ can be theoretically calculated by summing Equation 2.1 over all possible state sequences. Let $\pi_{s_1} = a_{s_0 s_1}$ be the initial probability of state s_1.

$$\begin{aligned} P(O|\lambda) &= \sum_{allS} P(O|S,\lambda)P(S|\lambda) \\ &= \sum_{allS} \pi_{s_1} b_{s_1}(o_1) a_{s_1 s_2} b_{s_2}(o_2) \ldots b_{s_T}(o_T) \end{aligned}$$

The computational complexity of this calculation is $O(TN^T)$, based on N^T possible state sequences with $2T$ terms in each product. To make this computation tractable, either a forward algorithm or backward algorithm can be used.

The forward variable $\alpha_i(t)$ is defined as

$$\alpha_i(t) = P(o_1 o_2 \ldots o_t, s_t = i|\lambda)$$

which is the joint conditional probability of the partial observation sequence, $o_1 o_2 \ldots o_t$ and state i at time t, given the model λ. Initializing at time $t = 1$, the forward variable can be computed inductively using the following steps:

1. **Initialization**

$$\alpha_i(1) = \pi_i b_i(o_1) \qquad 1 \leq i \leq N$$

2. **Recursion**

$$\alpha_j(t+1) = \Big[\sum_{i=1}^{N} \alpha_i(t) a_{ij}\Big] b_j(o_{t+1}) \qquad 1 \leq t \leq T-1,\ 1 \leq j \leq N$$

3. **Termination**

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

The backward variable $\beta_i(t)$ is defined as

$$\beta_i(t) = P(o_{t+1}o_{t+2}\dots o_T|s_t = i, \lambda)$$

which is the conditional probability of the partial observation sequence from time $t+1$ on, given both the model and known state occupancy in state i at time t. In the same way, we can initialize the backward variable at time $t = T$ and compute it inductively using the following steps:

1. **Initialization**

$$\beta_i(T) = 1 \qquad 1 \le i \le N$$

2. **Recursion**

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij}b_j(o_{t+1})\beta_j(t+1) \qquad 1 \le i \le N \ , \ t = T-1, T-2, \dots, 1$$

3. **Termination**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1)\beta_j(1)$$

2.2.2 Alignment: the Viterbi algorithm

The second problem is referred to as an alignment problem. To find the best state sequence for a given observation sequence, the Viterbi algorithm [52], a dynamic programming method, is used. If we define the quantity $\phi_i(t)$ as the partial state sequence probability

$$\phi_i(t) = \max_{s_1\dots s_{t-1}} P(s_1\dots s_{t-1}, s_t = i, o_1\dots o_t|\lambda)$$

then the Viterbi algorithm can be simply stated as

1. **Initialization**

$$\phi_i(1) = \pi_i b_i(o_1) \quad 1 \le i \le N$$

2. **Recursion**

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

3. **Termination**

$$P_{max}(S,O|\lambda) = \max_{1\le i\le N} \Big[\phi_i(T)\Big]$$

The final result of the algorithm is P_{max}, the probability of the most likely state sequence. The identity of the individual states within the sequence can be obtained by recording the *arg max(i)* at each step of the Viterbi recursion and backtracking after the final result is found. This algorithm will be used in the bootstrapping stage of the whole training process (see Chapter 3), in which we iteratively align the training utterances and update the model parameters using data aligned to each specific HMM state.

2.2.3 Training: the Baum-Welch re-estimation algorithm

The third problem given in Section 2.2 is referred to as a training problem, in which we need to estimate the parameter set $\lambda = (A, B, \pi)$ in such a way that the probability of the observation set $P(O|\lambda)$ is maximized over all training data and models. There is no known way to analytically solve this problem. However, we can choose $\lambda = (A, B, \pi)$ such that its likelihood, $P(O|\lambda)$, is locally maximized using an iterative procedure such as the Baum-Welch algorithm [6]. The Baum-Welch algorithm is an efficient algorithm to provide the Maximum Likelihood (ML) estimation of HMM parameters based on *Expectation Maximization* (EM) algorithm [15]. For the case of discrete HMM, the re-estimation formula are straightforward and can be summarized as follows:

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

$$\bar{b}_j(o_t) = \frac{expected\ number\ of\ times\ observing\ o_t\ from\ state\ j}{expected\ number\ of\ times\ in\ state\ j}$$

where the expectations on the right are determined using the current values of a_{ij} and $b_j(o_t)$. To compute these, we first need to define a variable, $\varepsilon_{ij}(t)$, the probability of being in state i at time t and state j at time $t+1$, given the model and the observation sequence, that is,

$$\varepsilon_{ij}(t) = P(s_t = i, s_{t+1} = j | O, \lambda)$$

Using the forward and backward variables, we can rewrite it in the form

$$\varepsilon_{ij}(t) = \frac{P(s_t = i, s_{t+1} = j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{P(O|\lambda)}$$

We refer to it as two-state occupancy probability. We also need to define the one-state occupancy probability,

$$\gamma_i(t) = P(s_t = i | O, \lambda)$$

Similarly, we can rewrite it as

$$\gamma_i(t) = \frac{P(s_t = i, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_i(t)\beta_i(t)}{P(O|\lambda)}$$

Put together, the Baum-Welch re-estimation formula is

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{2.2}$$

In the case of continuous HMM, where the state output distribution takes the form of Gaussian mixture model (GMM),

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(o_t; \mu_{jm}, \Sigma_{jm}) \tag{2.3}$$

the estimation formula for parameters of the GMM become [27]

$$\bar{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)}{\sum_{t=1}^{T} \sum_{l=1}^{M} \gamma_{jl}(t)} \tag{2.4}$$

$$\bar{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t) o_t}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{2.5}$$

$$\bar{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)(o_t - \mu_{jm})(o_t - \mu_{jm})'}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{2.6}$$

where the modified state occupancy probability is the joint probability of being in state j at time t with the mixture m accounting for observation o_t. It is given as

$$\gamma_{jm}(t) = \gamma_j(t) \frac{c_{jm} N(o_t; \mu_{jm}, \Sigma_{jm})}{\sum_{l=1}^{M} c_{jl} N(o_t; \mu_{jl}, \Sigma_{jl})} = \frac{1}{P} \alpha_j(t) \beta_j(t) \frac{c_{jm} b_{jm}(o_t)}{b_j(o_t)} \tag{2.7}$$

where we have simplified $P = P(O|\lambda)$ since it is constant for a given training utterance. Here we only give the formula for the case of a single training utterance. It is straightforward to extend them to the case of multiple training utterances.

2.3 Phonetic Decision Trees

A phonetic decision tree is a type of classification and regression tree (CART). Breiman et al. [10] provided the theoretical framework for developing decision trees. Decision trees are widely used in speech research, such as statistical modeling for natural language [2], speech unit selection in speech synthesis and acoustic modeling in speech recognition. The phonetic decision-tree used in acoustic modeling is a binary tree in which a yes-no question about phonetic context is attached to each node. According to the answer to the question, the data associated with a node are split into two subsets. This procedure repeats until certain stopping criteria are satisfied. Figure 2.2 illustrates a phonetic decision tree for clustering all triphone models that are derived from monophone /iy/. Each node (except leaf nodes) is associated with a phonetic question concerning the identity of left or right phone. A model for each triphone that is derived from monophone /iy/ can be found by traversing the tree to a leaf node by answering the questions along the path and using the model in the leaf node to represent it.

2.3.1 Definition

There are three basic elements of a phonetic decision tree:

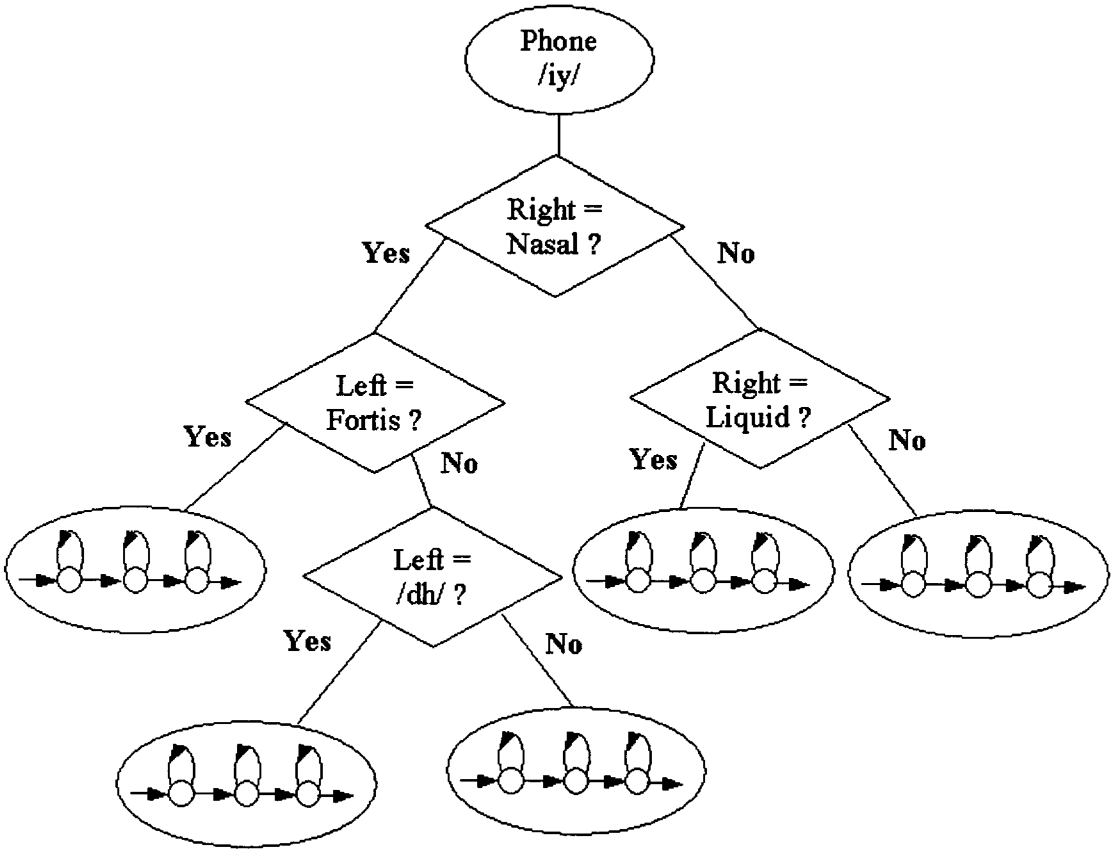


Figure 2.2: A phonetic decision tree to cluster triphones that are derived from monophone /iy/.

- A set of yes/no phonetic questions
- A goodness-of-split criterion
- A stop-splitting criterion

The set of questions represents possible partitions of the data. Let P_0 denote the current phone. Define *context* as the identities of the K previous phones and K following phones in the phone sequence, denoted as $P_{-k}, \ldots, P_{-1}, P_1, \ldots, P_k$. The questions ask about the characteristics of the phones in the context, P_i, for $i = \pm 1, \ldots, \pm k$. Particularly, for the triphone case, the questions are only concerned about the previous phone and the following phone, $i = \pm 1$. Let P denote the alphabet of phones, and N_p the size of this alphabet. Let

Q denote the question set that consists of questions of the form $[Is\ P_i \in S\ ?]$ [1], where S is a set of phones that are elements of P. S might contain one or more phones belonging to a given category. Typically, these categories are manually pre-defined and correspond to phonologically meaningful classes of phones commonly used in the analysis of speech.[2] For example, $S = \{p, t, k\}$ is the set of all unvoiced stops, and $S = \{p, t, k, b, d, g\}$ is the set of all stops. Each question is applied to each element P_i, for $i = \pm 1, \ldots, \pm k$, in the context.[3]

The goodness-of-split criterion is used to determine which of the available questions can best divide the data associated with a node. The best split maximizes the decrease in data impurity. In other words, the data within the child nodes after the split should be more similar than the data in the parent node before the split.

The stop-splitting criterion determines when to stop the splitting procedure and thus the size of the tree. Typically, two stopping criteria are utilized. One is that each leaf node should have at least some minimum amount of data. The other one is that the gain (according to the goodness-of-split criterion) resulting from the best split of a node should be greater than a threshold. If either of these criteria is not satisfied at a node, it is no longer split and becomes a leaf node.[4]

Let n denote a node in the tree, and $m(q, n)$ denote the goodness of the split induced by question $q \in Q$ at node n. We define a tested node as one on which we have evaluated $m(q, n)$ for all questions $q \in Q$ and either split the node or designated it as a terminal node. Since the construction of an optimal binary decision tree is an NP-hard problem, in practice a sub-optimal greedy algorithm is used to construct the tree, selecting a best question from the question set Q at each node.[5] In summary, a general decision-tree

[1]It is a simple question. More complex questions, formed by conjunction, disjunction and/or negation of simple questions, were investigated in [2] [24]. In [31][33][48], the question set was expanded to incorporate more information, such as stress, tone, and gender. Some improvement was obtained at the cost of increased computation.

[2]Automatic approaches of generating the question set from training data were investigated in [9][50].

[3]A different partition algorithm was investigated in [42]. It used the "CPA" [12] algorithm, which did not use any phonetic questions. However, it did not perform well, especially in larger tasks, due to the lack of capability to predict unseen triphones.

[4]A more complicated stopping criterion, using cross-validation, was investigated in [42]. However, the performance was worse.

[5]A look-ahead search was studied in [30]. However, surprisingly, no improvement was obtained.

construction algorithm [3] works as shown in Figure 2.3.

```
1. Start with all data at the root node.
2. While there are untested nodes do
   2.1 Select some untested node n.
   2.2 Evaluate $m(q, n)$ for all possible questions $q \in Q$ at this node.
   2.3 If a stopping criterion is met, declare this node as a terminal node, else
   2.4 Associate the question with the highest value of $m(q, n)$ with this node. Make two
       new successor nodes. All data that answer positively to the question are transferred to
       the left successor and all other data are transferred to the right successor.
```

Figure 2.3: An algorithm to build decision trees.

2.3.2 Goodness-of-Split Criteria

There are two typical goodness-of-split criteria. Bahl et al. [3] developed a criterion based on the discrete HMM framework. Since the VQ process in discrete HMM systems introduces distortion that limits the model precision, most modern systems use continuous HMMs. Young et al. [60] presented a criterion based on continuous HMMs that have a single-mixture Gaussian as the state output distribution. This criterion has proved very successful and is widely employed. It is based on the maximum likelihood (ML) of the training data, which is consistent with the criterion used for estimating HMM models. Several assumptions are made [44]:

1. The assignment of data to each HMM state is not altered during the clustering procedure.

2. The contribution of the HMM transition probabilities to the total likelihood can be

ignored.

3. The total likelihood of the training data can be approximated by a simple summation of the log-likelihood weighted by the probability of state occupancy.

Given these assumptions,

$$L = \sum_{e=1}^{E} \sum_{t=1}^{T} \sum_{s \in S} \ln(P(o_t^e; \mu_s, \Sigma_s)) \gamma_s^e(t) \;\approx\; \ln(P(O; S)) \tag{2.8}$$

is the approximate log-likelihood of a set of models comprising the set of distributions S generating the training data O that consist of E examples. Here $\gamma_s^e(t)$ is the probability of state s generating the data point at time t of example e. The value can be probabilistic (obtained from the forward-backward algorithm) or deterministic (obtained from labeled data either by manual labeling or by the forced alignment process using a well-trained model set). Building a decision tree is a procedure to find the optimal set of distribution S to maximize L.

For single-Gaussian distributions

$$\ln(P(o_t^e; \mu_s, \Sigma_s)) = \ln\left(\frac{1}{\sqrt{(2\pi)^n |\Sigma_s|}} e^{-\frac{1}{2}(o_t^e - \mu_s)' \Sigma_s^{-1} (o_t^e - \mu_s)}\right) \tag{2.9}$$

So,

$$L \;=\; \sum_{e=1}^{E} \sum_{t=1}^{T_e} \sum_{s \in S} -\frac{1}{2}(n \ln(2\pi) + \ln(|\Sigma_s|) + (o_t^e - \mu_s)' \Sigma_s^{-1} (o_t^e - \mu_s)) \gamma_s^e(t) \tag{2.10}$$

According to the ML parameter re-estimation formula in Section 2.2.3,

$$\Sigma_s = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_s^e(t)(o_t^e - \mu_s)(o_t^e - \mu_s)'}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_s^e(t)} \tag{2.11}$$

We can get

$$L \;=\; \sum_{s \in S} -\frac{1}{2}(n(1 + \ln(2\pi)) + \ln(|\Sigma_s|)) \sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_s^e(t) \tag{2.12}$$

The value of Σ_s can be calculated from the statistics for each unique context without reference to the original training data.

$$\begin{aligned}\Sigma_s &= E[o^2] - E[o]^2 \\ &= \frac{\sum_{c\in C(s)} \gamma_c(\Sigma_s + \mu_c\mu_c')}{\sum_{c\in C(s)} \gamma_c} - \left(\frac{\sum_{c\in C(s)} \gamma_c\mu_c}{\sum_{c\in C(s)} \gamma_c}\right)\left(\frac{\sum_{c\in C(s)} \gamma_c\mu_c}{\sum_{c\in C(s)} \gamma_c}\right)'\end{aligned}$$

where $C(s)$ is the set of triphones that are to be represented by the distribution of the tied states.

Because splitting a given distribution is assumed to have no effect on the remaining distributions, only the local improvement in the total likelihood needs to be calculated. Splitting a node changes the set of distributions S by replacing the parent p distribution with its left child node l and right child node r. So the changes in overall log-likelihood, which is the quantity that needs to be maximized, is just the difference between the likelihood of the parent and its children.

$$\begin{aligned}\delta L &= -\frac{1}{2}\ln(|\Sigma_l|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_l^e(t) - \frac{1}{2}\ln(|\Sigma_r|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_r^e(t) \\ &\quad +\frac{1}{2}\ln(|\Sigma_p|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_p^e(t)\end{aligned}$$

So at each node, the evaluation function $m(q,n)$ in Figure 2.3 is δL for question q at node n.

2.3.3 Phonetic Decision Tree Based Acoustical Modeling

Usually, one tree is built for each HMM state of each base phone to cluster all the corresponding states of all of the associated triphones. Figure 1.2 illustrates the idea. It is called a state-based decision tree. In contrast, in a model-based approach (see Figure 2.2), the unit to be clustered is an HMM model (as opposed to a state of an HMM model in the state-based approach), and thereby only a single tree is needed to cluster all triphones that are derived from the same base phone. Since the state-based approach provides a finer level of sharing and thereby achieves more accurate and robust acoustic models than the model-based approach, it is widely used.

In state-based clustering, it is also possible to build a single tree to cluster all HMM states of all triphones that are derived from a base phone. The motivation is that we might achieve some sharing among different HMM state positions. However, the computation cost is higher and studies [31] showed that the first few questions chosen in the tree were always the state-position questions, that is, "is it the first state?" and "is it the second state". Essentially, it was equivalent to a simple combination of separate state-based decision trees, therefore no advantage was obtained.

A typical procedure to build a phonetic decision tree based acoustic model is given below. There are four main steps [60]:

1. An initial set of monophone models with single-Gaussian output probability density functions is created and trained.

2. The state output distributions of the monophones are cloned to a set of unclustered context-dependent triphone models, which are then trained by using the forward-backward algorithm.

3. For each set of triphones derived from the same monophone, corresponding states are clustered using a phonetic decision tree.

4. After the clustering is finished, the number of mixture components in each clustered state is increased and the models are re-estimated until performance on a development set peaks, or the desired number of mixture components is reached.

You may notice that the models for decision tree building are single-Gaussian distributions while the final models have multiple-Gaussian distributions. This disparity is partially due to the fact that there exists an efficient algorithm to construct decision trees using single-Gaussian models while it is difficult to do so with multiple-Gaussian models. This problem will be explained in details in Chapter 6 and an approach will be proposed to address it.

The step 4 in the above procedure is realized by a mixture-splitting process in a step-by-step fashion to smoothly increase the model complexity. Figure 2.5 illustrates the mixture splitting procedure. First the largest (either has the biggest mixture weight or

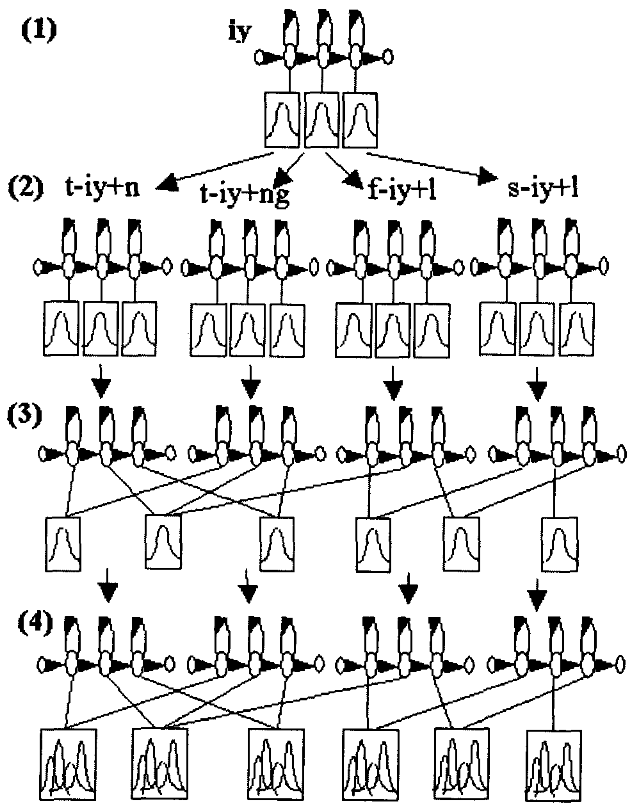


Figure 2.4: Procedure to build acoustic models using phonetic decision tree based context clustering(after [60]).

variance) Gaussian component is selected. Then the Gaussian component is duplicated. The resulting identical components have their means perturbed from their original values by a fraction (e.g., 0.2) of their standard deviations, their mixture weights halved and their variances left unchanged. The resulting model is then retrained using the Baum-Welch algorithm. This mixture splitting is then repeated as necessary to smoothly increase the model complexity, until its performance on a development set peaks or the desired number of mixture components is reached.

