Run-time Information Fusion in Large Vocabulary Continuous Speech Recognition

Chengyi Zheng
B.E., Shanghai JiaoTong University, Shanghai, China, 1995
M.S., Fudan University, Shanghai, China, 1998

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

April 2004

The dissertation "Run-time Information Fusion in Large Vocabulary Continuous Speech Recognition" by Chengyi Zheng has been examined and approved by the following Examination Committee:

Dr. Yonghong Yan
Associate Professor
Thesis Research Advisor

Dr. Jan P. H. van Santen
Professor

Dr. Peter A. Heeman
Assistant Professor

Dr. John-Paul Hosom
Assistant Professor

Dr. Hynek Hermansky
Professor, IDIAP, Switzerland

Dedication

To my parents.

Acknowledgements

I would like to thank Dr. Yonghong Yan, my advisor, for his many suggestions and constant support during this research. Without his supervision and guidance, I would not have been able to go through the chaos and confusion that has happened in the past five years. Special thanks also goes to the other members of the large vocabulary continuous speech recognition group: Dr. Xintian Wu, and Dr. Chaojun Liu. As the youngest of this group, I received tremendous amount of help from them. Our brotherhood will become my precious memory.

I also would like to thank Dr. Peter Heeman who has been my on-campus advisor in the early years of my Ph.D study. He provided many useful references and friendly encouragement.

Thanks are also due to my thesis committee members, Dr. Yonghong Yan, Dr. Jan P. H. van Santen, Dr. Peter Heeman, Dr. Hynek Hermansky, Dr. John-Paul Hosom, and Dr. James A. Larson, for reviewing my thesis and giving me invaluable feedback to improve my thesis.

Thanks goes to Dr. Hynek Hermansky and his students: Dr. Sachin Kajarekar, Dr. Pratibha Jain, Andre Adami, Sunil Sivadas, and others for their help during our collaboration on two national competitions.

I had the pleasure of meeting the folks in CSLU. They are wonderful people and their support makes research like this possible. Special thanks goes to Ed Kaiser, for spending his busy time on proof-reading my papers.

Of course, I am grateful to my parents for their patience and *love*. Without them this work would never have come into existence (literally).

Contents

List of Tables

List of Figures

Abstract

Run-time Information Fusion in Large Vocabulary Continuous Speech Recognition

Chengyi Zheng

Ph.D., OGI School of Science & Engineering
at Oregon Health & Science University
April 2004

Thesis Advisor: Dr. Yonghong Yan

Continuous speech recognition systems are environmentally sensitive and suffer from the great variability of speech. In order to achieve recognition robustness, there's a strong interest among researchers on how to fuse different information sources for speech recognition. A common problem of those approaches is that complementary information is lost either before or after recognition.

To avoid this unrecoverable information loss, and to better utilize this complementary information, we proposed a run time information fusion scheme. The hypothesis of this thesis is that by performing fusion at different levels and stages of a Large Vocabulary Continuous Speech Recognition (LVCSR) system, especially inside the decoder, more reliable and efficient fusion is possible.

The hypothesis is first tested in a speech segmentation task, which is essential to the performance of an LVCSR system. Furthermore, three different approaches of run

time fusion are proposed and implemented inside an LVCSR decoder. The experiments demonstrate the effectiveness and potential of these approaches.

Chapter 1

Introduction

The goal of this thesis work is to attack the major barriers that prevent existing speech technology from being used in real world noisy operating environment. Towards this end, this thesis will address an innovative work in performing run-time fusion of multiple information sources. Whereas the ASR field has evolved from speaker-dependent to speaker-independent systems, we believe that this direction of research is essential to achieve the next generation of "environment-independent" speech recognition.

Fletcher and his colleagues at Bell Labs extensively studied how humans process and recognize speech [1]. This work showed that the phones are processed in independent articulation bands and that these independent estimates are "optimally" merged to achieve the recognition results. Recent research activities on multistream or multiband also demonstrated the importance of looking at the data from different angles (different signal processing and features) and fusion of the information to obtain improved recognition accuracy (a greater than 20% error reduction was found in the SPINE1 task by doing so). However, both Fletcher and the recent activities did not explicitly conclude how different information should be fused to form the sound-unit recognition in order to achieve human-like performance.

In this thesis work, we investigated fusion strategies during the decoding stage (run time) so that critical information will be utilized earlier to avoid pruning errors that may not be recoverable in a later processing stage. Mathematically, this thesis work is different from previous work, which assumes either time-synchrony (concatenating the features to

form a single feature stream, such as appending energy to MFCC) or complete independence (running separate recognitions and combining the lattices).

From the theoretic side, how humans use complex components for speech recognition are not, as yet, deciphered. Research on using multiple feature streams in one system has a relative short history. From an engineering side, conducting information fusion at the decoding stage requires knowledge of traditional LVCSR decoders and a deep understanding of speech recognition at the system level.

This thesis will first lay down the theoretic framework of the run time fusion approach, then describes the detailed design and implementation from the engineering point of view. The goal of this thesis work is to bring multi-information systems to a new level of excellence and change the way in which complementary information extracted from different features is utilized.

1.1 Speech Recognition System Structure

Most state-of-the-art speech recognition systems (Figure 1.1) use a two pass decoding strategy in which the first pass (tree recognizer) produces a graph of the most likely word sequences, and the second pass (graph recognizer) searches this graph for the single best hypothesis. Each node (word or phone) in the graph has an acoustic likelihood estimated by matching HMM acoustic models against the input signal, and a language score calculated from an n-gram language model. The term *decoding* is often used to refer the process of recognizing the spoken words from the acoustic signal. Correspondingly, the recognizer is often called a *decoder* in reference to the information-theoretic model of speech production and recognition.

A speech recognition system can be divided into the following parts:

1. Pre-processing: speech segmentation and feature extraction

   In this step, the input speech stream is first transformed into a sequence of audio segments, which we call spoken utterances (sentences). The speech waveform

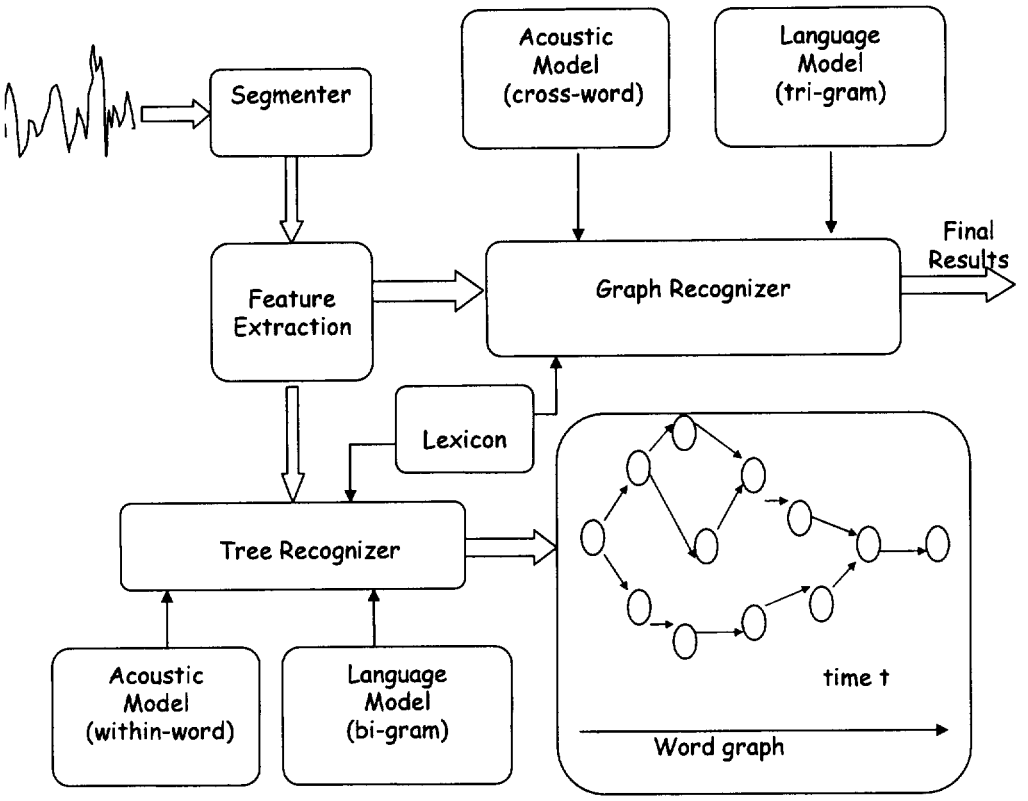


Figure 1.1: Structure of a speech recognition system

Speech sounds are first converted to a sequence of features and then the recognition is performed to find the hypotheses that best match the speech.

of each utterance is converted by a front-end signal processor into a sequence of acoustic vectors, $O = o_1, o_2, \ldots, o_t$. Each of these vectors is extracted from the short-time speech spectrum covering a period of typically 10 msec, which is called a frame. The extracted speech vectors are also called *speech features*, which are used as observations in the mathematical modeling of acoustics. Ideally, the speech features should contain as much information as possible about the linguistic content of the speech while being reasonably compact and free of redundant details.