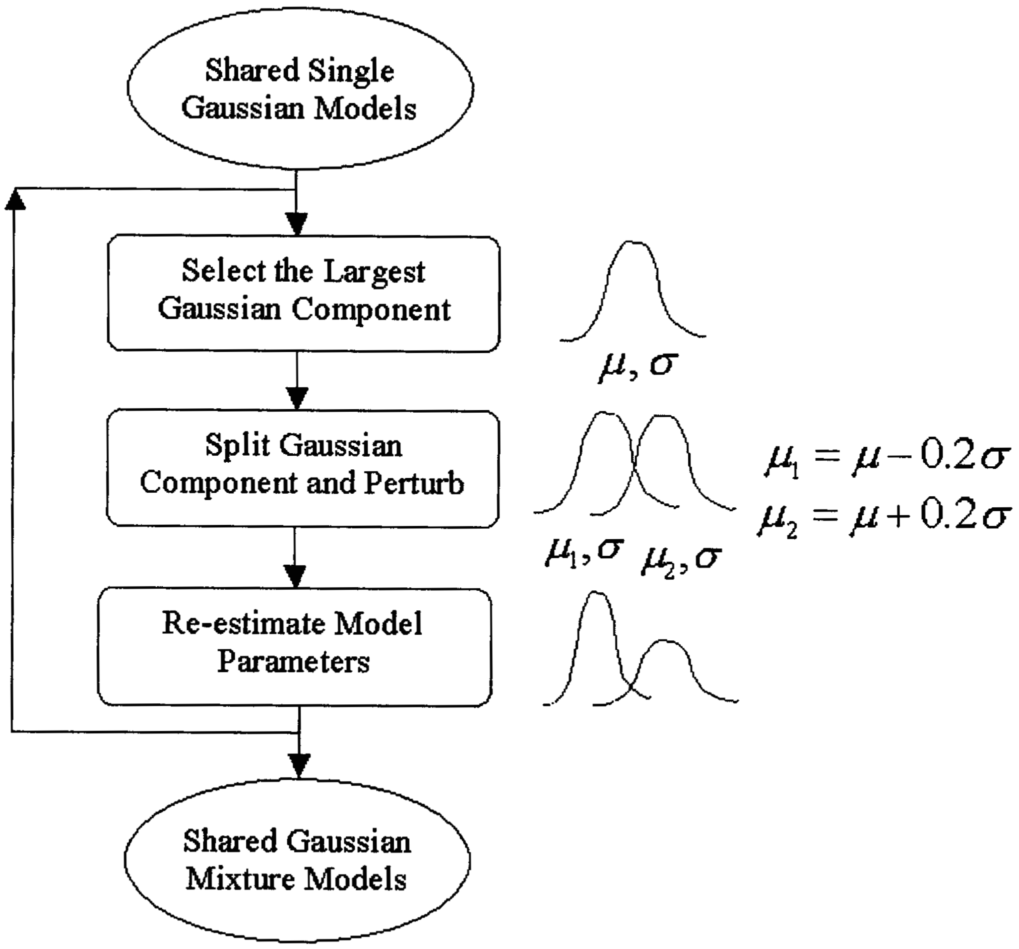

Shared Single Gaussian Models
Select the Largest Gaussian Component
Split Gaussian Component and Perturb
Re-estimate Model Parameters
Shared Gaussian Mixture Models
μ, σ
μ_1, σ
μ_2, σ
$\mu_1 = \mu - 0.2\sigma$
$\mu_2 = \mu + 0.2\sigma$

Figure 2.5: Procedure to split mixtures and increase model complexity.

Chapter 3

Tasks and Baseline System

This chapter describes the speech recognition tasks that are used throughout this thesis and the baseline system, which uses the traditional decision tree based acoustic modeling approach. Two standard training corpora, TIMIT and Wall Street Journal (WSJ), are used to build acoustic models. The test data include the DARPA standard CSR evaluation sets for November 1992, for both 5,000 word closed vocabulary [1] and 20,000 word open vocabulary tasks. Section 3.1 presents a brief overview of the training and test corpus. Section 3.2 explains how to build an acoustic model from scatch using phonetic decision tree based clustering. In Section 3.3, we propose a new training strategy to improve the original baseline system. Experimental results confirm that the new training approach is superior to the original one in terms of both accuracy and speed.

3.1 Databases and Tasks

Two databases are used in our experiments: TIMIT corpus and Wall Street Journal (WSJ) corpus. TIMIT is used only for initial training of acoustic models. WSJ corpus is a standard database for DARPA sponsored large-vocabulary continuous speech recognition evaluation initiated in 1992.

[1]Closed vocabulary means that all the words in the test sets are included in the lexicon. In contrast, open vocabulary means that not all words in the test sets are included in the lexicon. The out-of-vocabulary (OOV) words in the test data bring a great challenge to speech recognition systems.

3.1.1 TIMIT

The TIMIT corpus of read speech was designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems. It contains broadband recordings of 630 speakers of 8 major dialects of American English, each reading 10 phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16kHz speech waveform files for each utterance.

Example transcriptions for an utterance in the corpus are as follows.

Orthography:

0 61748 She had your dark suit in greasy wash water all year.

Word label:

7470	11362	she
11362	15420	had
15420	17503	your
17503	23360	dark
23360	28360	suit
28360	30960	in
30960	36971	greasy
36971	43120	wash
43120	49021	water
49021	52184	all
52184	58840	year

Phonetic label:

0	7470	h#
7470	9840	sh
9840	11362	iy
11362	12908	hv
12908	14760	ae
14760	15420	dcl
15420	16000	jh
16000	17503	axr
17503	18540	dcl
	....	
56654	58840	axr
58840	61680	h#

The phonetic transcription is very important for initial training (bootstrapping) of each individual HMM phone model, as we will see in Section 3.2.

3.1.2 Wall Street Journal

The Wall Street Journal corpus is a large database of spoken American English. The prompting texts were drawn from articles appearing in the Wall Street Journal. The articles used for both the training and testing portions of the database were filtered to limit the vocabulary to the 64,000 most frequently occurring words in the whole database, which consists of approximately 37 million words of text. The data was collected simultaneously through two channels. One used a close-talking head-mounted Sennheiser microphone; the other used a variety of desktop microphone. In our experiments, only the close-talking data are used. The WSJ training corpus consists of two parts:

- **WSJ0**

  WSJ0 consists of three sections of almost equal size.

  - Longitudinal Speaker Dependent, LSD
  - Long term speaker independent, SI12
  - Short term speaker independent, SI84

  Among these, only the SI84 section was used for training. It consists of 7193 sentences from 84 different speakers (42 males and 42 females) for a total of approximately 12.2 hours of speech.

- **WSJ1**

  WSJ1 consists of two sections of almost equal size.

  - Long term speaker independent, SI25
  - Short term speaker independent, SI200

  Again, only the short term speaker independent data set SI200 was used for training. It contains 29320 sentences from 200 new speakers (100 males and 100 females) for a total of 45.1 hours of speech.

Some examples of the sentences in the WSJ corpus are given in Figure 3.1. Note that the prompts have two conditions, verbalized punctuations (vp) or non-verbalized punctuations (nvp). In the verbalized-punctuation condition, punctuations are pronounced verbally.

For the 20,000 word tasks explained below, both the short term speaker independent portions of the database, SI84 and SI200, were used for training and these were collectively referred to as SI284.

Despite the decline in stockprices trading volume wasn 't overwhelming.

And the total allowance for credit losses remained unchanged at one point three billion dollars.

Clearly ,comma something is changing in Europe .period

A white house game of "double-quote chicken "double-quote with Germany and Japan ,comma played by talking down the dollar ,comma hasn 't helped either .period

But what exactly is its threat to life or health ?question-mark

Figure 3.1: Sample utterances in the WSJ training corpus. The first two utterances are nvp sentences; the last three are vp.

Table 3.1: Summary of Wall Street Journal database test sets

Test Set	Vocabulary	Sentences	Speakers	Words
eval set nov92	5K close	330	8	5353
dev set si-dt-05	5K close	442	10	7114
eval set nov92	20K open	332	8	5643

3.1.3 Tasks

Official ARPA Continuous Speech Recognition evaluations were conducted in Nov'92, Nov'93 and Nov'94. There are two tracks: one for the 5,000 word task (WSJ 5K) and one for the 20,000 word task (WSJ 20K). The recommended training data sets for WSJ 5K and WSJ 20K are SI84 and SI284, respectively. Table 3.1 gives some statistics for the test sets used in this thesis.

3.2 Original Baseline System

During the past few years, our research group has been actively focused on research and development of large vocabulary continuous speech recognition systems (LVCSR) and participated in government sponsored annual evaluations (Broadcast News Transcription

(HUB4) 1997 [57] and 1998 [55], Speech In Noise Environment (SPINE) [56]). Our research software platform is a continuous HMM-based speech recognition system, which uses a statistical n-gram language model. The platform has a complete package for signal processing and feature extraction (the commonly used features such as MFCC, PLP, LPC; and noise/channel variation reduction techniques such as Cepstral Mean/Variance Normalization), speaker and channel segmentation (using Bayesian Information Criterion (BIC) [11]), acoustic model training, speaker adaptation (such as commonly used Maximum A Posteriori (MAP) [32], Maximum Likelihood Linear Regression (MLLR) [36], Speaker Adaptive Training [1], Vocal Tract Length Normalization (VTLN) [35]) and decoders (both single-pass decoder and two-pass decoder).

For a very large vocabulary task, such as transcribing broadcast news where the vocabulary size is over 60K, usually the multiple-pass decoding strategy is utilized to make the computation tractable and memory size manageable. In the first pass decoding, crude acoustic models (such as a within-word triphone model) and simple statistical language models (such as a bi-gram) are used to generate a graph of the most likely word sequences. The second pass searches this graph to find the single best hypothesis, using more sophisticated acoustic models (such as a cross-word triphone or even quinphone model) and language models (such as a tri-gram or even four-gram).

For this thesis, however, we chose the WSJ 5K and 20K tasks as the test environments because they are difficult enough to represent some applications, but have a tractable computational cost to speed up the experiment turn-around time. For WSJ tasks, we can afford to use just a one pass decoding strategy, where a cross-word triphone model and a statistical tri-gram are used in a single pass to generate the best hypothesis. It avoids training two sets of acoustic models (within-word and cross-word) for every experiment. In the following subsections, we will briefly explain every component of the acoustic modeling part of our LVCSR system and the procedure to build an acoustic model from training data.

3.2.1 Signal Processing and Feature Extraction

The speech signal is highly redundant because of the strong correlation between adjacent segments. Therefore, speech recognition systems always use a parametric representation rather than the speech waveform itself. Not only is useful information compactly extracted from the waveform, but also computation is saved for both training and decoding. It is crucial to extract a maximum of related information for a specific task and discard unrelated information. For speech recognition, information about speech contents (linguistic and phonetic information) must be preserved, while information about speaker identity is irrelevant. Over the years, various types of parametric representation for speech recognition have been proposed. Most of them are based on short-time spectrum analysis of the speech signal. A fundamental assumption underlying the short-time analysis is that over a long-time interval speech is non-stationary, but that over a sufficiently short-time interval it can be regarded as stationary. Thus, the Fourier transform of a short segment of speech (called a frame) should give a good representation of the speech during that time interval.

Probably the most successful and commonly used representations (or acoustic features) of the speech signal for recognition purposes are Mel-Frequency Cepstral Coefficients (MFCC) [14] and Perceptual Linear Prediction (PLP) [22]. We will illustrate the procedure of extracting MFCC features below as it was used in all the experiments in this thesis.

1. The input speech signal is sampled at 16 kHz (this step is usually skipped as the speech corpus has already been digitalized and stored on CDs).
2. The sampled waveform is blocked into frames. Each frame spans 25 msec.
3. A pre-emphasis filter $H(z) = 1 - 0.97z^{-1}$ is applied to get rid of the lip effect [40].
4. Fast Fourier Transform (FFT) is applied to obtain the spectral representation, followed by a logarithm conversion.
5. Use Mel-spaced filterbanks to map the spectrum of linear scale to mel scale based on perceptual studies of human's hearing.

6. Discrete Cosine Transform (DCT) is applied to the filterbank output to convert the spectral domain coefficients to cepstral domain. The major advantage of doing this is that in cepstral domain it is much easier to get rid of some channel distortions, using techniques like Cepstral Mean Subtraction/Normalization. The first 12 coefficients are preserved, which become our target MFCC feature vectors. The MFCC coefficients are generally uncorrelated, which enables the covariance matrices in the multivariate Gaussian distributions to be diagonal. This dramatically reduces the computation cost of HMM training and decoding.

Figure 3.2 gives a diagram of the acoustic feature processing in our system. Note that energy information is also extracted for every frame and appended to the MFCC feature vector.

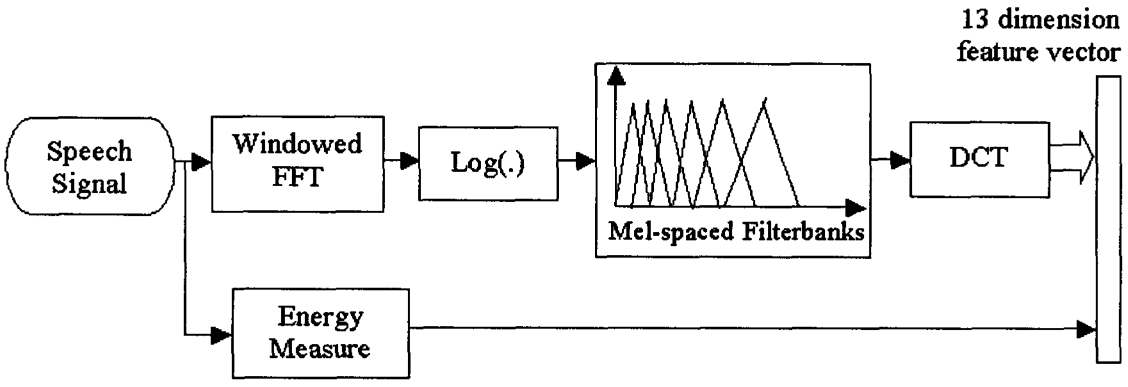


Figure 3.2: Diagram of extracting MFCC acoustic feature vectors from speech data.

As the spectral pattern of a frame only contains local and static information of a sound, it is necessary to use some dynamic feature to capture the change of spectral patterns over time. The most common method of doing this is to estimate the delta and acceleration of the spectral coefficients over a series of consecutive frames, and then append these measurements to the basic static feature vectors. Usually, a linear regression equation is used:

$$d_i(t) = \frac{\sum_{k=1}^{N}(c_i(t+k) - c_i(t-k))}{2\sum_{k=1}^{N} k^2}$$

where c_i denotes the i-th cepstral coefficient, d_i denotes its delta coefficient and $(2N+1)$ gives the size of the regression window.

So the final feature vector for our system has 39 elements, consisting of 12 MFCCs and normalized energy plus their first and second order time derivatives.

3.2.2 Acoustic Model Training

All phone models have three emitting states and a left-to-right topology without state skip. Figure 2.1 illustrated a typical HMM phone model. To capture the possible short pauses between words, a special model, called *tee* model, is used (Figure 3.3). It has only one emitting state. The entry and exit states have no output distribution functions. They are provided for concatenating with other models. Note that the entry state has a non-zero transition to the exit state so that the tee model may generate no feature vectors, which makes it an optional model.

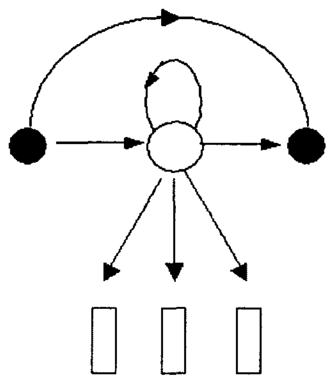


Figure 3.3: Optional short-pause model or tee model. The entry state has a non-zero transition to the exit state.

The complete training procedure involves a few steps:

1. monophone training

2. unclustered cross-word triphone training

3. building phonetic decision trees to cluster triphone states

4. training of clustered triphone models

A. Monophone Training (Bootstrapping)

The whole procedure of building the acoustic model starts with monophone training, then gradually increases the model complexity and accuracy step by step. A good seed monophone model is important to the quality of the final complex model. Usually a phonetically labeled database, such as TIMIT, is necessary for bootstrapping. We collect the segments of speech data corresponding to a particular monophone and use the Viterbi training algorithm and the Baum-Welch algorithm (or forward-backward algorithm) to train the monophone models.

Figure 3.4 illustrates the procedure of the Viterbi training algorithm. First, we define the prototype of every HMM phone model, which includes the HMM topology and initial values of transition probabilities. At this time, we do not need to initialize the output distribution functions of HMM states. Instead, a simple uniform segmentation is applied to every utterance, that is, the utterance is chopped into three (for a 3-state HMM model) segments of equal size. All corresponding data for a state are pooled together and used to estimate a Gaussian mixture distribution. Here the K-means algorithm [46] is used to cluster the vectors within each state. In our case, we only need to estimate a single Gaussian distribution for every HMM state.

Next, we use the Viterbi algorithm (see Section 2.2.2) to find the best HMM state sequence for every utterance, given the initialized HMM model. We use this new alignment to calculate new HMM parameters. At the same time, we can obtain the joint probability of the observations and the best state sequence, given the current HMM model. This probability (averaged over all utterances for one phone) can be used to determine whether the training procedure should stop or not. If the change of the probabilities between successive iterations is below a pre-defined threshold, or the number of iterations reaches a pre-defined limit, the procedure stops and we obtain a new HMM model.

The Viterbi training algorithm imposes a hard alignment of speech data against HMM states. In contrast, the forward-backward training algorithm provides a probabilistic (soft) alignment, which is generally more accurate than the hard alignment, of the speech data against HMM states. So usually after the Viterbi training, a forward-backward training is applied to improve the quality of the monophone models.

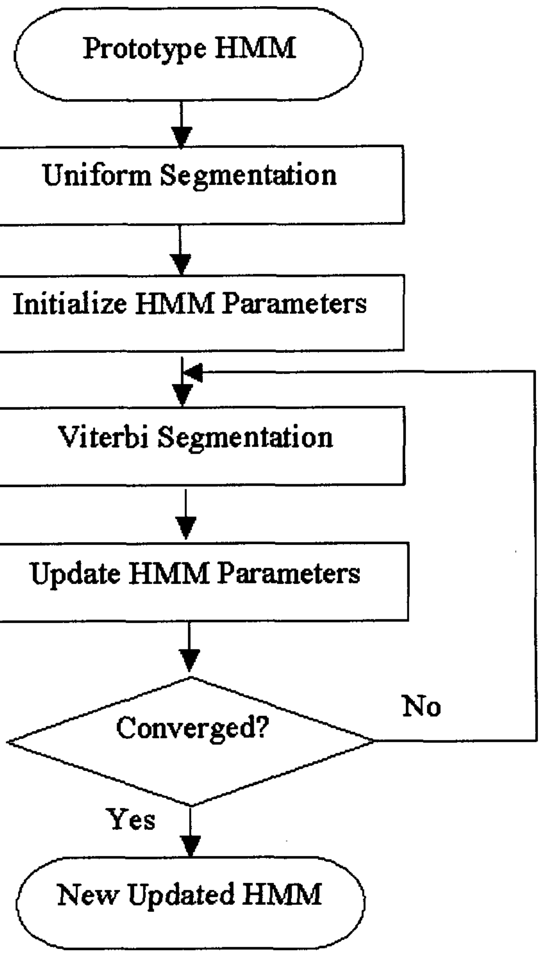


Figure 3.4: Viterbi training of HMM models.

The procedure of the forward-backward training algorithm is similar to the Viterbi training. The difference is that instead of using Viterbi segmentation, it uses the forward-backward algorithm to calculate the required statistics and accumulate them across all training utterances. Then HMM parameters are derived from the statistics according to those mathematical equations in Section 2.2.3.

B. Un-clustered Triphone Training

In this stage, the monophone models with single-Gaussian output distributions, obtained through previous training stages, are first cloned to triphone models. All triphones that are derived from the same monophone have identical HMM models. A few iterations of embedded HMM training using WSJ corpus are then applied to estimate the statistics for every unique triphone model. The embedded HMM training differs from the regular or single-form HMM training in that speech data for complete sentences are directly used, without the word or phone level boundary information available. Phone HMM models are concatenated together to form a big sentence HMM model, according to the word sequence in the sentence and the pronunciation dictionary. Each phone HMM model is embedded in the sentence HMM model. The forward-backward algorithm is used to map a complete speech sentence against the corresponding sentence HMM model.

While in the single HMM training only a single monophone HMM model is trained in each run, in embedded training the whole set of triphone models is loaded into memory and updated when all training utterances are iterated through (see Figure 3.5). Due to the large size of the training corpus like WSJ, it is necessary to distribute the training data to multiple CPUs in one or more machines, so that they can run in parallel and speed up the training process. In this case, a parallel and synchronized mechanism is necessary. We proposed a mechanism in Section 3.2.3 to effectively do this.

C. Building Phonetic Decision Trees to Cluster Triphone States

Given the updated triphone models and their occupancy statistics, a phonetic decision tree is constructed for every HMM state position of every phone. A general algorithm for building a decision tree was given in Figure 2.3. In practice, however, there is some variation. After the splitting procedure stops, it is found useful to do a further step to merge similar leaf nodes, using the same goodness-of-splitting criterion and the same likelihood threshold. Figure 3.6 illustrates the procedure. Not only is the number of parameters reduced, but also the greediness of the tree-building algorithm becomes less. The resulting tree is not, strictly speaking, a tree, but rather a graph due to the merging step.

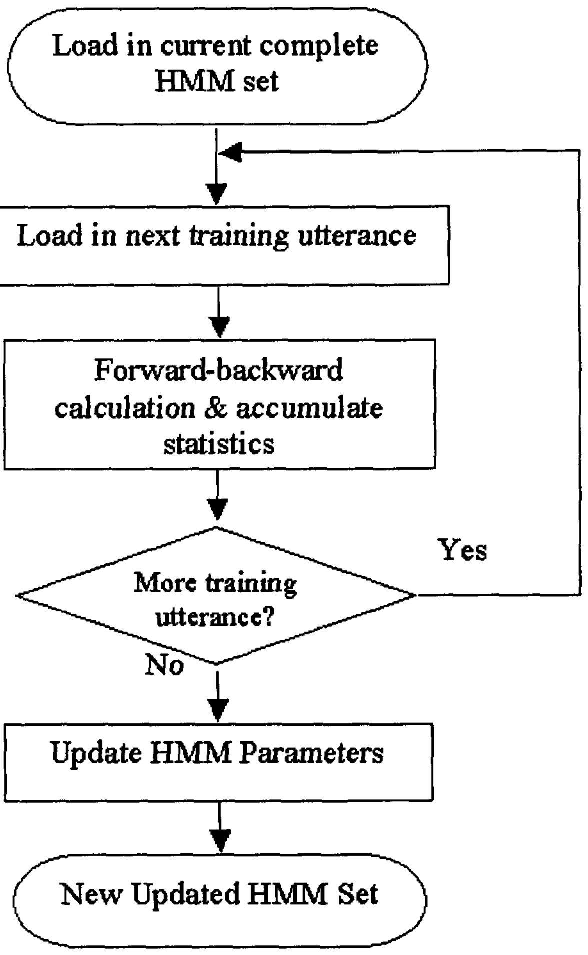

Load in current complete HMM set
Load in next training utterance
Forward-backward calculation & accumulate statistics
More training utterance?
Yes
No
Update HMM Parameters
New Updated HMM Set

Figure 3.5: One iteration of embedded forward-backward training.