The feature extraction processing can be further divided into:

(a) Speech Segmentation

Segmentation is the task of chopping long utterances into short ones and removing non-speech events.

(b) Channel and Gender Detection

For some systems, the pre-processing step may also include channel and gender detection. Experiments show that system performance degrades rapidly when the acoustic model is mismatched to the actual input speech. Multiple acoustic models are trained for each channel or gender condition in these systems. For each test utterance, the channel and gender detection step identifies an appropriate acoustic model to use in the later steps.

(c) Speaker Detection

Similar to channel and gender detection, a speaker adapted model performs better than a non-adapted model. Speaker detection is used to identify the speaker of the test utterances, thus the speaker adapted model can be used in recognition.

2. Speech recognition: search/decoding

In the recognition step, the speech recognizer performs a massive search to find the most likely word sequence that matches the acoustic observations. Suppose the input utterance consists of a sequence of words $W = w_1, w_2, \ldots, w_n$. The speech recognition system will determine the most likely word sequence $\hat{W}$ given the observed acoustic signal O. The search uses several knowledge sources:

(a) Acoustic Model (AM): Contains the probability of observing the vector sequence O given some specified word sequence W.

(b) Language Model (LM): Represents the a priori probability of observing W independent of the observed speech signal. In other words, it models the probabilities of word sequences.

(c) Pronunciation Lexicon: Is a list of vocabulary words in the system and their pronunciation rules.

3. Post-processing: hypotheses generation

This post-processing step generates the best hypothesis and outputs it in a certain format. Some optional procedures may be contained in the post-processing step:

(a) N-Best hypotheses generation.

The top N scored hypotheses are selected and outputs it in certain formats. Score is usually attached for each hypothesis. The hypothesis could be listed as a word sequence or a phoneme sequence. The N-Best hypotheses are usually in a text format.

(b) Word graph or word lattice generation.

Similar as N-Best hypotheses. Word graph contains the top scored hypotheses. However, word graph is not constrained by the 'N' as the N-Best approach. All the paths which reach the end of search could be included into the word graph. Thus word graph is constrained by the search pruning thresholds. Another difference between word graph and N-Best hypotheses is that the word graph is organized in the form of a graph whose nodes represent the hypothesized words. Word graph provides a more compact representation compared to N-Best hypotheses. Word graph is usually stored in some special binary format thus lacks readability.

(c) *Hypotheses re-scoring.*

The results obtained from the two procedures above can be re-scored by more accurate knowledge sources such as a high-order language model. Re-scoring can also be performed by running a second recognition pass with a more detailed acoustic model and language model.

1.2 Mathematical Basics of Speech Recognition

The task of a speech recognition system is to find a word string $W = w_1, w_2, ...w_L$ that maximizes the posterior probability of the string W given the speech observations $O = o_1, o_2, ...o_T$,[1] that is:

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

According to *Bayes'* rule:

$$P(W|O) = \frac{P(W)P(O|W)}{P(O)} \tag{1.2}$$

where $P(O)$ is the distribution of the speech observation. Since $P(O)$ is constant over the time period of interest, it can be omitted from the equation above.

Combining these two equations together, we get:

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

The recognition procedure is a massive search over all the possible word sequences to find a word sequence $\hat{W}$ that maximizes $P(W)P(O|W)$. $P(W)$ represents the "a priori" probability of observing W independently of the observed acoustic event, and acts as a grammar constraint on the word sequence. $P(W)$ is extracted from a language model during recognition, which is trained from a text database. $P(O|W)$ represents the probability of observing the vector sequence O given a specified word sequence W and measures how well the observed speech sound matches the word sequence. $P(O|W)$ is determined by an acoustic model during recognition and it is also trained beforehand on some speech databases.

1.3 Acoustical Modeling with HMMs

As stated in Section 1.2, the acoustic model is used to calculate the likelihood of generating any acoustic vector sequence O given a word sequence W. The acoustic model is

[1] This process is often called *decoding*.

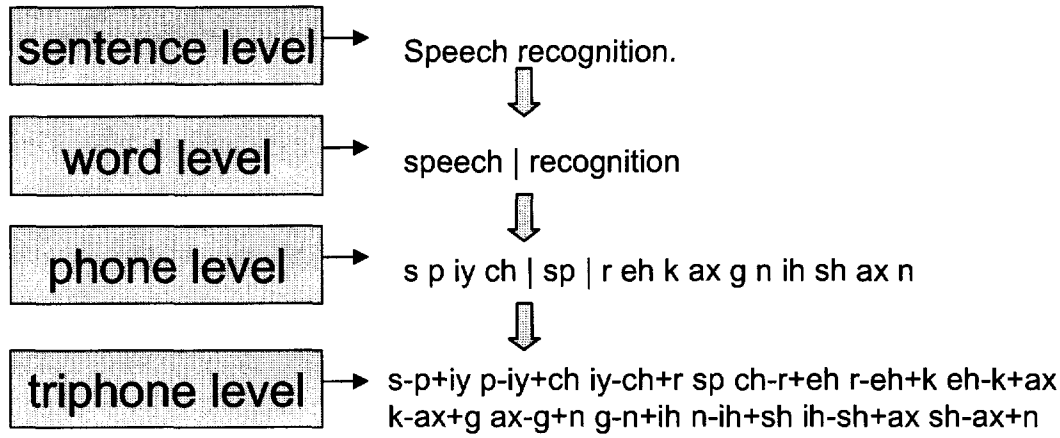


Figure 1.2: Different levels of modeling of a sample sentence

The modeling unit is context triphones. They are obtained by first expanding each word in a sentence into its pronunciations and then constructing triphones according to the left and right context of each phoneme. The phoneme /sp/ represents an *optional* short pause between two words. Usually /sp/ is skipped when considering contexts.

obtained by a statistical based training procedure that requires many speech samples of each word w for a reliable estimation. In large vocabulary systems, words are no longer the appropriate acoustic modeling unit because it is impractical to collect sufficient samples for each word. A unit smaller than a word, such as phone, is the most popular modeling unit in this case. To model the co-articulation effect of speech, context dependent phones are used instead. The simplest and most often used context dependent phone is the *triphone*, which is a single phone that takes into account its left and right neighboring phones. For distinction with triphone, the context independent phone is called a *monophone*. As shown in Figure 1.2, a word is represented first by a series of monophones according to its pronunciations and then by a series of context triphones. The construction of a triphone is to combine each phoneme with its left and right context. For example, "a-b+c" represents the phone "b" with a left neighboring phone "a" and a right neighboring phone "c". Each context triphone is modeled by a certain model. In Figure 1.2, there are two instances of phone "ih" with different neighboring phones, which are represented by different triphones and therefore different models. In the latter chapters,

the acoustic model is denoted as λ, following standard conventions.

1.4 Fusion in Speech Recognition

With the advance of computer hardware and speech technologies, automatic speech recognition (ASR) systems have been deployed in commercial applications in the last decade. However there is still a long way to go before an ASR system can reach performance comparable to humans [53].

One major challenge that current statistically based ASR technology is facing comes from the great diversity and variability of speech sources and transmission channels. It is still very difficult for an ASR system to deal with these variations. Often, a speaker change, or even an emotional change in the same speaker, causes recognition performance to degrade. Speech variations come from many sources: background noise, channel condition, music, speaker emotion, dialects, disfluency, murmur, etc.

Adverse acoustic environments, such as noise, music and background speech can also dramatically degrade the performance of current ASR systems.

Current statistically based ASR systems try to ease the problem by performing training on large amounts of collected speech data. However it's impossible to model all the diversity and variability with limited data. Even small mismatches between training and testing data can cause sharp performance degradation.

For many years, research has been devoted to finding a "perfect" feature representation of the speech signal. It seems an endless journey so far, just as we cannot make a "perfect" recognizer that makes no mistakes. Current speech recognizers adopt a single "best" feature set according to the task they are facing and measure the result on a development data set. Also feature representations have fixed parameters (such as the window and frame size, and the number and shape of the band filters) during feature extraction.

However, different features, or features with different parameters, can represent the same speech input differently. Information loss is inherent for any feature extraction

method [11]. The remaining information is different for features that are based on different feature extraction methods. These differences represent different subsets of information contained in the original speech signal. It is plausible that these differences will cause different recognition results.