D. Training of Clustered Triphone Models

At the end of step **C**, a typical HMM state with a single-Gaussian distribution, which has the largest variance among all the states in a leaf node, is selected to represent the distribution of that node. At this training stage, a series of alternate embedded training and mixture splitting processes is used to gradually increases the complexity and accuracy of the clustered triphone models. First, the single-Gaussian model is re-estimated through several iterations of forward-backward training. Then the model is split into a 2-mixture model using the mixture-splitting technique (see Section 2.3.3), followed by several iterations of forward-backward training to obtain new HMM parameters. This process repeats

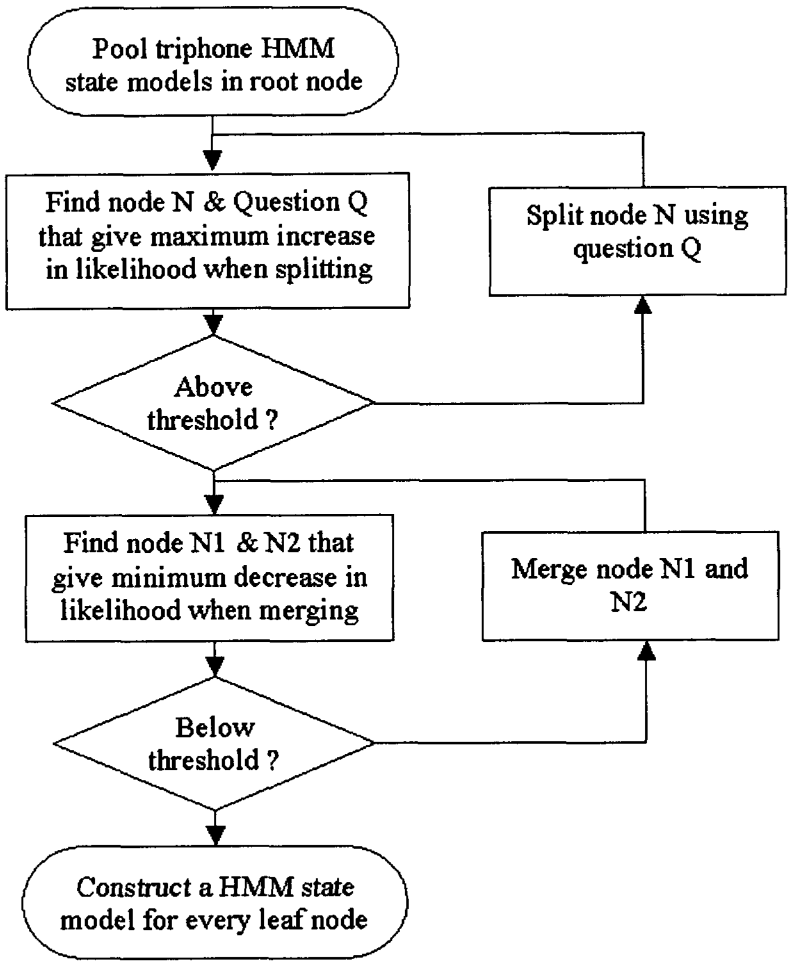

Pool triphone HMM state models in root node
Find node N & Question Q that give maximum increase in likelihood when splitting
Split node N using question Q
Above threshold ?
Find node N1 & N2 that give minimum decrease in likelihood when merging
Merge node N1 and N2
Below threshold ?
Construct a HMM state model for every leaf node

Figure 3.6: Flowchart to build a phonetic decision tree.

until the desired number of components is reached or the performance of the model on a development set peaks.

3.2.3 Some Practical Issues

Usually there are thousands of training utterances for a task that has a medium to large size of vocabulary. For testing, there are also hundreds of utterances. If only one machine or CPU is used to do a full round of training and decoding, it may take a few days or even weeks. To speed up the process, it is necessary to use multiple machines/CPUs. A straightforward way to do this is to partition the training and testing data in advance according to the number of available CPUs, as is used in HTK [58]. However, it may be difficult to fairly partition the data because (a) the speed of every machine may vary; (b) the length of every utterance is different. One solution is to distribute the data according the total time of data rather than the number of utterances. However, the computation cost for utterances of the same length may be dramatically different, especially for decoding where the computation cost also depends on the difficulty of the utterance. In practice, we found that this method is not very effective even after careful partitioning of data. Another problem with this method is that it has no effective way to recover from exceptions like machine crashes or program failures.

We devised a more complicated but much more effective parallel mechanism to distribute data and synchronize multiple CPUs. It is based on a server/client mechanism, as illustrated in Figure 3.7.

First, we run a server program on one host machine (usually not one of those that are participating in training). It keeps running at the background on the host machine, listening for requests from client processes at a particular communication port. The server program never terminates, unless the machine crashes or the program is killed by the user. It maintains a record for every session (such as a monophone training session, or a WSJ 5K decoding session using a particular acoustic model set), which is defined by the client process using a session name, a port (or channel) number and the user's ID. The record contains the information of every utterance in the session including (a) to which process it is distributed, and (b) its processing status (not assigned, assigned but not processed, processed). This record enables an effective way to recover from exceptions and make it easy to satisfy users' particular requirements. Upon the request for an utterance from a client process, the server program will check from the top of the record list and send back the ID of an unassigned utterance. As only the ID rather than the utterance itself is transferred through the network, the overhead of this communication is negligible.

Every client training process (running on one CPU) has its own copy of the complete HMM model set and the corresponding accumulators to store the intermediate statistics. It requests an utterance ID from the server and fetches the data from the shared database. When the utterance ID is out of the valid range for the database, this client process

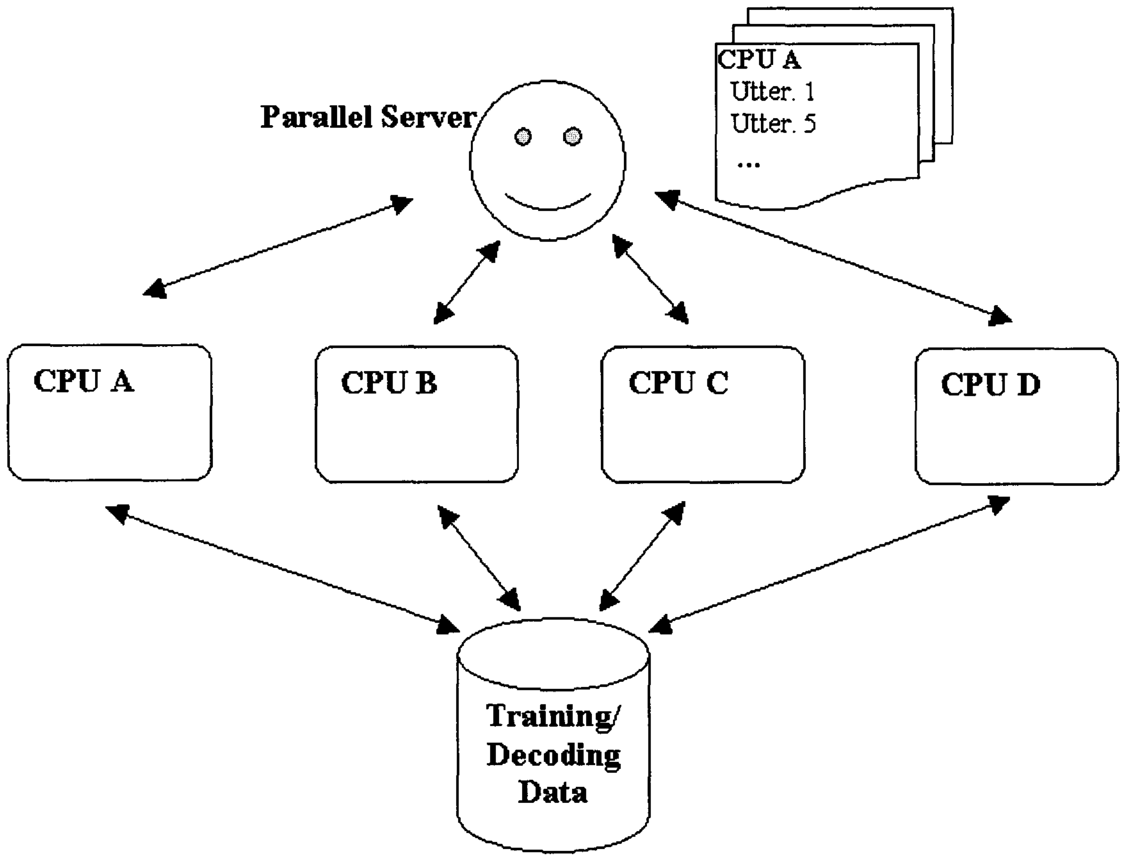


Figure 3.7: Parallel processing mechanism in our system.

terminates, and dumps out the accumulated statistics (for training sessions only). So in parallel processing, one iteration of forward-backward training involves two steps:

1. Every training process computes accumulated statistics for all the utterances assigned to it, and dumps them out;

2. Once all training processes finish, a re-estimation process is started to collect all the statistics from different training processes and use the Baum-Welch re-estimation equations to update the whole set of HMM parameters.

In the second step, a synchronization program is used to ensure that (1) all training processes are finished before it starts the re-estimation process, and (2) the new set of HMM models are dumped out and accessible to all machines before it starts a new iteration of parallel training. When exceptions occur, for example, a training process aborted before it dumped out the accumulated statistics for those utterances it has calculated, we can send a command to the server to change the status of those utterances to unprocessed so that they can be assigned to other live processes again.

Another practical issue about the training process is related to pruning unlikely paths that have very low likelihood in the forward-backward embedded training. For a sentence that has N phones, there are $3N$ HMM states (excluding optional tee model states) in the sentence HMM model. Theoretically, excluding $3N$ frames in the beginning and $3N$ frames in the end of the utterance, every frame could be in any one of the $3N$ states. The width of the forward/backward lattice is therefore $3N$ states. However, among all the possible paths in the lattice, many have very low likelihood and their contributions are negligibile and can be pruned out to reduce computation cost. Figure 3.8 illustrates the idea of pruning in the embeded training. Only those states within the range $[s_L(t), s_H(t)]$ will be considered in the computation at time frame t. To effectively estimate the upper and lower boundaries of the HMM state range, we adopt the method used in HTK [58].

It has two mechanisms to prune unlikely paths. First, the embedded training program calculates the backward probabilties $\beta_j(t)$ first and then the forward probabilites $\alpha_j(t)$. It is unnecessary to calculate these probabilites for all values of state j at time t since many of these combinations will be highly improbable. On the forward pass, we only keep those states for time frame t whose total likelihood as determined by the product $\alpha_j(t)\beta_j(t)$ is within a fixed distance from the total likelihood $P(O|\lambda)$. This pruning mechanism is safe and causes no loss of modeling accuracy.

Second, it is also possible to prune in the backward pass, using the same principle as in the first mechanism. However, in this case, the likelihood $\alpha_j(t)\beta_j(t)$ is unavailable as $\alpha_j(t)$ has not been calculated yet. So a much broader beam is used to avoid pruning errors. If the $\beta_j(t)$ value for a state j is too small, compared against the maximum value of $\beta(t)$ at time t, it is pruned out. The beam is controlled by users.

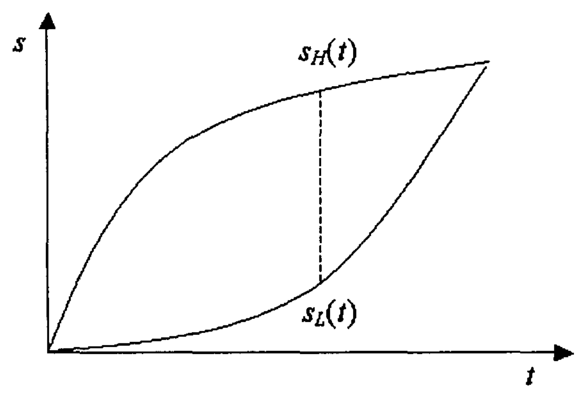


Figure 3.8: Pruning of HMM states in the embedded forward-backward calculation. Only states within the range $[s_L(t), s_H(t)]$ are considered at time frame t.

3.2.4 Experimental Results

To evaluate our new approaches in this thesis, we need to first build the best baseline system that uses the conventional approach, and then compare the results of new approaches against the baseline system under the same conditions. As the performances of the acoustic models are somewhat sensitive to the particular combination of thresholds used in the tree construction and the training procedure, we did a grid search to find the best settings. Two major factors that impact the model quality are the total number of clustered states in the system and the number of mixture components in every state. Selected results are listed in Table 3.2 and Table 3.3 and plotted in Figure 3.9 and Figure 3.10 for the WSJ 5K and 20K tasks, respectively. A bigram language model is used for the WSJ 5K task and a trigram language model for the WSJ 20K task.

We can see that the best acoustic model for the WSJ 5K task has about 3700 clustered triphone states, and 12 Gaussian components per state. The word error rates on the nov92 evaluation set and si-dt-05 development set are 8.1% and 10.2%, respectively. For the WSJ 20K task, the best acoustic model has about 7500 clustered triphone states, and 12 Gaussian components per state. The word error rate on the nov92 evaluation set is 12.9%. These systems will serve as our baseline systems for further improvement.

Table 3.2: Word error rates (%) of acoustic models with different sizes on the WSJ 5K task (nov92 set/si-dt-05 set).

states	2-mixture	4-mixture	8-mixture	12-mixture	16-mixture
1738	11.9/20.1	10.0/15.8	9.4/12.4	8.9/10.8	8.5/10.5
2682	11.3/18.9	9.8/14.9	8.8/12.0	8.5/10.6	8.4/10.5
3749	11.2/18.8	9.2/14.1	8.6/11.6	**8.1/10.2**	8.2/10.3
4656	11.0/18.6	9.1/13.8	8.5/11.9	8.3/10.0	8.5/9.9

Table 3.3: Word error rates (%) of acoustic models with different sizes on the WSJ 20K nov92 evaluation set.

states	2-mixture	4-mixture	8-mixture	12-mixture	16-mixture
5532	19.9	16.5	14.2	13.7	13.6
6542	19.7	16.4	13.8	13.3	13.4
7509	19.2	16.1	13.7	**12.9**	13.0
8562	19.0	15.7	13.7	13.1	13.0

3.3 Improvement on the Baseline System

The original training strategy in Section 3.2.2 generally yields a reasonably good acoustic model in a variety of tasks, such as Resource Management, WSJ 5K and 20K. When the task becomes larger, such as transcribing broadcast news, the variation of speech phenomena in the training corpus, which is much larger, is dramatically increased. In order to capture the variation of speech patterns, a large number of Gaussian components (e.g. 24 or 32) is used to represent each clustered state. However, we found that the original training strategy did not give us the expected gain from the increased model complexity. One possible reason might be due to the fact that the mixture splitting procedure is not an optimal solution to increase the model complexity. Its limitation may become more severe when the amount of training data is small.

We proposed a new training strategy to improve the original one. The construction of phonetic decision trees and clustering of triphone states is kept the same. But the initial HMM state for a leaf node comes from the corresponding multiple-mixture monophone

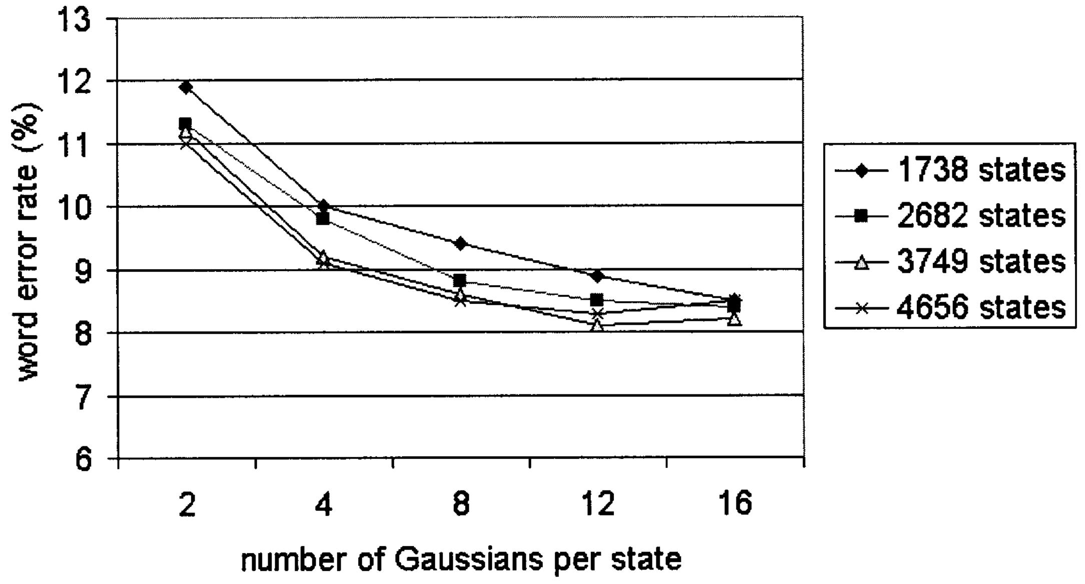


Figure 3.9: Word error rates of acoustic models with different sizes on the WSJ 5K nov92 evaluation set.

model, that is, all leaf nodes for one decision tree will use the same initial HMM model distribution. The multiple-mixture monophone model is obtained through a series of training steps, similar to those in the training of clustered triphone models in Section 3.2.2. It can be trained in parallel to the training of unclustered triphone models and decision tree building. Figure 3.12 shows the parallel paths in the new training diagram. The multiple-mixture monophone model is a crude but robust representation of each monophone's spectral space. A few successive iterations of forward-backward training will adapt (update) the HMM parameters for every clustered triphone state to its specific spectral space. Figure 3.11 and 3.12 illustrate both training strategies for comparison.

Another advantage of the new strategy is that when the phonetic decision tree changes, we do not need to retrain the multiple mixture monophone model, and only need to train the clustered model (that already has the desired number of Gaussian components per HMM state) for several (5 or 6) iterations. While in the original training strategy, we have to start from the initial clustered model that has only a single Gaussian component per state, and go through the complete mixture splitting and re-estimation procedure.

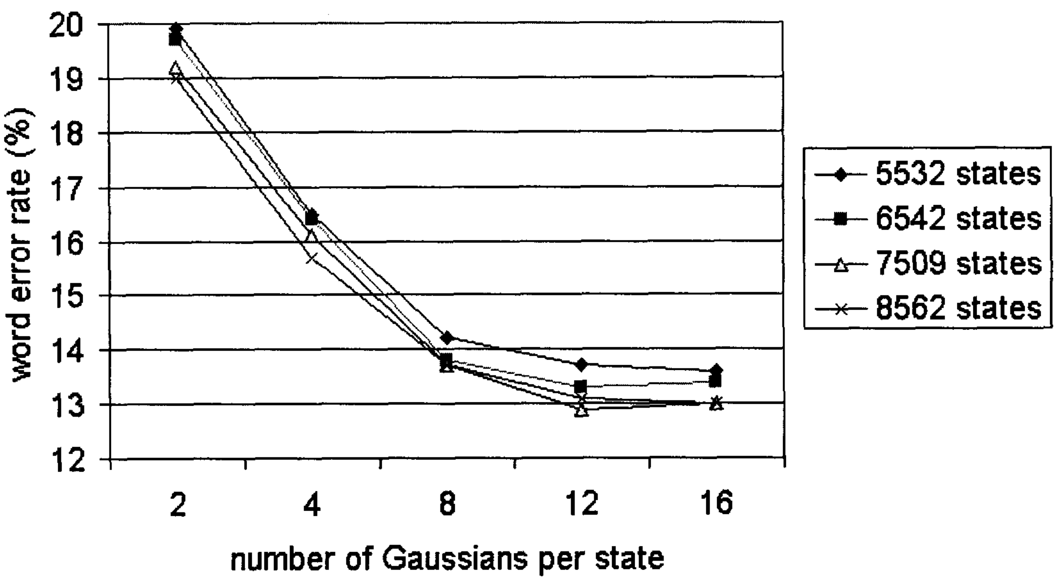


Figure 3.10: Word error rates of acoustic models with different sizes on the WSJ 20K nov92 evaluation set.

Apparently, the new training strategy significantly reduces the experiment turn-around time. In addition, the number of parameters and thresholds during the training procedure are reduced (as fewer steps are involved); therefore the results tend to be more stable and reliable.

The experimental results are listed in Table 3.4 and Table 3.5. Figure 3.13 and Figure 3.14 give the performance comparison on the WSJ 5K and 20K evaluation sets. We can see clearly that the new training strategy consistently outperformed the original training strategy. Not only was the word error rate (WER) reduced, the training time was also reduced by 30% to 40%. For the WSJ 5K task, the best system obtained has about 3700 states and 12 mixtures per state. Its word error rate was 7.5% on the nov92 evaluation set and 9.5% on the si-dt-05 development set. These correspond to a 7% reduction of word error rates compared with the original baseline system, whose word error rates were 8.1% and 10.2% on these two sets. For the WSJ 20K task, the best system has about 7500 states and 12 mixtures per state. Its word error rate on the nov92 evaluation set was 11.8%, which is 8.4% less than the word error rate (12.9%) of the original baseline system.

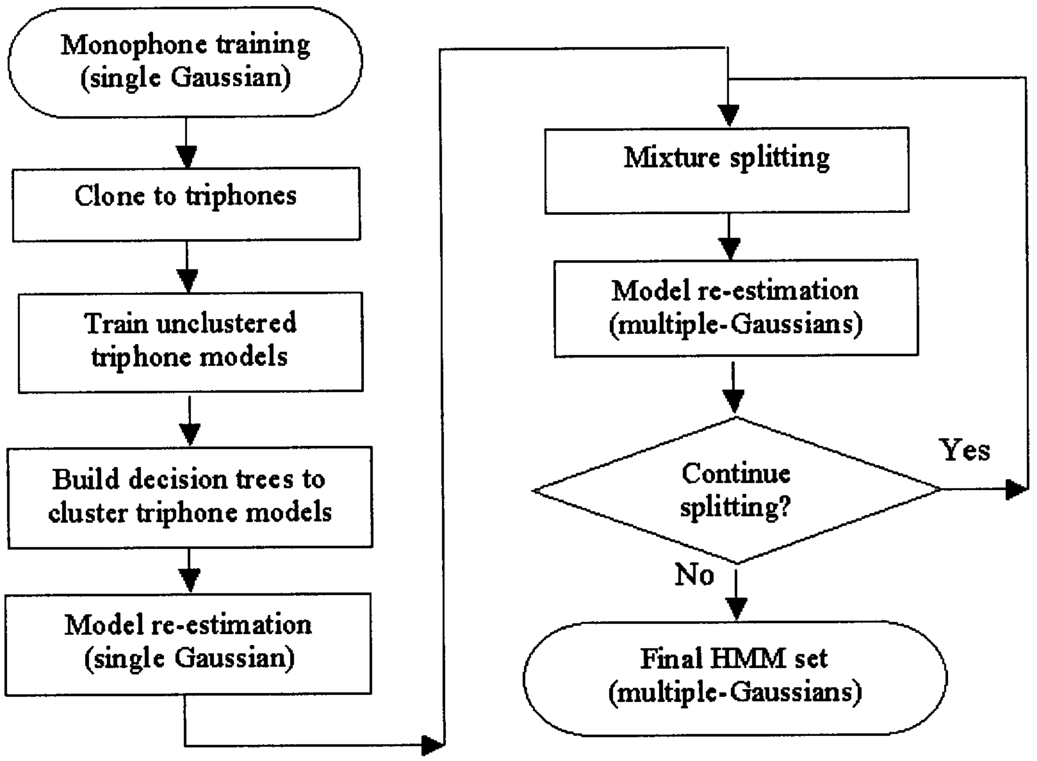

Monophone training (single Gaussian)
Clone to triphones
Train unclustered triphone models
Build decision trees to cluster triphone models
Model re-estimation (single Gaussian)
Mixture splitting
Model re-estimation (multiple-Gaussians)
Continue splitting?
Yes
No
Final HMM set (multiple-Gaussians)

Figure 3.11: Original training strategy.

We did a further fine tuning of the so far best systems on the number of clustered triphone states and the number of mixtures per state. However, no noticeable performance gain was obtained. So we will use these sytems as our new baseline systems.[2] All experiments in the following chapters will use this new training strategy, and their results will compare against the results of these new baseline sysytems.

[2]The results of our baseline systems on these test sets are close to the best published results for comparable systems. The baseline for Bell-Lab's system had a 6.7% WER on WSJ 5K nov92 set and 12.8% on WSJ 20K nov92 set. The HTK group in Cambridge University achieved the best results on those evaluations. Their baseline system had a 6.9% WER on WSJ 5K nov92 set and 9.5% on WSJ 20K nov92 set. However, their system on WSJ 20K task is not comparable to our system due to different system features.

Table 3.4: Results (WER %) of the new training strategy with acoustic models of different sizes on two WSJ 5K test sets (nov92 set/si-dt-05 set).

states	2-mixture	4-mixture	8-mixture	12-mixture	16-mixture
1738	12.0/18.1	10.2/14.0	9.0/11.8	8.5/10.2	8.3/10.1
2682	11.3/16.9	9.0/12.5	8.3/10.8	7.9/9.9	7.8/9.6
3749	10.9/16.6	8.7/12.5	8.1/10.7	**7.5/9.5**	7.6/9.5
4656	10.7/16.2	8.4/11.7	8.0/10.5	7.5/9.8	7.8/9.9

Table 3.5: Results (WER %) of the new training strategy with acoustic models of different sizes on the WSJ 20K nov92 evaluation set.

states	2-mixture	4-mixture	8-mixture	12-mixture	16-mixture
5532	18.5	14.6	13.3	12.7	12.5
6542	17.8	14.0	12.9	12.4	12.4
7509	17.9	13.8	12.4	**11.8**	12.0
8562	17.8	14.1	12.2	12.0	12.2

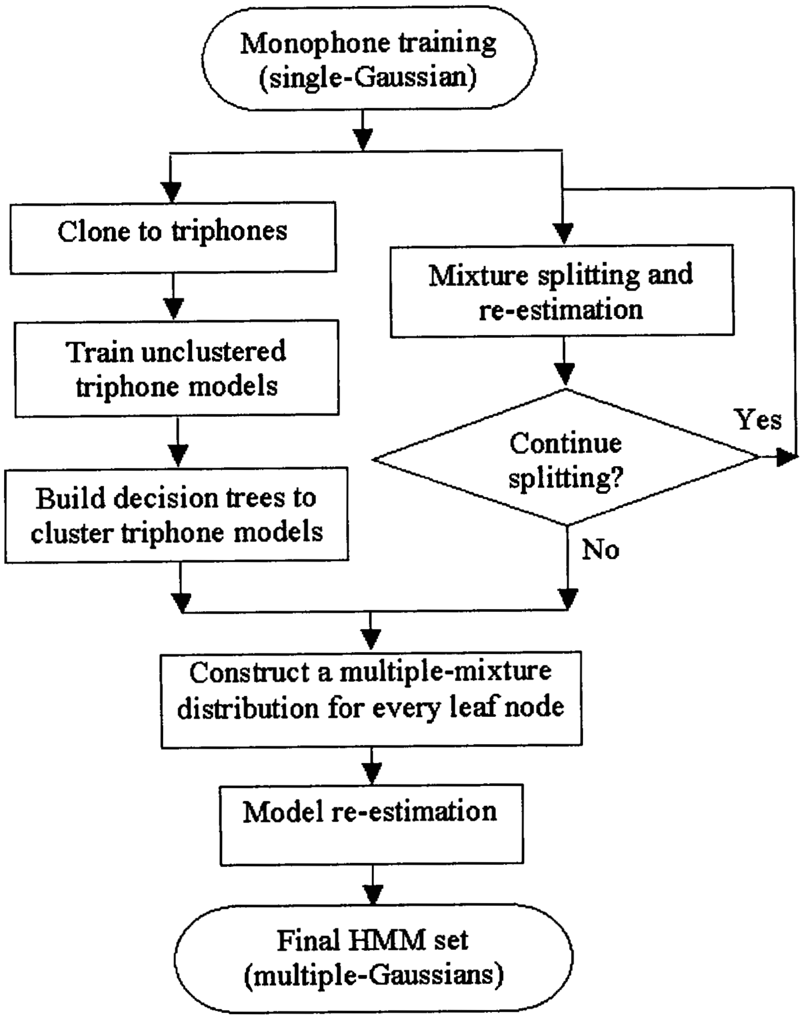

Monophone training
(single-Gaussian)
Clone to triphones
Train unclustered triphone models
Build decision trees to cluster triphone models
Mixture splitting and re-estimation
Continue splitting?
Yes
No
Construct a multiple-mixture distribution for every leaf node
Model re-estimation
Final HMM set
(multiple-Gaussians)

Figure 3.12: New training strategy.

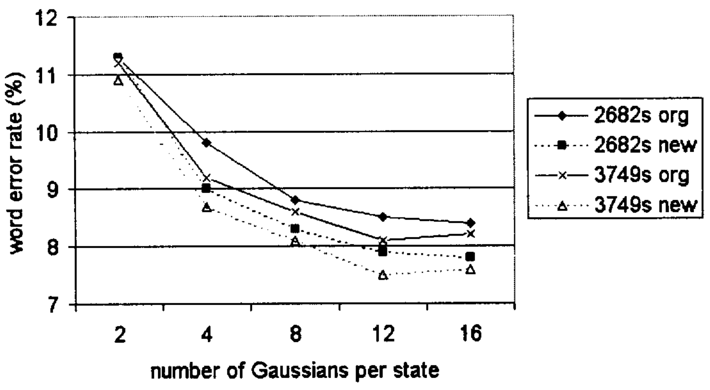


Figure 3.13: Comparison of results from the original training strategy and the new one on the WSJ 5K nov92 evaluation set.

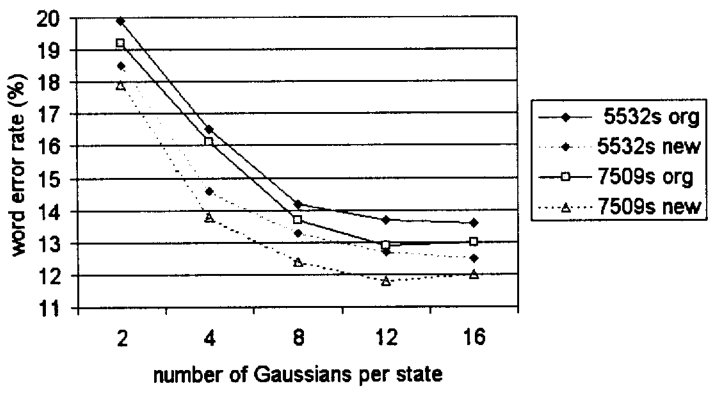


Figure 3.14: Comparison of results from the original training strategy and the new one on the WSJ 20K nov92 evaluation set.

Chapter 4

Two-level Phonetic Decision Trees

Starting from this chapter, we will present how we approach the three problems in the traditional phonetic decision-tree based acoustic modeling approach, rooted in its inefficient use of the limited training data. We propose a number of ways to address these problems. In this chapter, we begin by illustrating the first problem, caused by an implicit underlying assumption that Gaussian components and their mixture weights must be coupled together and estimated from the same set of data. Secondly, we present a new structure, called two level phonetic decision trees [37], to address this problem. Experimental results are given in Section 4.6 to demonstrate the advantages of the new approach.

4.1 Problem: Using Same Set of Data to Estimate both Gaussians and Mixture Weights

During the traditional decision tree construction, if the amount of training data (i.e. the state occupancy in Section 2.3.2) associated with all the states in a node is less than a threshold, the node is no longer split and becomes a leaf node. This is one of the stop-splitting criteria used in building phonetic decision trees. The purpose is to ensure that there will be sufficient training data in every leaf node, so that the Gaussian mixture distribution[1] for each leaf node can be robustly estimated. Otherwise, the estimated models might over-fit the training data and generalize poorly, and therefore cause performance

[1]At the time of building trees, it is only a single-Gaussian distribution. But it will be expanded to a multiple-Gaussian distribution for the final acoustic model.

loss, because too few data will not be statistically representative for a cluster of triphone states. In practice, the threshold is usually determined experimentally. In each leaf node, a Gaussian mixture model is estimated, and all the Gaussian components and the corresponding weights are shared across all the member states in the node, and no distinctions are provided among these member states.

This assumption can be viewed as one cause of inefficiently using the limited training data, and it restricts the improvement of model quality. To ensure that the Gaussian components can be robustly estimated, a relatively high threshold for the data counts (e.g. 100 or 120) is required. However, some states, particularly those states of the triphones that do not have much training data, are clustered together or with other states, not because they are acoustically very similar, but because they do not have enough training data to sustain a further split. This can be a cause of performance degradation.

To see this problem more clearly, let us look at Figure 4.1. The node **a** is one of the leaf nodes in one traditional decision tree. It contains a few triphone states that may be acoustically very different, and these states are forced to cluster together due to insufficient training data. A two-Gaussian mixture distribution is estimated to represent it. Assuming that the dimension of the feature vectors is 39 (a typical value for MFCC-based features) and the covariance matrix is diagonal, so each Gaussian component has 79 parameters (39 mean variables and 39 variance variables plus 1 mixture weight). Each two-Gaussian mixture distribution has $2 * 79 = 158$ parameters.

If we would like to distinguish those dissimilar triphone states in the node by further splitting node **a** into two child nodes **b** and **c** (using smaller minimum data count threshold), we will have to estimate a separate two-Gaussian mixture distribution for each of the child nodes. So the number of parameters will increase by 158. However, as node **a** became a leaf node because of insufficient data to support any further splitting in the original decision tree, its child nodes **b** and **c** will have even fewer training data. So, the estimated Gaussian-mixture distributions for node **b** and **c** will not be robust. As a result, contrary to the intention, the increased resolution for the training data will hurt the recognition accuracy over the test data.

However, we may overcome this difficulty by dropping an underlying assumption in the original decision trees that all elements of a Gaussian mixture distribution must be estimated from the same set of data. Figure 4.2 illustrates our idea for doing this. We still split the original leaf node **a** into node **b** and **c**, as done in Figure 4.1. However, node **b** and **c** will share the same set of Gaussian components estimated in node **a**. A different set of mixture weights, corresponding to the given set of Gaussian components, will be estimated from the training data in node **b** and **c**, respectively. In this way, the node **b** and **c** will be represented by a different Gaussian-mixture distribution, and therefore the resolution is increased. As estimating mixture weights (given the Gaussian components in node **a**) requires much fewer training data, the new sets of mixture weights in node **b** and **c** can still be robustly estimated.

The two new nodes **b** and **c** only add 2 parameters to the original system, rather than 158 as shown in the Figure 4.1. Usually the number of Gaussian components in a node is more than 2 (our baseline system has 12 Gaussian components per state). If we have 10 Gaussian components in a node, the increased number of parameters by this new approach is 10, while by the traditional approach it will be 790!

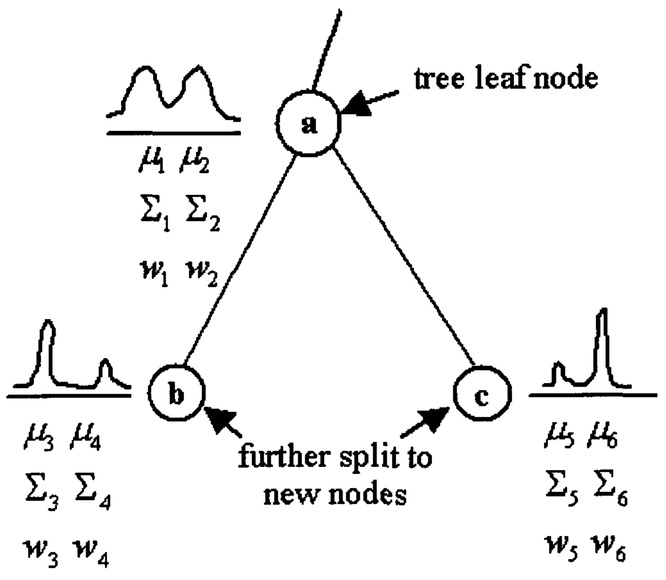


Figure 4.1: Traditional approach: Gaussian components and mixture weights are coupled together. A different Gaussian mixture distribution needs to be estimated for node b and for node c, using their own data.

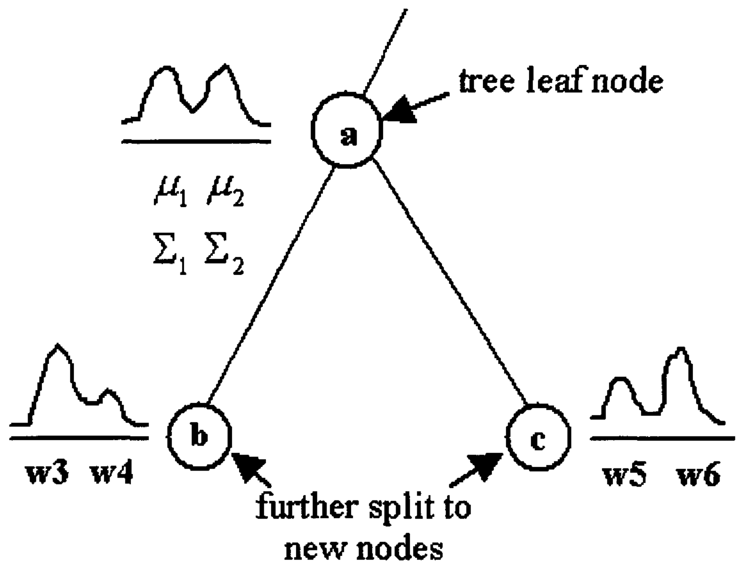


Figure 4.2: New approach: Gaussian components and mixture weights are decoupled. Node b and c share the same set of Gaussian components in node a. Only the mixture weights are estimated in node b and c.

4.2 Two-level Phonetic Decision Trees

To apply this idea, we proposed a new structure called a two-level decision tree. Figure 4.3 illustrates the two-level decision tree structure. It looks like a regular phonetic decision tree. However, the difference is that it has two levels (here "level" means the type of leaf nodes resulting from different tree-building processes). First, a decision tree (called a first-level tree) is built using the traditional approach; all the states in the same first-level leaf node share the same Gaussian pool. We call the first level the Gaussian-sharing level. Then, the second-level nodes are obtained by further splitting the first-level leaf nodes. Smaller thresholds (both minimum data count threshold and likelihood increment threshold) than those used in the first-level nodes are used in order to grow the second-level nodes.

All the second-level leaf nodes that come from the same first-level leaf node will share the same Gaussian pool, but they have their own set of weights for the Gaussians. All the states in the same second-level leaf node will share their mixture weights. We call the second level the weight-sharing level. Of course, they also share the corresponding Gaussian set in their ancestor node that is a first-level leaf node. So all states in the second-level leaf node are identical.

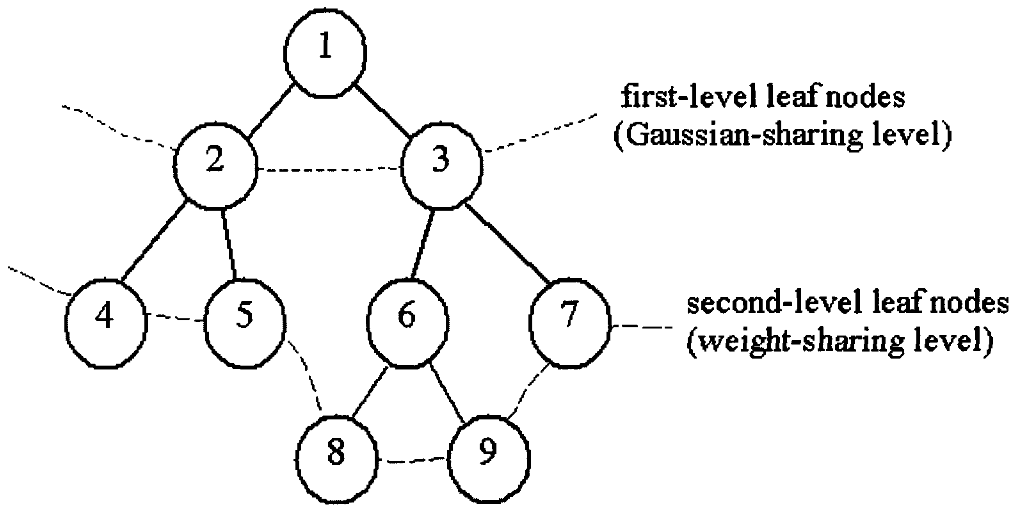


Figure 4.3: A two-level decision tree. Node 2 and 3 are first-level leaf nodes. Node 4, 5, 8, 9 and 7 are second-level leaf nodes.

If we do not expand the first-level leaf nodes, the tree is the same as a traditional decision tree (for contrast, we refer to it as a one-level tree), in which all states in the same leaf node share not only Gaussians but also corresponding mixture weights. By using the second-level nodes, better resolution can be achieved among those states in the same first-level leaf node by distinguishing their mixture weights, especially for those rarely seen triphone states that have to be clustered together due to the relatively high data count threshold in the first-level tree.

By controlling the number of second-level nodes, the increased number of mixture weights is only a very small percentage of total parameters. For example, if each first-level leaf node is split into only two second-level nodes, the two-level tree only adds about 1/79 of the total number of parameters. The increased number of mixture weights still can be robustly estimated, since estimating mixture weights requires very few training data given a robustly estimated Gaussian pool in the first-level leaf node (actually they are jointly estimated in order to obtain optimal values). But in contrast, if we continue to use the traditional approach in the second-level leaf nodes in order to get higher resolution for those states in the first-level leaf nodes, the number of parameters will double, as a set of Gaussians and corresponding mixture weights (assuming the same number as in the parent node) are added for each extra second-level leaf node. This will obviously result in too many parameters and over-fitting. As a result, the performance of such a system will degrade.

By de-coupling the Gaussians and the corresponding mixture weights in the traditional decision tree and creating a two-level tree structure, we can make better use of the limited training data, especially for those rarely seen triphones.

4.3 Algorithm to Construct Two-Level Phonetic Decision Trees

The algorithm to build two-level decision trees is based on the algorithm for building one-level trees, as shown in Figure 3.6. After the first level tree building procedure stops, every leaf node is labeled as a first-level leaf node. And then it continues splitting using smaller thresholds and the same goodness-of-split criterion and the same steps (i.e., splitting first and then merging similar nodes) in building the first level trees. Due to the merging steps in both levels, the final structure is not a real tree, but rather a complicated graph or network.

The Baum-Welch re-estimation formula (in Section 2.2.3) can directly apply to estimating HMM parameters of two-level tying without any change. With different clustering of Gaussian components and mixture weights, the sets of speech observation vectors, contributing to the Gaussian components and to the mixture weights, will be different.

4.4 Tuning of Two-Level Phonetic Decision Trees

So far, there has been an implicit assumption that two-level trees are built by expanding the optimal-sized one-level trees and thereby the two-level tree based acoustic models have more parameters than the one-level tree based models. However, it is not necessary to construct the tree in this way. In practice, we found that by tuning the balance of the first-level and second-level leaf nodes, we can get better performance with even fewer parameters than an optimal-sized one-level tree system.

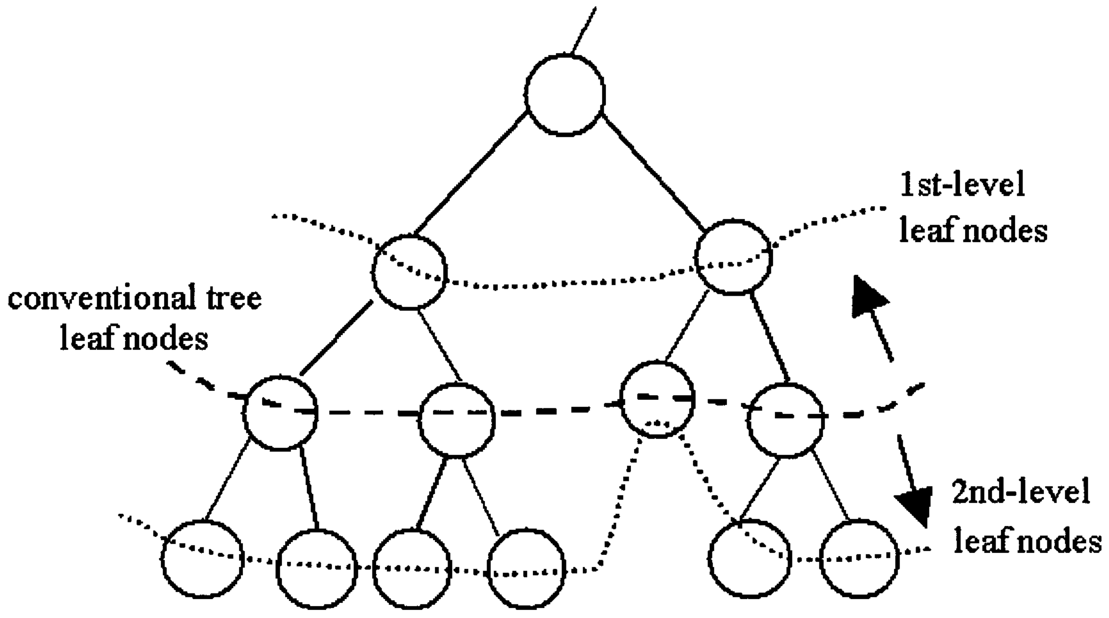


Figure 4.4: Tuning of a two-level phonetic decision tree. By pruning back the one-level tree leaf nodes to the first level, better robustness for Gaussian sets can be obtained. By further splitting one-level tree leaf nodes to the second level, high accuracy for mixture weights can be otained. Overall, both robustness and accuracy may be improved.