Human auditory studies also support that there are multiple forms of signal processing occurring in the auditory system [24]. Evidence was found in the mammalian auditory system that each auditory nerve fiber splits and transmits the same data through seven different types of nerve cells. Each type of cell produces a different response and their outputs are combined at higher level processing [1, 29]. Additional research shows that humans can recognize speech with limited spectral cues and can easily integrate acoustic cues from different frequency regions for speech perception. When the environment becomes noisier, humans rely on more cues from the speech signal.

For these reasons (from both theoretical and engineering viewpoints), there's a strong interest among ASR researchers on how to combine different features for speech recognition. The success of this research is partly due to the efficiency of improving recognition accuracy, partly due to the simplicity and ease of deployment. The existing art can be roughly classified into two categories: pre-recognition and post-recognition combination. Complementary information from multiple features is adopted either before or after recognition.

To utilize the complementary information from multiple features, pre-recognition approaches, such as the Multi-Stream approach [11, 42, 61], were proposed in recent years. Feature or probability combination is performed before the actual recognition engine started.

Similar complementary characteristics were observed when using different acoustic or language models; some models perform well for certain speakers or environments but degrade under other circumstances. Two systems with identical performance can have a huge difference in the errors they make. To utilize the differences in the multiple recognizers, people started to combine the outputs of several recognizers in a post-recognition

combination scheme. Approaches in this scheme include ROVER [28], hypothesis combination [82], etc. All of them have demonstrated their ability to improve recognition performance.

1.5 This Thesis

The fusion approaches presented above have proven to be effective in improving system performance. However some complementary information is either lost after the recognition (such as in post-recognition) or not fully used (such as in pre-recognition). To avoid these problems, we proposed a run time fusion scheme. The main idea of our approach is to conduct information fusion during the run time of a recognizer (or decoder). [2]

The first proposed approach is to use fusion to segment the speech into utterance at the run time of the recognizer ("Segmenter" in Figure 1.1). Multiple filter bank coefficients are fused to make the speech/non-speech detection. The segmentation approach is evaluated by performing the recognition on the segmented files and measuring the overall word error rates.

The other proposed approaches are performed inside the recognizer such as the "Tree Recognizer" and the "Graph Recognizer" in Figure 1.1. The core of an LVCSR system is a complex decoder coupled with an acoustic model and language model. Both the acoustic model and language model for an LVCSR system are very large in scale and contain statistical information obtained from a huge database. During the decoding, the decoder will produce a rather rich content that is largely ignored, but we believe it is worth exploring in this case. We believe that the content produced during recognition represents complementary information in a multiple level aspect (Figure 1.2). To better use the complementary information, the decoder is a suitable platform that provides more reliable control on a much richer context. During-recognition fusion not only can reduce

[2]While there are minor differences in the contexts in which the term recognizer and decoder are used, we consider them to be synonymous in this thesis

the unrecoverable information loss occurring in pre- or post-recognition approaches but also provides a better framework to more fully use the information. Maybe the complexity of current decoders hinders research in this direction, but we have the knowledge and tools to conduct this study. The complexity of our approach can be limited to the design and implementation of the decoder. As a black box, the re-designed decoder can still be easy to deploy without much increase in operation cost. It is our **hypothesis** that by wisely using the complementary features during recognition, a significant gain can be expected. In this thesis, various fusion algorithms and implementation techniques are studied.

1.6 Overview of the Rest of the Thesis

Chapter 2 contains the background technology used in this thesis. This chapter will give a general description of the structure of current continuous speech recognition systems including acoustic model training and decoding. This chapter introduces the concept of *Hidden Markov Model*, the most widely used statistical approach for speech recognition. This chapter describes the Viterbi algorithm, which is used to conduct an efficient search in a speech recognition system, which we will revisit in Chapter 5. Various technologies are used to control the search cost. The technologies described in this chapter are closely related to the implementation of our run time fusion and we will use these concepts in Chapter 5. This background information is necessary to understand how we implement our fusion approach in an efficient way.

Chapter 3 describes the speech recognition tasks that are used throughout this thesis and the essential components of our baseline system. This chapter gives some introductions on the speech signal processing and the speech feature extraction. This chapter also contains the details of building the baseline system used in this thesis work. Some add-on technologies such as retraining, class-based language modeling were implemented to improve our baseline system. At the end of this chapter, we obtained a very competitive baseline system, which is about the best system we can get by using traditional techniques.