Normally, a global threshold of data count is used for building all the trees, and the value is tuned until the final model set based on the decision tree clustering achieves best performance on a development set. Even with this global optimal value, it is still possible that some leaf nodes are under-trained and should be further split to get higher resolution, and some leaf nodes are over-trained and should be pruned back (although this problem is lessened by allowing a variable number of Gaussian components in each clustered state in the final model set). This problem can be addressed in the two-level decision tree structure because it provides more flexibility to control the total number of parameters. By using a relatively conservative value of the data count threshold (e.g. 150 frames) in the first-level

(Gaussian-sharing level) tree, we can ensure that the Gaussian pools in all of the leaf nodes can be robustly estimated. And by using the second-level (weight-sharing level) nodes, we can obtain high resolution. Although a global data count threshold (smaller than the one used in the first-level tree, e.g., 30 frames) is still used, the impact of this problem is smaller since it only affects the mixture weights.

4.5 Significance Test

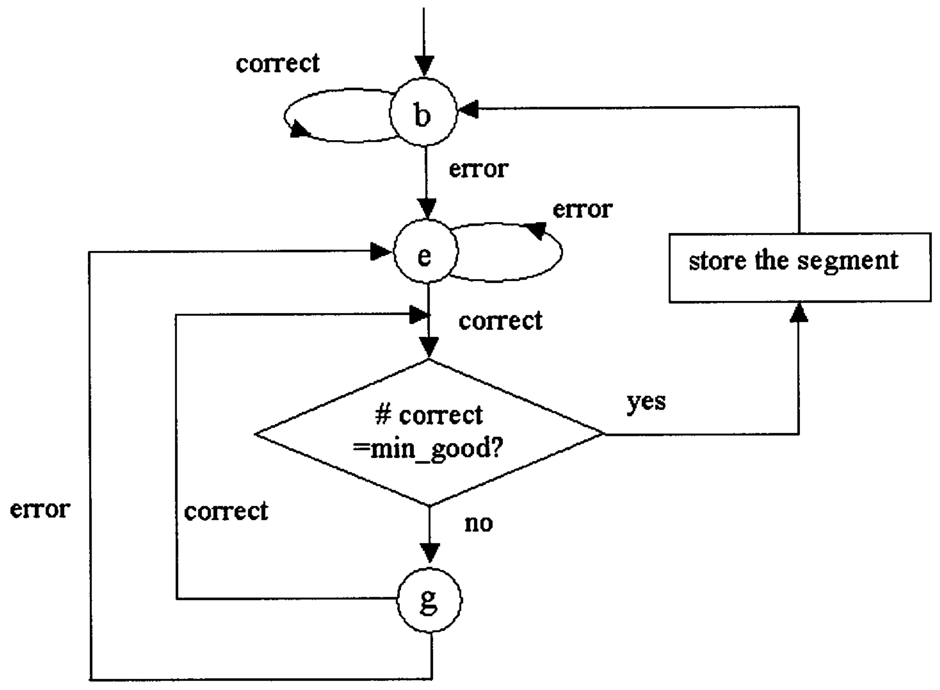
Before we present the experimental results of the two-stage decision tree based systems, it is necessary to introduce the concept of a statistical significance test. It can be used to perform comparisons on speech recognition algorithms (or systems) by comparing the recognition results on the test data set and by measuring whether the difference in performance is statistically significant. The statistical significance tests we use in this thesis determine the probability that a certain given result will occur entirely by chance, even when error rates for two systems are identical. It was orginally developed for DARPA speech recognition benchmark tests, and now it is widely accepted in the general speech recognition research community. The NIST[2] standard scoring package provides four significance tests, two of which are most often used (also used in this thesis) and will be briefly explained below.

Matched Pairs Sentence-Segment Word Error (MAPSSWE) Test

MAPSSWE test [20] [21], sometimes simply called matched-pairs test, is a parametric test that looks at the numbers of errors occuring in units that are larger than single words and smaller than entire utterances. The units, called sentence segments, are chosen in a way to approximately validate the independence assumption. The segments are bounded on both sides by words correctly recognized by both systems under test, or the beginning and end of utterances. Because the number of units is large, the central limit theorem permits the approximate assumption that the average number of errors per segment are normally distributed. The sentence segments are detected using a state machine illustrated

[2]stands for National Institute of Standards and Technology

in Figure 4.5.



state b: Have not found any error yet.
state e: If both systems are correct, then check to see if number of correct words (# correct) equals to the minimum (min_good). If it is, then mark the segment and go to state b.
state g: If next word is correct, increase # correct and loop back to do the check. Otherwise, go to state e.

Figure 4.5: State machine for locating sentence segments.

The term "correct" means that both of the two systems correctly recognize the current word. The term "error" means that at least one system incorrectly recognizes the current word. A sentence segment is thus a sequence of words that ends with a given number (*min_good*) of correctly recognized words for both systems. The value *min_good* is set to one in this thesis. Here is an example of detected sentence segments:

	I		II		III		IV	
REF:	it was	the best	of	times it	was the worst	of times		it was
SYS A:	ITS	the best	of	times it	IS the worst	of times	OR	it was
SYS B:	it was	the best	—	times it	WON the TEST	of times		it was

There are four segments detected by the state machine. For segments I and IV, A is incorrect and B correct (a substitution and a deletion in I, and an insertion in IV). For segment II, A is correct and B incorrect (a deletion). For segment III, both are incorrect (one substitution in A, two in B).

For each segment i, define d_i as the difference of the number of misrecognized words from the two systems. The hypotheses of the matched pairs test are as follows.

$$\textit{The null hypothesis } H_0 \; : \; \bar{d} = 0$$

$$\textit{The alternative hypothesis } H_a \; : \; \bar{d} \neq 0$$

where $\bar{d}$ is the mean of the differences, $\bar{d} = \sum_{i=1}^{n} d_i/n$. n is the total number of segments.

The test statistic is defined as $z = \sqrt{n}\bar{d}/\bar{\sigma}$, where $\bar{\sigma}$ is the estimated standard deviation, $\bar{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2$. The decision rule of the matched pairs test is therefore: reject H_0 if $|z| > z_\alpha$, where z_α is a critical value [41] from a standard normal table corresponding to the confidence level $100(1-\alpha)\%$. When the confidence level is 95% ($\alpha = 0.05$), $|z_\alpha|$=1.96.

A matched-pairs test is generally more powerful than other tests like Wilcoxon test, due to its inherent large number of units (as it uses smaller units, sentence segments rather than whole utterances). It is not usual that other tests reject the null hypothesis while a matched-pairs test does not.

Wilcoxon Signed Rank Test

The Wilcoxon signed rank test [53] is a non-parametric test that utilizes information on both the signs and the magnitudes of the performance differences in two systems. The NIST implementation is one variant of the standard Wilconxon signed rank test. It uses word accuracy as the measurement of performance. The hypotheses of the test are as follows.

The null hypothesis:

The two populations represented by the respective matched pairs are identical.

The alternative hypothesis:

The two populations are not identical and there is a difference between them.

The procedure to calculate the test statistic for the Wilcoxon test is:

1. Calculate the differences of the word accuracy rates of speaker i of the two systems and denote it as d_i.

2. Rank the absolute values of the differences, $|d_i|$, by assigning 1 to the smallest, 2 to the second smallest, and so on. Tied observations are assigned the average of the ranks that would have been assigned with no ties.

3. Calculate the rank sum for the positive differences and label this value as T_+. Similarly, calculate T_-, the rank sum for the negative differences.

For large enough n (≥ 8), T_+ has an approximately normal distribution. Its mean and variance are

$$\mu = \frac{n(n+1)}{4}$$
$$\sigma^2 = \frac{n(n+1)(2n+1)}{24}$$

Then the z statistic

$$z = \frac{T_+ - \mu}{\sigma}$$

can be used as a test statistic. The decision rule for the Wilcoxon test is that, based on a 95% ($\alpha = 0.05$) confidence interval, the null hypothesis is rejected when $|z| > 1.96$.

4.6 Experimental Results

To build acoutic models using two-level decision tree based clustering, we used the same procedure as in building the baseline systems. All the parameters and thresholds in the training procedure (except those used in building two-level decision trees) are kept the same to ensure fair comparison. The thresholds for building two-level decision trees are optimized on the development set.

Table 4.1: Results (WER %) of acoustic models built with different sizes of two-level decision trees on two WSJ 5K test sets (nov92/si-dt-05). The first row of each block has the same number of leaf nodes in the first level and second level. They correspond to the baseline system in Section 3.3.

# of 1st-level leaf nodes	# of 2nd-level leaf nodes	2-mixture	4-mixture	8-mixture	12-mixture	16-mixture
2682	2682	11.3/16.9	9.0/12.5	8.3/10.8	7.9/9.9	7.8/9.6
2682	11264	11.0/16.8	8.8/12.2	8.1/10.5	7.7/9.7	7.7/9.4
2682	21111	10.9/16.7	8.6/12.2	7.9/10.2	7.5/9.6	7.4/9.4
3364	3364	11.0/16.8	8.8/12.7	8.3/10.8	7.7/9.7	7.7/9.2
3364	11457	10.8/16.6	8.4/12.2	7.5/10.2	6.8/8.5	6.5/8.2
3364	21268	10.5/16.5	8.1/12.0	7.1/9.8	**6.4/8.4**	6.4/8.3
3749	3749	10.9/16.6	8.7/12.5	8.1/10.7	**7.5/9.5**	7.6/9.5
3749	11607	10.7/16.6	8.5/12.3	7.8/10.2	7.3/9.2	7.3/9.0
3749	21367	10.5/16.4	8.4/12.0	7.4/10.0	7.1/8.9	7.1/8.9
4656	4656	10.7/16.2	8.4/11.7	8.0/10.5	7.5/9.8	7.8/9.9
4656	11682	10.4/16.0	8.1/11.3	7.6/10.0	7.2/9.3	7.2/9.2
4656	21528	10.2/15.7	7.9/11.0	7.5/9.9	7.2/9.0	7.4/9.1

The best performances for the standard one-level tree system were given in Section 3.3. For the purpose of comparison, they are listed again here. Results for two-level tree systems on the WSJ 5K test sets are given in Table 4.1. Some selected results are plotted in Figure 4.6. Both thresholds for the minimum data count and log-likelihood increase in building the second-level tree are smaller than those used in building the first-level tree.

We can see from the experimental results that

- When further splitting one-level trees to become two-level trees, the two-level tree systems always outperform the corresponding one-level tree systems. But the two-level tree systems have more parameters than the one-level tree systems.

- When pruning back the first-level leaf nodes to have fewer number of Gaussians than the corresponding one-level tree systems, we can generally get better or equivalent results by using more second-level leaf nodes (than the one-level tree leaf nodes). And the two-level tree systems have fewer parameters than the one-level tree systems.

Table 4.2: Comparison of the best results of the baseline system (one-level tree) and the two-level tree based system on two WSJ 5K test sets. MP means the matched-pairs test.

test set	one-level WER	two-level WER	reduction of WER	reduction of # of param.	significance	
					Wilcoxon	MP
nov92	7.5%	6.4%	15%	5%	yes	yes
si-dt-05	9.5%	8.4%	12%	5%	yes	yes

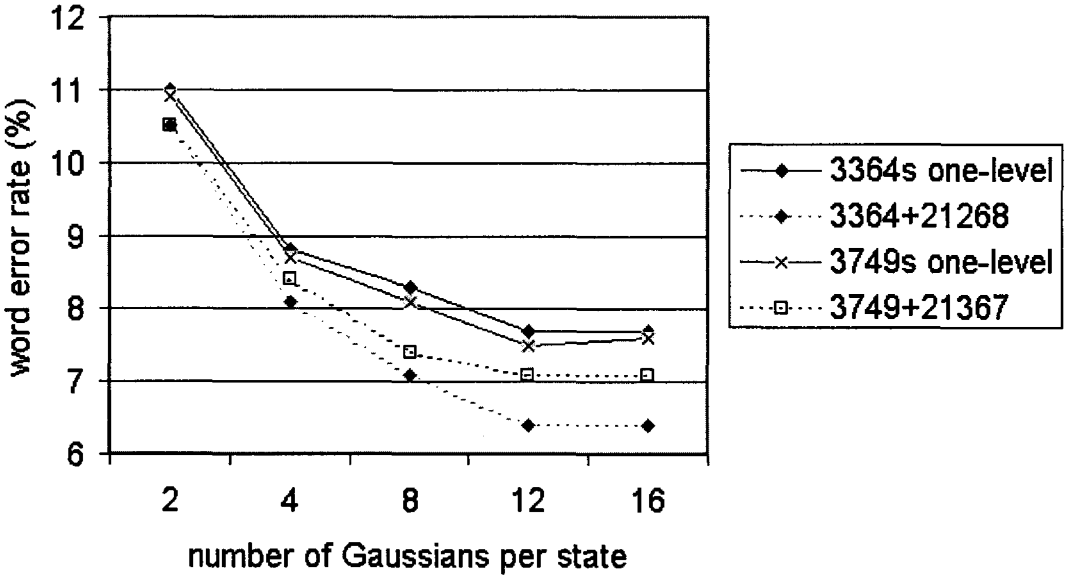


Figure 4.6: Results of the baseline and the two-level tree systems with different model sizes on the WSJ 5K nov92 test set.

- The best two-level tree system gives lower word error rates than the best one-level tree system. And furthermore, the two-level tree system has fewer number of parameters. Table 4.2 gives a comparison of the best one-level tree and two-level tree systems.

The best one-level tree system has 3749 states (or leaf nodes) and 12 Gaussian components per state. The minimum data count is 120. The best two-level tree system has a total of 3364 first-level leaf nodes and 21268 second-level leaf nodes. Each first-level node has 12 Gaussian components. The minimum data counts for building the first-level and second-level trees are 180 and 40, respectively. There are a total of 3.6 million parameters in the one-level tree system and 3.4 million parameters in the two-level tree system. With a 5% reduction in the number of parameters, 15% and 12% word error rate reductions are obtained for the two test sets. The significance tests show that this is a statistically significant improvement.

Table 4.3: Word error rates (%) and number of parameters (in brackets, in unit of million) of different acoustic models on the WSJ 5K combined test set (nov92 +si-dt-05).

# of 1st level leaf nodes	# of 2nd level leaf nodes	2-mix	4-mix	8-mix	12-mix	16-mix
2682	2682	14.50(0.42)	11.00(0.85)	9.73(1.70)	9.04(2.54)	8.83(3.39)
2682	11264	14.31(0.44)	10.74(0.88)	9.47(1.76)	8.84(2.65)	8.67(3.53)
2682	21111	14.21(0.46)	10.65(0.92)	9.21(1.84)	8.70(2.76)	8.54(3.68)
3364	3364	14.31(0.53)	11.0(1.06)	9.73(2.13)	8.84(3.19)	8.56(4.25)
3364	11457	14.11(0.55)	10.57(1.10)	9.04(2.19)	7.77(3.29)	7.47(4.38)
3364	21268	13.92(0.57)	10.33(1.13)	8.64(2.27)	**7.54(3.40)**	7.48(4.54)
3749	3749	14.15(0.59)	10.87(1.18)	9.58(2.37)	**8.64(3.56)**	8.68(4.74)
3749	11607	14.07(0.61)	10.67(1.22)	9.17(2.43)	8.38(3.65)	8.27(4.86)
3749	21367	13.87(0.63)	10.45(1.26)	8.88(2.51)	8.13(3.77)	8.13(5.02)
4656	4656	13.84(0.74)	10.28(1.47)	9.43(2.94)	8.81(4.41)	8.99(5.89)
4656	11682	13.60(0.75)	9.93(1.50)	8.97(3.00)	8.39(4.50)	8.34(6.00)
4656	21528	13.34(0.77)	9.67(1.54)	8.87(3.08)	8.23(4.62)	8.37(6.16)

To help compare the one-level and two-level tree systems in terms of both performance and model size, we convert the Table 4.1 to Table 4.3, which gives results on the combined test set (nov92+si-dt-05) of the WSJ 5K task, together with the number of parameters for every acoustic model. Two comparisons are conducted:

1. Given a certain performance level of the one-level tree system, how much model size reduction can be obtained by two-level tree systems?

2. Given a certain level of model size, how much performance improvement can be achieved by the two-level tree system over the one-level tree system?

The result of the first comparison is given in Table 4.4. We can see that at the same performance level of the one-level tree system, the two-level tree system can reduce the model size by as much as 36%.

Table 4.4: Comparison of model sizes for one-level tree systems and two-level tree systems with fixed word error rates on the WSJ 5K combined test set (nov92+si-dt-05).

WER	one-leve tree system size(million)	two-level tree system size(million)	reduction in # of parameters
8.64%	3.55 (3749s, 12mix)	2.27 (3364+21268, 8mix)	36%
8.84%	3.19 (3364s, 12mix)	2.65 (2682+11264, 12mix)	17%
9.04%	2.54 (2682s, 12mix)	2.19 (3364+11457, 8 mix)	14%

For the second comparison, there are very few pairs of results available due to the limited experimental results. But we can see that when the number of parameters is fixed to be the same as the best two-level tree system, that is, 3.4 million, the best available result for one-level tree systems is 8.83% (2682 states, with 16 mixture per state), slightly worse than the best baseline system (8.64%). With this model size, the two-level tree system reduces the word error rate of the one-level tree system by 14.6% on the combined test set.

For the WSJ 20K task, experimental results are given in Table 4.5 and Table 4.6. For comparison convenience, two pairs of models are selected from the table and plotted in Figure 4.7. Similar conclusisons as those for the WSJ 5K task can also be drawn from the analysis of the results.

The one-level tree system has 7509 states and 12 Gaussian components per state. The minimum data count is 120. The two-level tree system has a total of 6542 first-level leaf nodes and 22234 second-level leaf nodes. Each first-level leaf node has up to 12 Gaussian components. The minimum data counts used in the construction of the first level and second level are 180 and 40, respectively. There are a total of 7.1 and 6.4 million parameters in the one-level tree system and two-level tree system, respectively. With a 10% reduction in the number of parameters, we obtained a 12% word error rate reduction, which is a statistically significant improvement over the baseline system.

Table 4.7 gives the comparison of model sizes between one-level and two-level tree

Table 4.5: Word error rates (%) and number of parameters (in brackets, in unit of million) of different acoustic models on the WSJ 20K nov92 test set. The first row of each block has the same number of leaf nodes in the first level and second level. They correspond to the baseline system in Section 3.3.

# of 1st-level leaf nodes	# of 2nd-level leaf nodes	2-mix	4-mix	8-mix	12-mix	16-mix
5532	5532	18.5(0.87)	14.6(1.74)	13.3(3.50)	12.7(5.24)	12.5(6.99)
5532	22098	17.8(0.91)	14.0(1.81)	12.2(3.63)	11.3(5.44)	11.2(7.26)
5532	32356	17.7(0.93)	13.8(1.86)	12.4(3.71)	11.2(5.56)	11.4(7.42)
6542	6542	17.8(1.03)	14.0(2.06)	12.9(4.13)	12.4(6.20)	12.4(8.27)
6542	22234	17.1(1.07)	12.9(2.13)	11.4(4.26)	**10.4(6.39)**	10.3(8.52)
6542	32508	17.0(1.09)	13.0(2.17)	11.4(4.34)	10.5(6.51)	10.4(8.68)
7509	7509	17.9(1.19)	13.8(2.37)	12.6(4.75)	**11.8(7.12)**	12.0(9.49)
7509	22366	17.1(1.22)	13.0(2.43)	11.8(4.86)	10.9(7.30)	10.9(9.73)
7509	32782	16.8(1.24)	12.9(2.47)	12.0(4.95)	10.9(7.42)	11.1(9.90)

Table 4.6: Comparison of the best results of the baseline system (one-level tree) and the two-level tree based system on the WSJ 20K task. MP means the matched-pairs test.

test set	one-level WER	two-level WER	reduction of WER	reduction of # of param.	significance	
					Wilcoxon	MP
nov92	11.8%	10.4%	12%	10%	yes	yes

systems. We can see that at the same performance level of the one-level tree system, the two-level tree system can reduce the model size by up to 40%.

The experimental results confirm that using the proposed two-level decision tree based acoustic modeling is advantageous over the traditional decision-tree based acoustic modeling. It not only reduces word error rates but also yields a smaller acoustic model than the traditional approach, as a result of better use of training data.

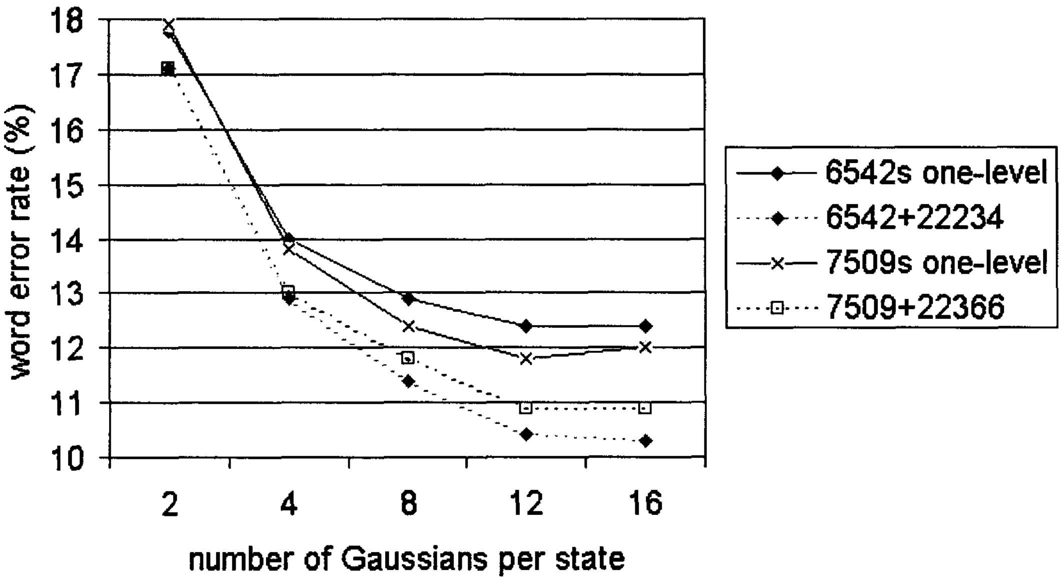


Figure 4.7: Results of the baseline and the two-level tree systems with different model sizes on the WSJ 20K nov92 test set.

Table 4.7: Comparison of one-level tree systems and two-level tree systems with fixed word error rates on the WSJ 20K nov92 set.

WER	one-leve tree system size(million)	two-level tree system size(million)	reduction of # of parameters
11.8%	7.12 (7509s, 12mix)	4.86 (7509+22366, 8mix)	32%
12.4%	6.20 (6542s, 12mix)	3.71 (5532+32356, 8mix)	40%

Chapter 5

Two-Stage Decision Tree Building

In this chapter, we will address a different problem with the conventional phonetic decision tree based acoustic modeling, also rooted in its inefficient use of data for triphones that have very few training data (called *rarely-seen* triphones). It is related to the parameter estimation of unclustered rarely-seen triphones. First, we will explain the problem in the conventional approach in Section 5.1. Some related work will be presented in Section 5.2. Next, we will present our approaches based on a two-stage decision tree building strategy [38] and on a MAP-based smoothing technique in Section 5.3 and Section 5.4, respectively. Experimental results are given in Section 5.5, which clearly demonstrate the advantages of our approaches over the conventional approach.

5.1 Problem: Waste of Data for Rarely-Seen Triphones

In practice, no matter how large a corpus is, the training data are usually distributed unevenly in terms of occurrence of triphones. Table 5.1 lists the distribution of training data in the WSJ 5K task in percentages. We can see that as many as 12.8% of all 18,532 distinct seen triphones only occur once, and 33% of all distinct seen triphones occur no more than 3 times in the training data.

For those triphones that have very few training exsamples, the Gaussian distributions associated with them can be poorly estimated. When they directly participate in the tree construction, the errors in estimating the statistics of the rarely seen triphones can have a long-term adverse effect on the quality of the decision-tree based state clustering. It is very important to obtain a better estimate for the rarely seen triphones given very limited training data in order to minimize the negative effects of the non-robustly estimated statistics of the triphones to decision tree building.

One might think that we could use statistics for those frequently-seen triphones to build decision trees to minimize the negative effect caused by those rarely seen triphones. However, in this way, we also throw away the information contained in the training data for the rarely seen triphones. The traditional approach uses a simple back-off mechanism to prevent unreliable statistics of rarely seen triphone models from causing bad effects in the decision tree building. If the amount of training data for a triphone model is less than a given threshold, the model is simply backed-off to the corresponding monophone model[1]. This is not a good solution, because the data (although very limited, they still contain useful information) for those backed-off triphones are wasted. Their contribution to the tree construction is minimized. Even though each individual rarely seen triphone occurs so infrequently that, by itself, it doesn't make much difference in the tree construction, there are enough rarely seen triphones that together they make a significant difference. Therefore, it is necessary to find a better solution to address this problem associated with rarely seen triphones.

Table 5.1: Distribution of triphone occurrences in the WSJ 5K training data.

number of occurences	1	2	3	4	5	6-10	10+
percentage of triphones	12.8%	12.2%	8.3%	6.7%	4.7%	15.3%	40%

5.2 Related Work

Reichl and Chou [47] proposed an approach to address this, in which the rarely seen triphones were clustered into various types of *generalized triphones* [34] before building decision trees. The rarely seen triphones are clustered by relaxing the triphone contexts.

[1]For triphones that are never seen in the training data, no models will be built for them during training. Instead, they will be predicted by using the decision trees during decoding. The statistics of rarely seen triphones as well as other seen triphones will help predict the never seen triphones.

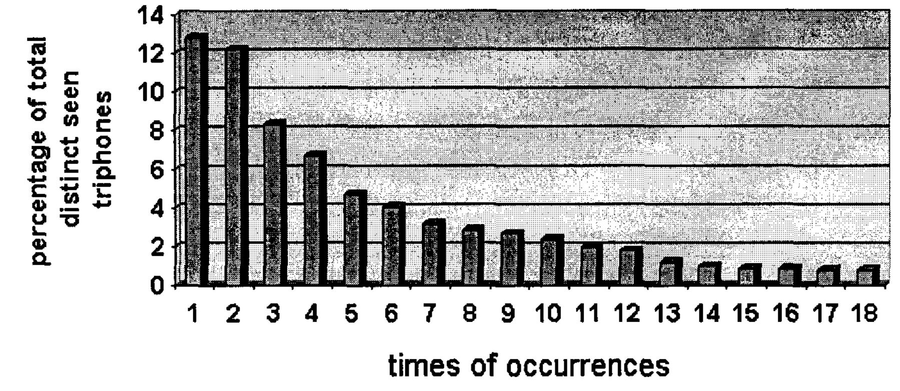


Figure 5.1: Distribution of triphone occurrence in the WSJ5K training data.

First, the left contexts of all rarely seen triphones that are derived from a base phone are relaxed, that is, the rarely seen triphones that share the same right context are grouped together, if the amount of training data is above the threshold. Each of these groups is a generalized triphone. For all remaining unclustered triphones, if there are any, the right contexts of them are relaxed. If there are still remaining unclustered triphones, then both left and right contexts are relaxed, that is, they are reduced to the corresponding monophone models. When these generalized triphones participate in the subsequent building process of the regular phonetic decision trees, the phonetic identity for each of them is the intersection of the phonetic properties of all rarely seen triphones in each cluster (generalized triphone). Figure 5.2 gives an example of clustering rarely seen triphones to generalized triphones.

This approach is simple and easy to implement. However, we believe it is inadequate for clustering rarely seen triphones, as it does not take into account any similarity among the underlying data for those triphones (although there are not enough data, they still provide some information about each unique triphone) nor any similarity among the phonetic identities of those triphones. Furthermore, an assumption was made in the approach that a rarely seen triphone can only be clustered with other rarely seen triphones, even if it is more similar to a triphone that has sufficient training data. Obviously this assumption is not appropriate.

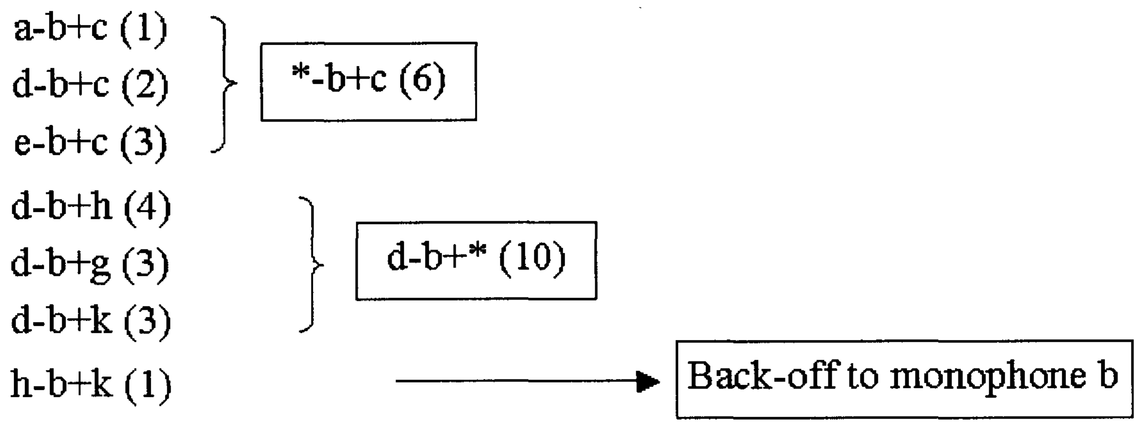


Figure 5.2: Clustering rarely seen triphones into different generalized triphones. The numbers in the parentheses are times of occurrences in the training data. The threshold for clustering is 5 occurrences.

5.3 Two-Stage Phonetic Decision Tree Building

In contrast to Reichl's approach, we drop his assumptions and cluster rarely seen triphones based on both distance measurements and phonetic knowledge. We build the phonetic decision trees in two stages as shown in Figure 5.3.

In the first stage, we try to cluster rarely seen triphones by using phonetic decision trees. Initial statistics of all unclustered triphones are obtained after several iterations of training. Very big decision trees are built using these statistics, in which the rarely seen triphones are clustered with their most similar peers, which may or may not be rarely seen triphones, so that the resulting clusters have sufficient data to estimate robust representative Gaussian distributions. But any frequently-seen triphone is kept as unique, even if some rarely seen triphones are clustered with it. In other words, the clustered rarely seen triphones use pooled data to estimate their statistics (so they are smoothed), while the frequently-seen triphones still use their own data to estimate their statistics.

In the second stage, we first update the statistics for the resulting triphones from the first stage clustering, using the Baum-Welch algorithm. These statistics are then used

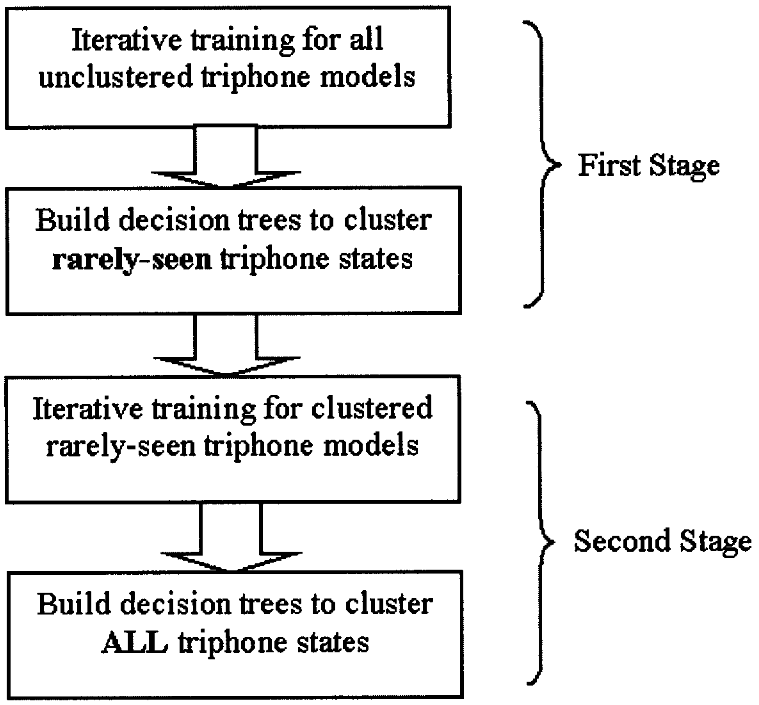

Iterative training for all unclustered triphone models
Build decision trees to cluster **rarely-seen** triphone states
Iterative training for clustered rarely-seen triphone models
Build decision trees to cluster ALL triphone states
First Stage
Second Stage

Figure 5.3: Two-stage phonetic decision tree building.

to build the final decision trees. Compared to the back-off mechanism in the traditional approach, the limited training data for the rarely seen triphones are fully exploited to estimate their statistics, so the rarely seen triphones may be better estimated, and therefore better final decision trees may be constructed in this stage.

Because both distance similarity and phonetic similarity are considered in clustering the rarely seen triphones using phonetic decision trees in the first stage, better clustering might be obtained, compared to using generalized triphone method. In Reichl's approach, if there are many rarely seen triphones (derived from the same monophone) that share the same, say right, context, they will be simply grouped all together. This might cause unnecessary over-clustering, and reduce model accuracy. In our approach, we can easily control the clustering degree by specifying the data count threshold in building decision trees in the first stage. So it has more flexibility, and the limited training data are used more efficiently. As the decision tree based clustering in the first and second stage use the same criterion, it might also be more consistent.

The conventional back-off approach is still used when estimating the statistics of unclustered triphone models for building decision trees in the first stage. So the trees built in the first stage may not be optimal. However, the estimation of rarely seen triphones still benefits from the clustering and predicting capability reflected in the tree structure, which is learned from the knowledge available contained in the data of those frequently-seen triphones. With the improved clustering of the rarely seen triphones and therefore better estimated statistics, we can expect the trees built in the second stage will also be better constructed.

5.4 Maximum A Posteriori Smoothing of Context-Dependent Models

The approach in the previous section can be regarded as a parameter sharing technique. There is another class of techniques called parameter smoothing, to address the problem of data sparseness in parameter estimation. Examples of this technique are Maximum A Posteriori (MAP) [19], a well-known parameter estimation criterion, and deleted interpolation [26]. The deleted interpolation is generally only applicable to discrete HMMs or semi-continuous HMMs, and the procedure is more complicated. In this thesis, we propose to use MAP-based smoothing technique to obtain robust estimation of unclustered triphone models for decision tree building. It will be compared against the traditional approach and the two-stage decision-tree based approach. An introduction to MAP technique will first be given in Section 5.4.1. In Section 5.4.2, we will derive a simplified MAP estimation formula for Gaussian mixtures, which will be more easily understood and implemented.

5.4.1 Maximum A Posteriori

The MAP framework provides a way of incorporating prior information in the estimating process, which is particularly useful in dealing with problems posed by sparse training data, for which Maximum Likelihood (ML) gives inaccurate estimates. The difference between MAP and ML estimation lies in the assumption of an appropriate prior distribution of the parameters to be estimated. If λ, which is assumed to be a random vector, is the parameter vector to be estimated from the observation O with the probability density funtion (pdf) $f(O|\lambda)$, given a prior pdf of λ, $g(\lambda)$, then the MAP estimate is defined as the maximum of the posterior pdf of λ, that is,

$$\lambda_{MAP} = \arg\max_{\lambda} P(\lambda|O) = \arg\max_{\lambda} f(O|\lambda)g(\lambda)$$

If λ is assumed to be fixed but unkown, then there is no knowledge about λ. This is equivalent to assuming a non-informative prior, i.e., $g(\lambda)$ is constant. Under such assumptions, MAP reduces to ML.

Given the MAP formulation, there are three key issues to be addressed: the choice of a parametric form for the prior, the estimation of the prior parameters and the evaluation of the MAP. In fact, these issues are closely related, since the choice of an appropriate prior distribution can greatly simplify the estimation of the maximum a posteriori. If there exists a sufficient statistic of a fixed dimension, there must exist a standard family of distributions of the parameter λ, which has the following property:

- If the prior distribution of $g(\lambda)$ belongs to that family, then for any sample size and any values of the observation in the sample, the posterior distribution of λ must also belong to the same family.

A family of distributions with this property is said to be closed under sampling and is called a *conjugate* family of distributions.

The main distribution in an HMM is the state output function, which is a multivariate Gaussian mixture. However, to simplify the presentation, we assume here a univariate Gaussian mixture distribution, and adopt the same notation in [17].

$$P(x|\theta) = \sum_{k=1}^{K} \omega_k N(x|m_k, r_k)$$

where $\theta = (\omega_1, \ldots, \omega_K, m_1, \ldots, m_K, r_1, \ldots, r_K)$, ω_k is the mixture weight, m_k the mean and $r_k = 1/\sigma_k^2$ the precision of the k-th Gaussian component. K is the number of Gaussian components in a state.

For such a probability density function, there exists no sufficient statistic of fixed dimension for θ and therefore no conjugate distribution. Gauvain [17] proposed to use a prior joint probability density, which is the product of a Dirichlet density and a gamma-normal density:

$$P(\theta) \propto \prod_{k=1}^{K} \omega_k^{\lambda_k - 1} r_k^{1/2} exp(-\frac{\tau_k r_k}{2}(m_k - \mu_k)^2) r_k^{\alpha_k - 1} exp(-\beta_k r_k)$$

The choice of this prior density can be justified by the fact that

- The Dirichlet density is the conjugate distribution of the multinomial distribution (for the mixture weights).
- The gamma-normal density is the conjugate density of the normal distribution (for the mean and precision parameters).
- The parameters of the individual mixture components and the mixture weights are assumed to be independent.

If we assume two regularity conditions, (1) $\lambda_k = \tau_k$; and (2) $\alpha_k = (\tau_k + 1)/2$, then the MAP estimates of HMM parameters for the state output distribution can be derived, using the EM algorithm, as following [18]:

$$define\ \ c_{ik} = \frac{\omega_k N(x_i | m_k, r_k)}{\sum_{k=1}^{K} \omega_k N(x_i | m_k, r_k)} \tag{5.1}$$

$$\omega_k' = \frac{\lambda_k + \sum_{i=1}^{n} c_{ik}}{n + \sum_{l=1}^{K} \lambda_l} \tag{5.2}$$

$$m_k' = \frac{\tau_k \mu_k + \sum_{i=1}^{n} c_{ik} x_i}{\tau_k + \sum_{i=1}^{n} c_{ik}} \tag{5.3}$$

$$r_k' = \frac{2\alpha_k - 1 + \sum_{i=1}^{n} c_{ik}}{2\beta_k + \sum_{i=1}^{n} c_{ik}(x_i - m_k')^2 + \tau_k(\mu_k - m_k')^2} \tag{5.4}$$

It is straightforward to generalize them to a multivariate Gaussian mixture distribution. Since the covariance matrix in each Gaussian is assumed to be diagonal, each dimension of the parameter vectors can be estimated separately by using the above formula.

The parameters for the prior density were estimated as follows:

$$\beta_k = \frac{\tau_k}{2r_k} \tag{5.5}$$

$$\mu_k = m_k \tag{5.6}$$

5.4.2 Interpretation of MAP Formula

To better understand the MAP estimation formula, we make a further derivation as follows:

$$\omega'_k = \frac{\tau_k + \sum_{i=1}^n c_{ik}}{n + \sum_{l=1}^K \tau_l} = \frac{\tau_k + \sum_{i=1}^n c_{ik}}{\sum_{l=1}^K (\tau_l + \sum_{i=1}^n c_{il})} \tag{5.7}$$

Note that $\sum_{k=1}^K \sum_{i=1}^n c_{ik} = \sum_{i=1}^n \sum_{k=1}^K c_{ik} = n$.

$$m'_k = \frac{\tau_k}{\tau_k + \sum_{i=1}^n c_{ik}} \mu_k + \frac{\sum_{i=1}^n c_{ik}}{\tau_k + \sum_{i=1}^n c_{ik}} \frac{\sum_{i=1}^n c_{ik} x_i}{\sum_{i=1}^n c_{ik}}$$

$$= \eta_k \mu_k + (1 - \eta_k)\bar{m}_k \tag{5.8}$$

where

$$\eta_k = \frac{\tau_k}{\tau_k + \sum_{i=1}^n c_{ik}}$$

and

$$\bar{m}_k = \frac{\sum_{i=1}^n c_{ik} x_i}{\sum_{i=1}^n c_{ik}}$$

$\bar{m}_k$ is the ML estimate of the mean variable (see Section 2.2.3). It is apparent that the new MAP-estimated mean is an interpolation of the prior mean and the ML-estimated mean. The interpolation coefficient depends on the amount of training data $\sum_{i=1}^n c_{ik}$ for the k-th Gaussian component. If the amount of training data is very little, then η_k is close to 1, the contribution of the prior is big, and the MAP estimate relies more on the prior. As the amount of training data increases, $1 - \eta_k$ approaches 1 and the MAP solution relies more and more on the training data, and converges to the ML solution.

For the variance estimate, we can derive

$$\sigma_k^{2\prime} = \frac{2\beta_k + \sum_{i=1}^n c_{ik}(x_i - m_k')^2 + \tau_k(\mu_k - m_k')^2}{2\alpha_k - 1 + \sum_{i=1}^n c_{ik}}$$

$$= \frac{\tau_k\sigma_k^2 + \sum_{i=1}^n c_{ik}(x_i - m_k')^2 + \tau_k(m_k - m_k')^2}{\tau_k + \sum_{i=1} nc_{ik}}$$

$$= \eta_k(\sigma_k^2 + m_k^2) + (1 - \eta_k)(\bar{\sigma}_k^2 + \bar{m}_k^2) - {m_k'}^2 \tag{5.9}$$

where

$$\bar{\sigma}_k^2 = \frac{\sum_{i=1}^n c_{ik}(x_i - \bar{m}_k)^2}{\sum_{i=1}^n c_{ik}}$$

is the ML estimate of the variance.

Combining Equations 5.8 and 5.9, the MAP estimates of the output distribution can be readily interpreted as the interpolation of two sets of Gaussian mixture distributions. One is the prior distribution, with mean m_k and variance σ_k^2 for the k-th Gaussian component. The other is the ML estimate, with mean $\bar{m}_k$ and variance $\bar{\sigma}_k^2$ for the k-th Gaussian component. The interpolation coefficient is a function of the training data assigned for this Gaussian mixture distribution, which is obtained during the forward-backward training.

We can use the parameters of the monophone model for the prior distribution parameters. We will call these prior monophone models as "seed models". When there are no training data available, these would be the best guess we can have about the triphone model that we are going to estimate. As the monophone model is more robust but less accurate, and the ML estimate of triphone model is more accurate but less robust, the MAP estimate gives a good tradeoff in combining them naturally.

5.5 Experimental Results

To evaluate our new approaches proposed in Section 5.3 and Section 5.4, we compare their performance with the baseline system's performance (in Section 3.3), where the simple back-off mechanism was used to estimate the statistics of the rarely seen triphones. Table 5.2 shows how the baseline system performance changes with different back-off thresholds in estimating the unclustered rarely-seen triphone models. The final clustered triphone model has 3749 states and 12 Gaussians per state.

Table 5.2: Word error rates of the baseline systems with different back-off thresholds on the WSJ 5K task.

back-off threshold (frames)	percentage(%) of back-off states	si-dt-05 set	nov92 set
2	5.3	10.0%	7.9%
5	17.8	**9.5%**	**7.5%**
10	34.0	9.9%	7.8%

In the unclustered triphone model set for the WSJ 5K task, there are 55594 unique states in total. We can see that the best result was obtained when the back-off threshold was set to 5 frames, which means if a particular state had less than 5 frames of data, it was simply backed-off to the monophone model. Of the total states, 17.8% were backed-off to their corresponding monophone models.

Similar results were obtained for the WSJ 20K task as given in Table 5.3. There are 69187 unique HMM states in the unclustered triphone model set. When the back-off threshold was set to 5 frames, the best baseline acoustic model was obtained, which gives 11.8% word error rates on the nov92 set. 7.5% states were backed-off to the corresponding monophone model. The final clustered triphone model set has 7509 states and 12 Gaussians per state.

5.5.1 Two-Stage Decision Tree Approach

In our two-stage tree building approach, we use the statistics of unclustered initial models to build very big decision trees in the first stage. By using a relatively bigger threshold

Table 5.3: Word error rates of the baseline systems with different back-off thresholds on the WSJ 20K task

back-off threshold (frames)	percentage(%) of back-off states	nov92 set
2	2.2	12.2%
5	7.5	**11.8%**
10	15.0	12.1%

(than that used in training the unclustered model, e.g., 20 frames) of data count, we can guarantee that all rarely-seen triphones are clustered with their closest peers. The threshold for likelihood gain is set to zero to ensure that triphones that have enough training samples will not be clustered together. Then we train these loosely clustered models for several iterations and obtain their updated statistics. In the second stage, we use these models and their statistics to build the final trees. The trees in the second stage have the same number of leaf nodes as the baseline model, that is, 3749 states. Also, the number of Gaussian components per state is the same as in the baseline system. To find the best data count threshold in building decision trees in the first stage, a simple grid search is conducted and the results are listed in Table 5.4.

Table 5.4: Results of systems built by using different phonetic decision trees in the first stage on the WSJ 5K task.

threshold of data count(frames)	10	20	30	40	50
# of leaves in 1st-stage tree	40705	33060	28362	25098	22612
WER on si-dt-05 set	9.9%	9.1%	8.9%	9.0%	9.2%
WER on nov92 set	7.3%	7.2%	7.0%	7.2%	7.6%

We can see that when the threshold of data count is set to 30 frames, we obtained the best performance, 8.9% and 7.0% word error rates on the si-dt-05 set and nov92 set, respectively. It is 5% better than the performance of the baseline system.