Chapter 4 gives a background review on some existing approaches to performing fusion in speech recognition. Information fusion in speech recognition is a relatively new and active research area. It was based on the research findings on human speech recognition (Section 4.1). Current fusion approaches can be roughly classified into two categories: pre-recognition (Section 4.2) and post-recognition combination (Section 4.3). We will give a review on these existing fusion approaches and their advantages and disadvantages.

Chapter 5 will present how we approach the problems in the existing fusion approaches, rooted in their inefficient use of the complementary information. We propose a run time fusion framework to address these problems. We further present the detailed design and implementation of our approach. Under the general high level fusion framework, we designed three different fusion approaches. These three approaches are based on the same hypothesis, that by applying complementary information at an earlier stage of the recognition process, the final system will be able to obtain much better accuracy. These three approaches differ from each other at when, where and how the fusion is performed. We investigate these three approaches (or system architectures) in the hope of making the best use of multi-information sources. Experimental results are given after each approach to demonstrate the advantages of our solutions.

Chapter 7 presents a new speech segmentation approach based on two levels of fusion. The first level of fusion applies to the spectral sub-bands and fuse multiple filter bank coefficients. This new approach takes advantage of current feature extraction procedure, with little additional computation cost. Another level of fusion was performed by fusing the results from several segmentation systems. Experiments show our fusion based approaches significantly reduced the WER compare to two classifier-based approaches. Compared to a manual segmentation, our approach only has 0.3% WER increase.

Section 8 summarizes this thesis work and describes out our future research directions.

Chapter 2

Background Technology

This chapter contains the background technology used in this thesis. Section 2.1 introduces the concept of hidden Markov model and its application in speech recognition. Section 2.2 describes the traditional acoustic model training approach. Section 2.3 describes the Viterbi algorithm, which is used to conduct an efficient search in a speech recognition system, which we will revisit in Chapter 5. The potential search space in an LVCSR task is prohibitive for a full search, and various technologies are used to tackle this problem (Section 2.4). The technologies described in this section are closely related to the implementation of our run time fusion and we will use these concepts in Chapter 5. This background information is necessary to understand how we implement our fusion approach in an efficient way.

2.1 Hidden Markov Model

The acoustic models mentioned above are basic entities used for modeling certain speech features. There are many modeling techniques in the history of speech recognition: dynamic template comparison, knowledge based matching, neural network, and *Hidden Markov Models* (HMM). These techniques have their own advantages in certain applications. HMM is so far the most widely used and most effective approach. Its popularity is mostly due to its efficient algorithms for training and recognition, and to its performance superiority over other modeling techniques.

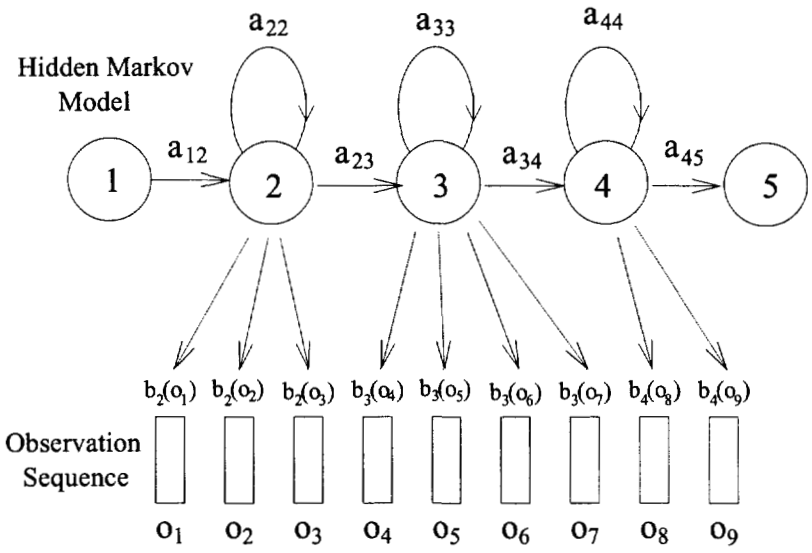


Figure 2.1: An example of Hidden Markov Model

There are 5 states in this figure and state 1 is the entry state and 5 is the exit state of this HMM. An HMM can be viewed as a finite state machine with transition probability a_{ij} from one state i to another j. The observation sequence is generated with output probability density $b_j(o_t)$ represented by Gaussian mixture densities. State 1 and 5 do not have associated observation probabilities.

An HMM has a certain number of states connected with directional arcs. It can be viewed as a finite state machine that changes its state once every time unit by following the arcs in the HMM topology. Figure 2.1 illustrates a typical three state HMM. (State 1 and 5 are pseudo states used for modeling a triphone.) The transition from state i to state j is determined by the probability a_{ij}. At each state j, a speech vector (observation) o_t is generated with probability density $b_j(o_t)$, represented by Gaussian mixture densities:

$$b_j(o_t) = \sum_{m=1}^{M} \omega_{jm} N(o_t; \mu_{jm}, \Sigma_{jm}) \tag{2.1}$$

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

The $\omega_{j,m}$ satisfy:

$$\sum_{m=1}^{M} \omega_{j,m} = 1, \tag{2.2}$$

and

$$N(o_t;\, \mu_{j,m},\, \Sigma_{j,m}) = \frac{1}{\sqrt{(2\pi)^D \mid \Sigma_{j,m} \mid}} \exp[-\frac{1}{2}(o_t - \mu_{j,m})^T \Sigma_{j,m}^{-1}(o_t - \mu_{j,m})] \qquad (2.3)$$

is a multivariate Gaussian distribution with mean vector $\mu_{j,m}$ and covariance matrix $\Sigma_{j,m}$, where D is the feature dimension and $(o_t - \mu_{jm})^T$ denotes the transposition of $(o_t - \mu_{jm})$.

An HMM contains two stochastic processes: a transition process accounts for temporal variability, and an observation process for spectral variability. These two stochastic processes have been successfully used to model the speech variability, and at the same time be flexible enough for building a practical system. For ASR, only the observed sequence of events is known and the underlying transition process is unobservable. This is why it is called a *"hidden"* Markov model.

There are two basic assumptions in the HMM based ASR systems:

1. The first-order N state Markov assumption claims that the current system status depends only on its previous N states:

$$P(s_t \mid s_{t-1}, s_{t-2}, ..., s_0) = P(s_t \mid s_{t-1}, s_{t-2}, ..., s_{t-N}) \qquad (2.4)$$

   where s_t stands for the system being at state s_t at time t. In most ASR systems, as shown in Figure 2.1, N is 1 which means only the previous state has influence on current status.

2. The observation independence assumption assumes that the observation probability of a state depends only on the state, regardless of when and how the state is entered.

Under these two assumptions, the joint probability of an observation sequence $O = o_1, o_2, ..., o_T$ and its corresponding state sequence $s = s_0, s_1, ..., s_T$ can be calculated as follows:

$$P(O,\, s \mid \lambda) = \prod_{t=1}^{T} a_{s_{t-1}, s_t} b_{s_t}(o_t) \qquad (2.5)$$

2.1.1 Likelihood Evaluation: the Forward and Backward Algorithm

Given an HMM represented by λ and an observation sequence $O = (o_1 o_2 \dots o_T)$, the occurrence probability of the observation $P(O|\lambda)$ can be theoretically calculated by summing Equation 2.5 over all possible state sequences. Let $\pi_{s_1} = a_{s_0 s_1}$ be the initial probability of state s_1.

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

Let o_1^t be the partial observation sequence of $(o_1 o_2 \dots o_t)$. The forward probability $\alpha_i(t)$ is defined as

$$\alpha_i(t) = P(o_1^t, s_t = i|\lambda) \tag{2.6}$$

which is the joint conditional probability of the partial observation sequence, $o_1 o_2 \dots o_t$ and state i at time t, given the model λ.

And the backward variable $\beta_i(t)$ is defined as

$$\beta_i(t) = P(o_{t+1}^T|s_t = i, \lambda) \tag{2.7}$$

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.

Initializing at time $t = 1$, the forward variable can be computed inductively using the following steps:

1. **Initialization**

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

2. **Recursion**

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

3. **Termination**

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

In the same way, we can initialize the backward variable at time $t = T$ and compute it inductively using the following steps:

1. **Initialization**

$$\beta_i(T) = 1 \quad 1 \leq i \leq N \tag{2.8}$$

2. **Recursion**

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

3. **Termination**

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

The total likelihood $P(O,\ s \mid \lambda) =$ in Equation 2.5 can be calculated by:

$$P(O,\ s \mid \lambda) = \sum_{i=1}^{N} P(O, s_t = i) = \sum_{i=1}^{N} \alpha_i(t) \beta_i(t). \tag{2.9}$$

When $t = T$, we can use Equation 2.8 to further simplify the equation above to

$$P(O,\ s \mid \lambda) = \sum_{i=1}^{N} \alpha_i(T). \tag{2.10}$$

So we only need the forward recursion for calculation $P(O,\ s \mid \lambda)$ and this forward recursion is called the forward algorithm. The counterpart recursion is called backward algorithm and is used in training to estimate the model parameter.