The results of statistical significance tests are given in Table 5.5. The details of Wilcoxon test on the combined set nov92+si-dt-05 is also given in Table 5.6. According to these significance tests, the two-stage tree based system has a statistically significant difference from the baseline system, and the two-stage tree based system is better. Note that due to the relatively small number of sentences (330) and low word error rates (about 7%), the number of detected sentence segments is small and therefore it is difficult even for the matched-pairs test to find significant differences between the two systems.

Table 5.5: Significance tests between the baseline system and two-stage decision tree based system on the WSJ 5K task.

			significance	
test set	baseline	2-stage	Wilcoxon	matched-pairs
nov92	7.5%	7.0%	no	no
si-dt-05	9.5%	8.9%	no	yes
nov92 + si-dt-05	8.6%	8.1%	yes	yes

On the WSJ 20K tasks, similar results and conclusions were obtained and are given in Table 5.7. When the data count threshold is set to 30 frames in building decision trees in the first stage, the best acoustic model is obtained, and it reduces the word error rate of the baseline system by 6%. The model has the same number of clustered states and number of Gaussian components per state as the baseline model. According to significance tests, both Wilcoxon test and matched-pair test reject the Null hypothesis. Therefore, the two-stage tree based system is a significant improvement over the baseline system. Table 5.8 gives some details of the matched-pair test. Note that the number of reference words in the table is the number of reference words only in the detected errorful segments. Since it is much smaller than the number of all reference words, the error rates in the table are much higher.

5.5.2 MAP-based Approach

It can be seen from Section 5.4.1 that when the values of τ_k are known, all the other prior parameters can be directly estimated from the seed HMM parameters. In this case, τ_k can be regarded as a weight associated with the k-th Gaussian component. When this weight is large, the prior density is sharply peaked around the values of the seed HMM

Table 5.6: Wilcoxon test between the baseline system and two-stage decision tree based system on the WSJ 5K nov92+si-dt-05 set. $T_+ = 141$, $T_- = 30$, $z = 2.42$. As z is above the rejection threshold 1.96, the null hypothesis is rejected.

speaker	2-stage	baseline	difference	rank	signed-rank
050	96.09	96.23	-0.14	2.0	-2.0
051	91.20	91.90	-0.70	10.0	-10.0
052	90.59	88.92	1.68	17.0	17.0
053	90.52	90.93	-0.41	4.0	-4.0
22g	91.30	90.87	0.43	5.0	5.0
22h	91.14	89.57	1.57	16.0	16.0
420	94.44	93.66	0.77	12.0	12.0
421	93.63	92.78	0.85	13.0	13.0
422	81.36	79.49	1.86	18.0	18.0
423	91.53	91.53	0.00	1.0	1.0
440	94.93	94.47	0.46	7.0	7.0
441	87.76	87.13	0.63	9.0	9.0
442	93.77	92.52	1.25	15.0	15.0
443	94.68	94.22	0.46	6.0	6.0
444	91.49	90.95	0.54	8.0	8.0
445	93.18	94.18	-1.00	14.0	-14.0
446	95.92	95.78	0.15	3.0	3.0
447	91.93	91.17	0.76	11.0	11.0
Avg.	91.97	91.46	0.51	–	–

parameters, which will be only slightly modified by the training process. Conversely, if τ_k is small, the MAP estimate will mainly depend on the training data. To increase the robustness, the τ_k values can be constraint to be identical for all Gaussians of a given state, or for all states of an HMM, or even for all the HMMs.

In this thesis, we only investigate the case that τ_k is tied for all HMMs in the model set. We did some experiments to search for the best value of this globally tied τ, as shown in Table 5.9 and Table 5.10 for the WSJ 5K and 20K task, respectively. The baseline results are from Section 3.3. We used the MAP estimation to obtain the statistics of unclustered triphone models for decision tree building. The size of the final clustered triphone model set is about the same as the baseline system.

Table 5.7: Results of systems built by using different phonetic decision trees i n the first stage on the WSJ 20K task.

Threshold of data count(frames)	10	20	30	40	50
# of leaves in 1st-stage tree	60878	54690	50254	46738	43862
WER on nov92 test set	11.7%	11.4%	11.1%	11.2%	11.3%

Table 5.8: Matched-Pairs test between the baseline system and two-stage tree based system on the WSJ 20K nov92 set. # of detected segments: 401, # of sentences: 330, mean: 0.092, std dev: 0.984, Z statistic: 2.067 (above the rejection threshold 1.96)

Reference words	baseline	two-stage
Total of 1329	656	619
Percentage error	49.4%	46.6%

It can be seen that the MAP-based systems consistently outperform the baseline systems, and when the prior parameter τ_k is set to 2, the MAP-based systems give the best performance, which reduce the word error rates of the baseline systems by about 5% on both WSJ 5K and WSJ 20K tasks. We also compared the MAP-based system against the two-stage decision tree based system as well as the baseline system in Table 5.11 and Table 5.12. The results of the two-stage decision tree based system are from Section 5.5.1. we can see that the two-stage decision tree based systems give only slightly better results than the MAP-based systems, which are not statistically significant on these test sets.

The results of statistical significance tests are listed in Table 5.13. It shows that the MAP-based systems are significantly different from (and better than) the baseline systems.

Table 5.9: Word error rates of MAP-based systems, on the WSJ 5K nov92 set, with different τ values in MAP estimation.

states	mixtures	baseline	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 5$
1738	12	8.5%	8.3%	**8.0%**	8.2%	8.3%
2682	12	7.9%	7.6%	**7.5%**	7.5%	7.8%
3749	12	7.5%	7.2%	**7.1%**	7.2%	7.4%
4656	12	7.5%	7.3%	**7.0%**	7.1%	7.5%

Table 5.10: Word error rates of MAP-based systems, on the WSJ 20K nov92 set, with different τ values in MAP estimation.

states	mixtures	baseline	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 5$
5532	12	12.7%	12.5%	**12.1%**	12.4%	12.6%
6542	12	12.4%	11.8%	**11.5%**	11.6%	12.0%
7509	12	11.8%	11.4%	**11.2%**	11.2%	11.7%
8562	12	12.0%	11.8%	**11.5%**	11.4%	11.7%

Table 5.11: WER comparison of the MAP-based system with the baseline system and two-stage tree based system on the WSJ 5K task.

test sets	baseline	MAP($\tau = 2$)	two-stage
nov92	7.5%	7.1%	7.0%
si-dt-05	9.5%	9.0%	8.9%

Table 5.12: WER Comparison of the MAP-based system with the baseline system and two-stage tree based system on the WSJ 20K task.

test sets	baseline	MAP($\tau = 2$)	two-stage
nov92	11.8%	11.2%	11.1%

Table 5.13: Results of significance tests between baseline systems and MAP-based systems.

test sets	baseline	MAP($\tau = 2$)	significance	
			Wilcoxon	matched-pairs
5K nov92	7.5%	7.1%	no	no
5k si-dt-05	9.5%	9.0%	no	yes
5K nov92 + si-dt-05	8.6%	8.2%	yes	yes
20K nov92	11.8%	11.1%	yes	yes

Chapter 6

Decision trees with multiple-Gaussian mixture models

In this chapter, we will address the third problem that the traditional decision tree based approach does not take full advantage of the training data, particularly, the training data for the frequently-seen triphones. It is related to one assumption that the representation of every unclustered triphone state for decision tree building must be a single-Gaussian mixture distribution. We propose a new approach to relax this assumption and build phonetic decision trees using multiple-Gaussian mixture models. In Section 6.1, we will explain the problem in details. Some literature review about this problem is given in Section 6.2. Then, we present our new approach in Section 6.3. Experimental results and discussions are given in Section 6.4.

6.1 Problem: Waste of Information in Frequently-Seen Triphone Data

As mentioned before, the traditional approach uses a single-mixture model for each unclustered triphone state to build the decision trees, rather than the target multiple-mixture model. Obviously a multiple mixture distribution gives a more accurate representation of the acoustic space of a triphone state than a single mixture distribution does. If we could use multiple mixture models for the unclustered triphone states, we might be able to construct better trees and hence better clustering of triphone states, as more information in the data is exploited to represent every unique triphone state. Note that 40% of total seen triphones in the WSJ 5K training data occur more than 10 times (see Table 5.1). The amount of training data for these triphones should be able to warrant robust estimates of multiple mixture distributions for them. For example, each state should be better modeled by a mixture model with two Gaussian components, corresponding to the male and female speakers' data.

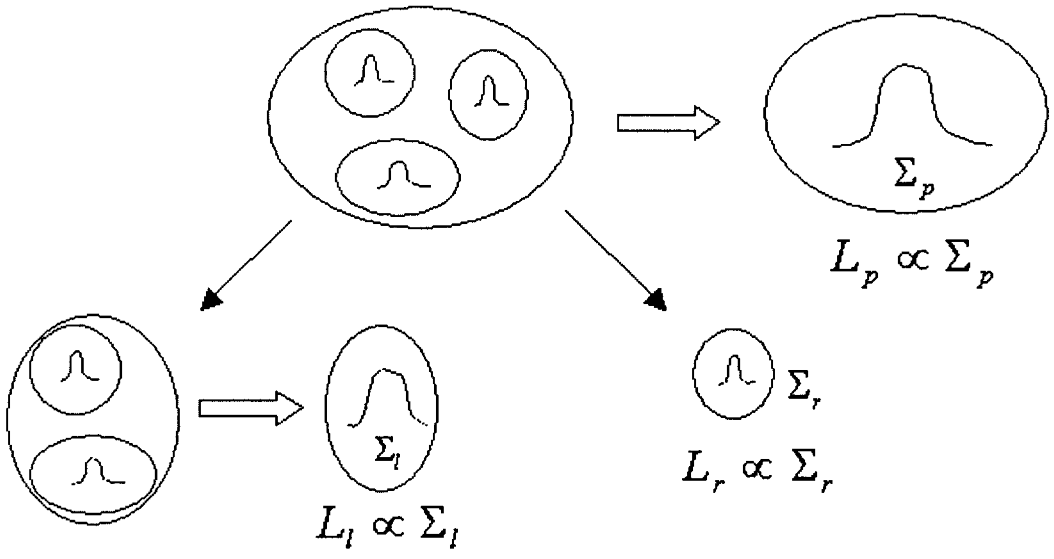


Figure 6.1: Building a decision tree using single-Gaussian state models. The statistics for each node can be directly calculated from the statistics of its member states. The likelihood for a node is a function of the node's distribution (covariance matrix).

The reason that the traditional approach uses only single mixture models to represent unclustered triphone states is mainly due to the computational complexity in the tree-building process. Theoretically, the multiple-Gaussian mixture distribution for a tree node needs to be re-estimated from the training data, whereas the single-Gaussian statistics for the node can be calculated efficiently from the sufficient statistics of the member states, which are modeled by single-mixture Gaussian distributions, without re-accessing the original training data. So at every node, for every hypothesized split, we would have to iterate through every piece of original training data to estimate the distribution of every child node, and thereby calculate the likelihood gain resulting from the split. This is

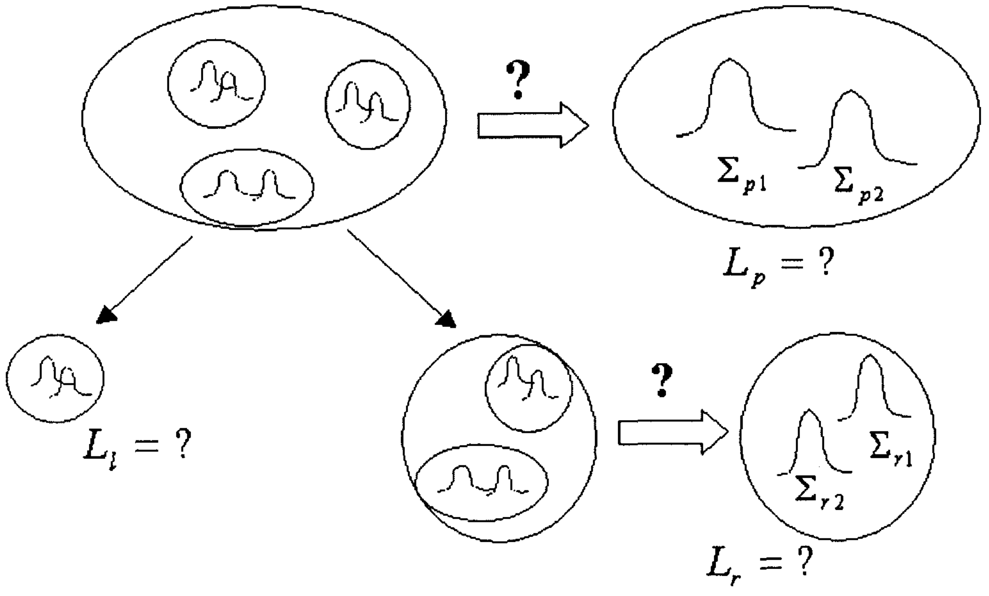


Figure 6.2: Building a decision tree using multiple-Gaussian state models. No efficient algorithm is available to estimate a multiple-Gaussian distribution for a node from its member state models. No established realtionship is available between the likelihood for a node and its multiple-Gaussian distribution.

computationally too expensive to be feasible. Figure 6.1 and Figure 6.2 illustrate the computational advantage of the traditional approach, and the difficulty in using multiple-Gaussian mixture models to build decision trees.

In Figure 6.1, the distribution (a pooled single Gaussian) of every node can be accurately calculated from its member states' Gaussian parameters and their associated state occupancy, as they (mean, variance, and state occupancy of a Gaussian distribution) consist of the sufficient statistics for every state. The likelihood is a function of the parameters of this distribution (see Section 2.3.2). However, in Figure 6.2, there is no way to directly calculate a multiple-Gaussian mixture distribution for every node from its member states' multiple-Gaussian mixture distributions, because in this case, there are no sufficient statistics for the member states.

However, the single Gaussian distribution is a very crude representation of the acoustic space of triphone states and might be inadequate to model the acoustic variation in the training data. Decision trees based on such crude models might not give good clustering of triphone states. This constraint limits the decision tree building from making effective use of data (Instead, it uses the data in a wasteful manner, throwing away a large amount of information in the training data).

6.2 Related Work

To address this problem, Chou et al. [13] incorporated a so-called "m-level optimal subtree" into the traditional tree construction to approximate a multiple-Gaussian mixture distribution of each node, although each member state still has only single-Gaussian mixture distribution as in the traditional approach. It is somewhat similar to the look-ahead search [30]. They proposed a scheme to reduce the dramatically increased computation. Modest improvement was obtained through their approach. However, as their approach still uses single-mixture Gaussian distributions for the unclustered triphone states, the training data were not efficiently used.

Nock et al. [42] estimated the multiple-Gaussian mixture distributions of unclustered triphone states by using the fixed state alignment provided by a previously trained and accurate model set. However, the method of using the multiple-Gaussian mixture models was not mentioned in their paper, and their approach failed to achieve any improvement in terms of recognition accuracy.

Kim et al. [29] proposed a goodness-of-split criterion to use multiple-Gaussian mixture models in building decision trees. However, to estimate a multiple-Gaussian mixture distribution for a node from its member states, an assumption was made that all the Gaussians in a triphone state model corresponded to the Gaussians in another triphone state model just in an arbitrary order. In practice, all first (here first just means that it is stored as the first element of the array that holds a Gaussian mixture model) Gaussians of all triphone states in a node were merged into one Gaussian, and it became the first Gaussian component for the node. And so was for the second Gaussians, and so on. Obviously this assumption was too strong. Also their test set (a subset of the WSJ 5K si-dt-05 set) was so small (60 sentences) that the improvement obtained might not be significant.

6.3 Our Approach

In our view, this problem can be directly approached if some appropriate assumptions are made. We divide this problem into two related sub-problems:

- How to estimate the likelihood for a node, given a multiple-Gaussian mixture distribution
- How to estimate a multiple-Gaussian mixture distribution of a node from its member state, each having a multiple-Gaussian mixture distribution

For the first sub-problem, we need to create a new goodness-of-split criterion as the original one in Section 2.3.2 is only applicable to the single-Gaussian mixture case. For the second problem, an efficient algorithm is necessary to approximate the distribution, without re-accessing the original training data.

We modify the third assumption in Section 2.3.2 to be that

- the total log likelihood can be approximated by a simple summation of the log-likelihood for each Gaussian component weighted by the probability of the Gaussian component's occupancy.

So,

$$L = \sum_{e=1}^{E} \sum_{t=1}^{T} \sum_{s \in S} \sum_{k} \ln(P(o_{tk}^{e}; \mu_{sk}, \Sigma_{sk}))\gamma_{sk}^{e}(t) \tag{6.1}$$

Here $\gamma_{sk}^{e}(t)$ is the probability of the Gaussian component k in state s generating the data point at time t of example e. Following the same derivation in Section 2.3.2, we can obtain

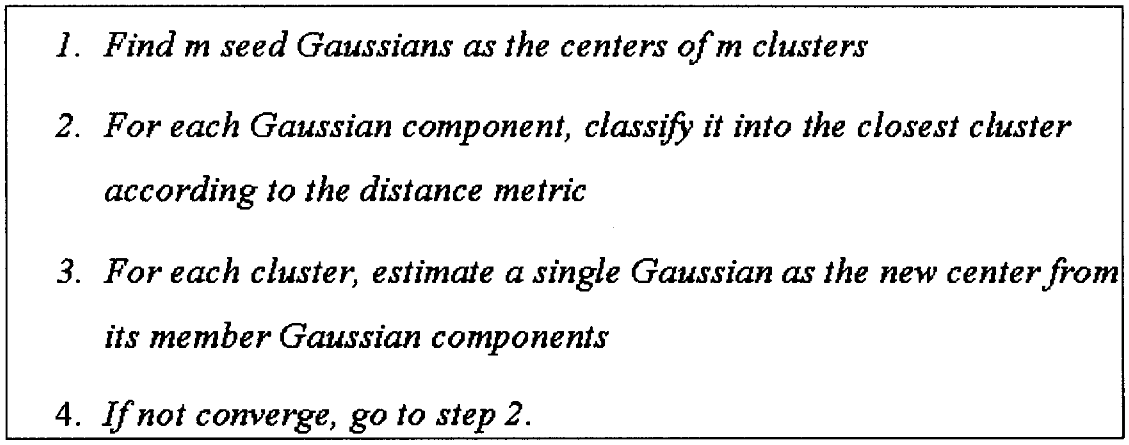
$$L = \sum_{s \in S} \sum_{k} -\frac{1}{2}(n(1 + \ln(2\pi)) + \ln(|\Sigma_{sk}|)) \sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{sk}^{e}(t) \tag{6.2}$$

Because it is a binary tree, for each split, there are two child nodes, the likelihood difference is

$$\begin{aligned}\delta L &= -\frac{1}{2}\sum_k \ln(|\Sigma_{lk}|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{lk}^{e}(t) - \frac{1}{2}\sum_k \ln(|\Sigma_{rk}|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{rk}^{e}(t)\\ &\quad +\frac{1}{2}\sum_k \ln(|\Sigma_{pk}|)\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{pk}^{e}(t)\end{aligned}$$

$\Sigma_{lk}, \Sigma_{rk}, \Sigma_{pk}$ are covariance matrices of the k-th Gaussian component in the left child node, right child node and parent node, respectively. Given the above likelihood formula, the remaining question is how to estimate the distribution (or more specifically, the covariance matrix of each Gaussian component) of a node from its member states.

Consistent with our assumption made to derive the new goodness-of-split criterion, we treat each Gaussian component in the mixture distribution of each member state as independent, that is, disassemble the mixture structure. Each Gaussian component has associated data count (occupancy). All Gaussian components in the node are pooled together, from which a new multiple-Gaussian mixture distribution is to be estimated. The K-means algorithm [46] was adapted to do this, which is given in Figure 6.3.



1. *Find m seed Gaussians as the centers of m clusters*
2. *For each Gaussian component, classify it into the closest cluster according to the distance metric*
3. *For each cluster, estimate a single Gaussian as the new center from its member Gaussian components*
4. *If not converge, go to step 2.*

Figure 6.3: An algorithm to estimate a multiple-Gaussian mixture distribution from a group of Gaussian mixture distributions

To find the seed Gaussians in the first step, we first estimate a global Gaussain from a node's all member Gaussians; then split to two using the same approach as in the mixture-splitting. Then these two new Gaussains are seed Gaussians. For more than 2 mixture case (e.g. 4), we base on the 2-mixture distribution and split each component to 2 new ones and the resulting 4 Gaussians are seed Gaussians, and so on. The distance metric is defined as the loss of likelihood due to merging two Gaussians. It is consistent with the distance metric used in the decision tree building. Figure 6.4 illustrates the idea of estimating the likelihood for a node using our approach.

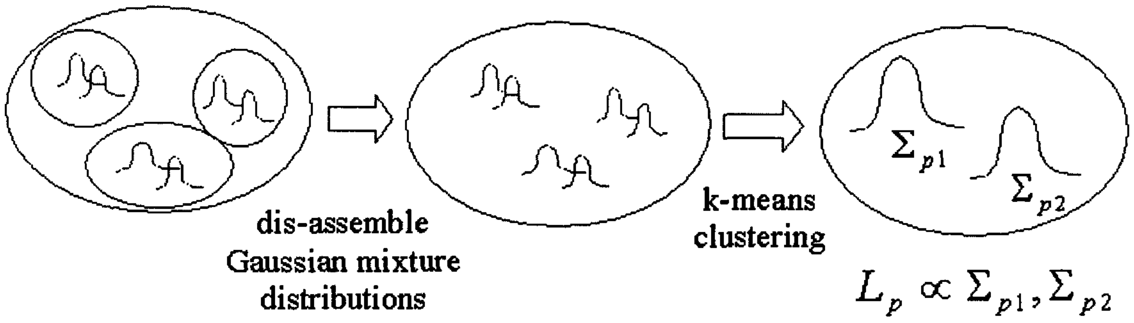

dis-assemble Gaussian mixture distributions
k-means clustering
Σ_{p1}
Σ_{p2}
$L_p \propto \Sigma_{p1}, \Sigma_{p2}$

Figure 6.4: Using the new approach to estimate the likelihood for a node whose member states have multiple-Gaussian mixture distributions.

One issue in the new approach is how to handle the triphones that do not have enough data to estimate multiple-Gaussian mixture models. We may use either our two-stage decision tree based approach or MAP-based approach. As MAP based approach is more straightforward and easy to implement, it is used in our experiments.

6.4 Experimental results

The experimental results of the new approach on the WSJ 5K tasks are given in Table 6.1. The 2-mixture system has about 3700 clustered states and 12 Gaussian components per state, that is, about the same number of total parameters as in the baseline system.

It reduced the word error rates of the baseline systems by 7%. Again, the statistical significance tests were conducted and are given in Table 6.2. It showed that the new system was significantly better than the baseline systems.

When we increased the number of Gaussian components from 2 to 4 for every unclustered triphone state model to build phonetic decision trees, however, we did not obtain any significant gain over the baseline system. This may be due to the fact that (1) many triphone states do not have enough data to sustain robust estimation of 4-mixture models, and (2) the approximation error in estimating the multiple-Gaussian distributions for tree nodes (directly from their member states, rather than using the accurate way to estimate from the original training data) becomes bigger, and therefore it offsets the gain.

Table 6.1: WSJ 5K results (word error rates) of the system built using multiple-Gaussian mixture models.

methods	nov92 set	si-dt-05 set
Baseline (1-mixture)	7.5%	9.5%
2-mixture	6.9%	8.8%

Table 6.2: Significance tests between the 2-mixture system and the baseline system on WSJ 5K test sets.

	baseline	2-mixture	significance	
test sets	WER	WER	Wilcoxon	matched-pairs
nov92	7.5%	6.9%	no	yes
si-dt-05	9.5%	8.8%	no	yes
nov92 + si-dt-05	8.6%	8.0%	yes	yes

Similar conclusions could be drawn from the results on the WSJ 20K task and are given in Table 6.3. The 2-mixture system had about the same number of total parameters as in the baseline system. The detailed analysis of Wilcoxon test on the WSJ 20K nov92 test set was given in Table 6.4. It confirmed that the new approach is superior to the traditional approach.

Table 6.3: WSJ 20K result (word error rate) of the system built using multiple-Gaussian mixture models.

methods	nov92 set	Wilcoxon test	matched-pairs test
Baseline (1 mixture)	11.8%	–	–
2-mixture	11.1%	yes	yes

Table 6.4: Significance test (Wilcoxon) on the WSJ 20K nov92 set. $T_+ = 34.0$, $T_- = 2.0$, $z = 2.24$, z is above the rejection threshold -1.96, so the null hypothesis is rejected.

speaker	2-mixture	baseline	difference	rank	signed rank
440	91.80	90.63	1.17	7.0	7.0
441	78.89	78.29	0.60	4.0	4.0
442	89.69	89.83	-0.14	2.0	-2.0
443	87.11	86.55	0.56	3.0	3.0
444	87.26	87.26	0.00	1.0	1.0
445	92.87	92.06	0.81	6.0	6.0
446	91.65	91.05	0.61	5.0	5.0
447	90.91	89.46	1.45	8.0	8.0

6.5 Put All Together

All of our approaches described in previous sections and chapters improve the traditional acoustic modeling approach by making better use of limited training data. However, each of them targets a different aspect of the traditional approach in a different way. The two-level decision tree approach neither tries to improve the unclustered triphone model for building decision trees nor builds better decision trees (i.e., chooses a better question for every split). Instead, it improves the way to use training data to estimate the *clustered* triphone model set. In contrast, all the other new approaches try to build better decision trees and therefore better clustering by estimating a better *unclustered* triphone model set. Specifically, the two-stage decision tree approach and the MAP-based approach improve the estimation of *unclustered rarely-seen* triphone models by using the limited training data for the rarely seen triphones more efficiently; the decision tree building approach using multiple-mixture models improves the accuracy of the *unclustered* triphone model set, particularly the models for those *frequently-seen* triphones, and therefore builds better decision trees. So, the advantages of these approaches may be complementary, and a system combining multiple approaches may be even better than any individual one.

In this section, we present the experimental results on combining two or more of our new approaches. The best results of all combinations of different approaches are listed in Table 6.5 for the WSJ 5K task and Table 6.6 for the WSJ 20K task. As we did not observe any significant differences between using the MAP-based smoothing approach and using the two-stage decision tree based approach, we only consider the MAP-based approach in the following experiments due to its simplicity. Note that the system ("multi-mix" in the table) that used multiple-Gaussian mixture models to build decision trees also used MAP to obtain robust estimation of unclustered triphone models before building decision trees. So it is actually a combination of two approaches. For ease to comparing, all previous results using a single approach are also included.

From the table, We can see that systems combining multiple approaches always yield better results than systems using any single approach. The best performance is given by the system that combines all of the three approaches. It shows that to some extend the contributions of different approaches are complementary to each other. The best results on the WSJ 5K nov92 and si-dt-05 set are 6.2% and 8.2%, respectively. These correspond to 17% and 14% word error rate reductions, compared with the baseline system. The final model set has about 3300 first-level leaf nodes and 21000 second-level leaf nodes. Each clustered state has 12 Gaussian components. The total number of parameters is 3.3 million, a 8% reduction over the baseline system.

For the WSJ 20K task, the best system has a 10.0% word error rate on the nov92 set, which is 15% better than the baseline system. The system has about 6500 first-level leaf nodes and 22000 second-level leaf nodes, with 12 Gaussians per state. The total number of parameters is 6.3 million, which is 11% less than the baseline system.

Table 6.5: Results (WERs) of systems combining one or more approaches on the WSJ 5K task. Note that the significance tests are conducted on the combined test set, nov92+si-dt-05 set. MP means the matched-pairs test.

approach(es)	nov92 set	si-dt-05 set	# of param. (million)	significance	
				Wilcoxon	MP
baseline	7.5%	9.5%	3.6	–	–
2-level	6.4%	8.4%	3.4	yes	yes
MAP	7.1%	9.0%	3.6	yes	yes
multi-mix (+ MAP)	6.9%	8.8%	3.6	yes	yes
MAP+2-level	6.2%	8.3%	3.3	yes	yes
MAP+2-level+multi-mix	**6.2%**	**8.2%**	3.3	yes	yes

Table 6.6: Results (WERs) of systems combining one or more approaches on the WSJ 20K task.

approach(es)	nov92 set	# of param. (million)	significance	
			Wilcoxon	matched-pairs
baseline	11.8%	7.1	–	–
two-level	10.4%	6.4	yes	yes
MAP	11.2%	7.1	yes	yes
multi-mix (+ MAP)	11.1%	7.1	yes	yes
MAP + 2-level	10.2%	6.3	yes	yes
MAP + 2-level + multi-mix	**10.0%**	6.3	yes	yes

Chapter 7

Conclusions and Future Work

In this thesis, we propose to address several problems in the traditional approach of phonetic decision-tree clustering for acoustic modeling. These problems are due to the fact that the traditional approach does not make use of the limited training data in an efficient and effective way. We proposed a number of ways to address these problems. Specifically, our contributions include:

- A two level phonetic decision tree approach. It uses a new structure of two level nodes to enable different level of sharing among the Gaussian components and mixture weights. By decoupling Gaussian and mixture weights, it can
  - make use of the limited training data more efficiently;
  - generate acoustic models with higher accuracy and better robustness;
  - provide more flexibility to control the total number of parameters.
- A two-stage decision-tree building process, which involves two stages to build different decision trees. By using decision trees in the first stage to make an initial clustering of rarely-seen triphones, it can obtain more robust statistics for them, and therefore better decision trees can be constructed in the second stage.
- Use of MAP to better estimate the unclustered triphone models for decision tree clustering. Like the two-stage decision tree approach, it explicitly addresses the problem associated with parameter estimation of rarely-seen triphones due to data sparsity. It uses the well-known MAP technique to smooth the unclustered triphone models

with the more robust context-independent models (monophone models). Compared to the traditional approach, the information contained in the limited training data for those rarely-seen triphones are better exploited. Therefore, the decision trees are better constructed, and better system performance are obtained.

- An approach to building phonetic decision trees using multiple-Gaussian mixture models for the unclustered triphones. By making an appropriate assumption, we proposed a new goodness-of-split criterion to directly build decision trees using multiple-Gaussian mixture models. An efficient algorithm is used to approximate a multiple Gaussian mixture distribution for a tree node from its member states, each having a multiple Gaussian mixture distribution. As the unclustered triphones are more accurately represented by multiple-Gaussian mixture models and the information in the training data is efficiently used, better phonetic decision trees and therefore more effective acoustic models are constructed.

Through our extensive study in this thesis, we demonstrate that by more **efficient** use of data in phonetic decision tree based acoustic modeling, more **effective** acoustic models can be obtained, and therefore resulting in a better system performance.

We should mention that although our approaches are proposed and evaluated in the context of phonetic decision tree based clustering, they are not limited to it. It is straightforward to apply them to other clustering approaches, such as the bottom-up data-driven approach.

Our future research directions will be:

- Discriminative training. As it is well known that the Maximum Likelihood (ML) criterion is not an optimal criterion for speech recognition, whose ultimate goal is to discriminate different speech patterns. Some alternative criteria based on discrimination have been proposed, such as Minimum Classification Error (MCE) [28] and Maximum Mutual Information (MMI) [43]. Usually the training based on these new criteria has more computation costs than that of ML-based training. Recently, the MMIE algorithm has been successfully applied to large vocabulary tasks [51]. It is used to estimate the model parameters, replacing the conventional ML-based

Baum-Welch algorithm. However, triphone states are still clustered by the traditional phonetic decision trees, which are based on the ML criterion. No discriminative criterion consistent with the MMIE or MCE criterion has been proposed for decision tree based clustering. It would be an interesting research topic to incorporate the discriminative principles into phonetic decision tree building. The optimal tree structure might be the one that maximizes the discrimination (distances) among different tree leaf nodes (states) (i.e., the inter-state distances), and also minimize the variation of each leaf node (i.e., the intra-state distances), rather than the one that maximizes the likelihood of the training data given the clustering.

- Use of prosodic information in acoustic modeling. For a long time, prosodic information has been recognized critical for human speech perception. However, it is not used in most of today's English LVCSR systems. Although much research has been conducted to incorporate a variety of information in the training data into the phonetic decision tree building process, there is no systematic approach being proposed to incorporate prosodic information (such as stress, pitch, etc.) into decision tree based acoustic modeling. We believe this is an interesting research topic that needs to be explored.

Bibliography

[1] ANASTASAKOS, T., MCDONOUGH, J., AND MAKHOUL, J. Speaker adaptive training: A maximum likelihood approach to speaker normalization. In *Proceedings of the 1997 International Conference on Acoustics, Speech, and Signal Processing* (1997), vol. 2, pp. 1043–1046.

[2] BAHL, L. R., BROWN, P. F., DE SOUZA, P. V., AND MERCER, R. L. A tree-based language model for natural language speech recognition. *IEEE Transactions on Acoustic, Speech Signal Processing 37* (1989), 1001–1008.

[3] BAHL, L. R., DE SOUZA, P. V., GOPALAKRISHNAN, P. S., NAHAMOO, D., AND PICHENY, M. A. Decision-trees for phonological rules in continuous speech. In *Proceedings of the 1991 International Conference on Acoustics, Speech, and Signal Processing* (1991), pp. 185–188.

[4] BAHL, L. R., JELINEK, F., AND MERCER, R. L. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 5* (1983), 179–190.

[5] BAKER, J. K. The dragon system – an overview. *IEEE Transactions on Acoustics, Speech and Signal Processing, 23(1)* (1975), 24–29.

[6] BAUM, L. An inequality and associated maximization techniques in statistical estimation for probabilistic functions of Markov processes. In *Inequalities: Proceedings, academic press* (1972), vol. 3, pp. 1–8.

[7] BAUM, L. E., AND PETRIE, T. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics 37* (1966), 1554–1563.

[8] BELLAGARDA, J., AND NAHAMOO, D. Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustic, Speech and Signal Processing 38* (1990), 2033–2045.

[9] BEULEN, K., AND NEY, H. Automatic question generation for decision-tree based state tying. In *Proceedings of the 1998 International Conference on Acoustics, Speech, and Signal Processing* (1998), vol. 2, pp. 805–808.

[10] Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees.* Wadsworth & Brooks, 1984.

[11] Chen, S., and Gopalakrishnan, P. S. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proceedings of the Broadcast News Transcription and Understanding Workshop* (1998), pp. 127–132.

[12] Chou, P. Optimal partitioning for classfication and regression trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13* (1991), 340–354.

[13] Chou, W., and Reichl, W. High resolution decision tree based acoustic modeling beyond CART. In *Proceedings of Fifth International Conference on Spoken Language Processing* (1998), pp. 2203–2206.

[14] Davis, S., and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 28* (1980), 357–366.

[15] Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistics Society, Series B, 39* (1977), 1–38.

[16] Digalakis, V., monaco, P., and Murveit, H. Genones: Generalized mixture tying in continuous speech hmm-based speech recognizers. *IEEE Transactions on Speech and Audio Processing 4* (1996), 283–289.

[17] Gauvain, J.-L., and Lee, C.-H. Bayesian learning of gaussian mixture densities for hidden markov models. In *Proceedings of DARPA Speech and Natural Language Workshop* (1991), pp. 185–190.

[18] Gauvain, J.-L., and Lee, C.-H. Map estimation of continuous density hmm: Theory and applications. In *Proceedings of DARPA Speech and Natural Language Workshop* (1992), pp. 272–277.

[19] Gauvain, J.-L., and Lee, C.-H. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech Audio Processing, 2* (1994), 291–298.

[20] Gillick, L., and Cox, S. J. Some statistical issues iin the comparison of speech recognition algorithms. In *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing* (1989), pp. 532–535.

[21] Gillick, L., and Cox, S. J. Tools for the analysis of benchmark speech recognition tests. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing* (1990), vol. 1, pp. 97–100.

[22] Hermansky, H. Perceptual linear predictive (plp) analysis of speech. *Journal of the Acoustical Society of America, 87* (1990), 1738–1752.

[23] Huang, X., and Jack, M. Semi-continuous hidden markov models for speech signals. *Computer Speech and Language 3* (1989), 239–252.

[24] Hwang, M.-Y. *Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition.* PhD thesis, Carnege-Mellon University, December 1993.

[25] Jelinek, F. Continuous speech recognition by statistical methods. *IEEE Proceedings 64* (1976), 532–536.

[26] Jelinek, F., and Mercer, R. L. Interpolated estimation of markov source parameters from sparse data. *in Pattern Recognition in Practice, edited by E. S. Gelesma and L. N. Kanal, Amsterdam, The Netherlands: North-Holland* (1990), 381–397.

[27] Juang, B., Levinson, S., and Sondhi, M. Maximum likelihood estimation for multivariate observations of Markov chains. *IEEE Transactions on Information Theory, 32(2)* (1986), 307–309.

[28] Juang, B.-H., and Katagiri, S. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing 40* (1992), 3043–3054.

[29] Kim, D., Liu, C., Wu, X., and Yan, Y. High accuracy acoustic modeling based on multi-stage decision tree. In *Proceedings of Sixth European Conference on Speech Communication and Technology* (1999), vol. 3, pp. 1335–1338.

[30] Kuhn, R., Lazaridesi, A., Normandin, Y., and Brousseau, J. Improved decision-trees for phonetic modeling. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing* (1995), pp. 552–555.

[31] Lazarides, A., Normandin, Y., and Kuhn, R. Improving decision-trees for acoustic modeling. In *Proceedings of Fourth International Conference on Spoken Language Processing* (1996), pp. 1053–1056.

[32] Lee, C.-H., and Gauvain, J.-L. Speaker adaptation based on map estimation of hmm parameters. In *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing* (1993), vol. 2, pp. 558–561.

[33] Lee, C. Z., and O'Shaughnessy, D. Clustering beyond phoneme contexts for speech recognition. In *Proceedings of Fifth European Conference on Speech Communication and Technology* (1997), vol. 1, pp. 19–22.

[34] Lee, K.-F. Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition. *in Readings in Speech Recognition, Edited by A. Waibel and K-F. Lee, Morgan Kaufmann Publishers, Inc.* (1990), pp. 347–366.

[35] Lee, L., and Rose, R. Speaker normalization using efficient frequency warping procedures. In *Proceedings of the 1996 International Conference on Acoustics, Speech, and Signal Processing* (1996), vol. 1, pp. 353–356.

[36] Leggetter, C. J., and Woodland, P. C. Maximum likelihood linear regression for speaker adaptation of HMMs. *Computer Speech and Language, 9* (1995), 171–186.

[37] Liu, C., Wu, X., , and Yan, Y. High accuracy acoustic modeling using two-level decision-tree based state tying. In *Proceedings of Sixth European Conference on Speech Communication and Technology* (1999), vol. 4, pp. 1703–1706.

[38] Liu, C., and Yan, Y. Improving acoustic modeling by more efficient use of data in decision tree based clustering. In *Proceedings of the 2001 International Conference on Artificial Intelligence* (2001), vol. 2, pp. 568–573.

[39] M-Y, H., and Huang, X. Shared distribution hidden markov models for speech recognition. *IEEE Transactions on Speech and Audio Processing 1* (1993), 414–420.

[40] Markel, J. D., and Gray, A. H. *Linear Prediction of Speech.* Springer-Verlag, 1976.

[41] Mason, R. L., Gunst, R. F., and Hess, J. L. *Statistical Design and Analysis of Experiments.* John Wiley & Sons Inc, 1989.

[42] Nock, H. J., Gales, M. J. F., and Young, S. J. A comparative study of methods for phonetic decision-tree state clustering. In *Proceedings of Fifth European Conference on Speech Communication and Technology* (1997), vol. 1, pp. 111–114.

[43] Normandin, Y. Maximum mutual information estimation of hidden markov models. *in Automatic Speech and Speaker Recognition, Edited by C.-H. Lee and K.K. Paliwal and F.K. Soong, Kluwer Academic Publishers* (1996), 57–81.

[44] Odell, J. J. *The Use of Context in Large Vocabulary Speech Recognition.* PhD thesis, Queen's College, University of Cambridge, March 1995.

[45] POLYMENAKOS, L., OLSEN, P., KANVESKY, D., GOPINATH, R., GOPALAKRISHNAN, P., AND CHEN, S. Transcription of broadcast news - some recent improvement to ibm's lvcsr system. In *Proceedings of the 1998 International Conference on Acoustics, Speech, and Signal Processing* (1998), vol. 2, pp. 901–904.

[46] RABINER, L., AND JUANG, B.-H. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[47] REICHL, W., AND CHOU, W. A fast segmental clustering approach to decision tree tying based acoustic modeling. In *Proceedings of IEEE workshop on Automatic Speech Recognition and Understanding* (1997), pp. 185–191.

[48] REICHL, W., AND CHOU, W. A unified approach of incorporating general features in decision tree based acoustic modeling. In *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing* (1999), vol. 2, pp. 573–576.

[49] SCHWARTZ, R. M., CHOW, Y. L., ROUCOS, S., KRASNER, M., AND MAKHOUL, J. Improved hidden markov modeling of phonemes for continuous speech recognition. In *Proceedings of the 1984 International Conference on Acoustics, Speech, and Signal Processing* (1984), pp. 35.6.1–4.

[50] SINGH, R., RAJ, B., AND STERN, R. Automatic clustering and generation of contextual questions for tied states in hidden markov models. In *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing* (1999), pp. 117–120.

[51] VALTCHEV, V., ODELL, J. J., WOODLAND, P. C., AND YOUNG, S. J. MMIE training of large vocabulary recognition systems. *Speech Communication 22* (1997), 303–314.

[52] VITERBI, A. J. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory, 13* (1967), 260–269.

[53] WOLFE, D. A. *Non-parametric Statistical Methods.* John Wiley & Sons Inc, 1973.

[54] WOODLAND, P. C., ODELL, J. J., VALTCHEV, V., AND YOUNG, S. J. Large vocabulary continuous speech recognition using HTK. In *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing* (1994), vol. 2, pp. 125–128.

[55] WU, X., LIU, C., YAN, Y., KIM, D., CAMERON, S., AND PARR, R. The 1998 OGI-FONIX broadcast news transcription system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop* (1999). n.p.

[56] YAN, Y., LIU, C., AND ZHENG, C. A multiple feature front-end approach to speech in noise. In *(to appear) Proceedings of the Fourth IASTED International Conference on Signal and Image Processing* (2002).

[57] YAN, Y., WU, X., SCHALKWYK, J., AND COLE, R. Development of cslu lvcsr: the 1997 darpa hub4 evaluation system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop* (1998). n.p.

[58] YOUNG, S., KERSHAW, D., ODELL, J., OLLASON, D., VALTCHEV, V., AND WOODLAND, P. *The HTK Book.* Entropic Ltd., January 1999.

[59] YOUNG, S. J. Large vocabulary continuous speech recognition : A review. *IEEE Signal Processing Magazine* (1996), 1–23.

[60] YOUNG, S. J., ODELL, J. J., AND WOODLAND, P. C. Tree based state tying for high accuracy acoustic modeling. In *Proceedings of ARPA Workshop on Human Language Technology* (1994), pp. 286–291.

[61] YOUNG, S. J., AND WOODLAND, P. C. State clustering in hmm-based continuous speech recognition. *Computer Speech and Language 8* (1994), 369–384.

Biographical Note

Chaojun Liu was born in Hunan, China on September 18, 1971. He attended University of Science and Technology of China (USTC) from 1989 to 1994, and obtained his Bachelor Degree in Electrical Engineering in 1994. He then joined Institute of Acoustics, Chinese Academy of Science to start his graduate study in speech recognition. He obtained his Master Degree in 1997.

From 1997 to 2002, Chaojun Liu pursued his Ph.D. degree in Computer Science and Engineering Department at OGI School of Science & Engineering, Oregon Health & Science University (formerly Oregon Graduate Institute of Science and Technology). His major was speech recognition.

Starting from early 1998, Chaojun Liu was actively involved in the development of the OGI large vocabulary continuous speech recognition system. As part of the team, Chaojun Liu participated in the competitions of the DARPA Broadcast News benchmark tests in 1998 and the NIST Speech in Noise Environments test (SPINE) in 2001. His major contributions lie on building and improving acoustic models of the system.

His research interest includes: applications of speech recognition techniques, researches on speech recognition algorithms, or other related spoken language applications and researches.

List of Publications:

- LIU, C. AND YAN, Y., Robust state clustering using phonetic decision trees. Submitted to *Speech Communication Journal* (2002).

- YAN, Y., LIU, C. AND ZHENG, C., A multiple feature front-end approach to speech in noise. In *Proceedings of the Fourth IASTED International Conference on Singal and Image Processing* (2002).

- Liu, C. and Yan, Y., Improving acoustic modeling by more efficient use of data in decision tree based clustering. In *Proceedings of the 2001 International Conference on Artificial Intelligence* (2001) vol. 2, pp. 568-573.

- Liu, C. and Yan, Y., Speaker change detection using minimum message length criterion. In *ICSLP'00* (2000).

- Liu, C., Wu, X., and Yan, Y., High accuracy acoustic modeling using two-level decision-tree based state-tying. In *EUROSPEECH'99* (1999), vol. 2, pp. 1703-1706.

- Kim, D., Liu, C., Wu, X. and Yan, Y., High accuracy acoustic modeling based on multi-stage decision tree. In *EUROSPEECH'99* (1999).

- Wu, X., Liu. C., Yan, Y., Kim, D., Cameron, S. and Parr R., The 1998 OGI-FONIX broadcast news transcription system. In *Proceedings, Broadcast News Transcription and Understanding Workshop* (1999).