2.2 Acoustic Model Training: The Baum-Welch Algorithm

The goal of acoustic model training is to estimate λ by maximizing the probability of the observations $P(O \mid \lambda)$ over all training data. Solving the HMM training problem is difficult because the state sequence is hidden to us. There is no "correct" state sequence corresponding to a given observation sequence for all but the case of degenerate models. Therefore, no sufficient statistics of the state sequence is available to obtain a reliable estimation. Therefore it is impossible to analytically solve this problem. The best we can achieve is to obtain a λ such that $P(O \mid \lambda)$ is the local maximum for the available training data. The Maximum Likelihood (ML) estimation is usually obtained through the *Baum-Welch* algorithm, also called the *Expectation Maximization* (EM) algorithm [23]. For the case of discrete HMM, the re-estimation formula are straightforward 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

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{2.11}$$

In the case of continuous HMM, where the state output distribution takes the form of Gaussian mixture model (GMM) (Equation 2.1), the estimation formula for parameters of the GMM becomes:

$$\bar{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)}{\sum_{t=1}^{T} \sum_{l=1}^{M} \gamma_{jl}(t)} \tag{2.12}$$

$$\bar{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t) o_t}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{2.13}$$

$$\bar{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)(o_t - \mu_{jm})(o_t - \mu_{jm})'}{\sum_{t=1}^{T} \gamma_{jm}(t)} \tag{2.14}$$

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

$$\gamma_{jm}(t) = \gamma_j(t) \frac{c_{jm} N(o_t; \mu_{jm}, \Sigma_{jm})}{\sum_{l=1}^{M} c_{jl} N(o_t; \mu_{jl}, \Sigma_{jl})} = \frac{1}{P} \alpha_j(t) \beta_j(t) \frac{c_{jm} b_{jm}(o_t)}{b_j(o_t)} \tag{2.15}$$

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 Time Synchronous Recognition: The Viterbi Algorithm

As described in Section 1.2, the recognition procedure is a massive search over all the possible word sequences to find a word sequence $\hat{W}$ that maximizes Equation 1.3. Within the HMM framework, search for $\hat{W}$ is realized by searching all possible state sequences. We can use the forward probability for the likelihood calculation, and Equation 1.3 can be re-formulated as follows:

$$\hat{W} = \arg\max_{W} P(W)P(O|W) = \arg\max_{W} P(W) \sum_{s_1^T} P(O, s_1^T|W) \tag{2.16}$$

where s_1^T means all possible state sequences from time 1 to T. The summation in Equation 2.16 is for all possible state sequences under the constraint of word sequence W. Summing all possible state sequences will require a thorough search through the whole search spaces which is not affordable.

Therefore replacing "sum" with "max" in Equation 2.16, the new search equation becomes:

$$\hat{W} = \arg\max_{W} P(W)P(O|W) = \arg\max_{W} P(W) \max_{s_1^T} P(O, s_1^T|W), \tag{2.17}$$

where only the most probable state sequence is considered. This replacement is called the Viterbi approximation or maximum approximation: The most likely word sequence is **approximated** by the most likely state sequence. This approximation is certainly a sub-optimal assumption but in practice it works very well.

To find the best state sequence for a given observation sequence, the Viterbi algorithm, a dynamic programming method, is used. If we define the quantity $\phi_i(t)$ as the partial state sequence probability

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

then the basic Viterbi algorithm can be simply stated as

1. **Initialization**

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

2. **Recursion**

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

3. **Termination**

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

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 *argmax(i)* at each step of the Viterbi recursion and backtracking after the final result is found. By considering only the best state sequence at any time, the Viterbi algorithm does not need to store all the partial state sequences over time t. Therefore, it is memory efficient. Under the Viterbi algorithm, the search performs at both the state and word level and is executed in a time synchronous fashion in that it processes all states completely at time t before moving on to time $t + 1$.

2.4 Tree Based Time Synchronous Beam Search

There are two main components related to the computational cost of speech recognition: the acoustic model evaluation and the search. The former refers to the probability calculation of the acoustic models and speech observations. The latter refers to the search for the best word sequence that matches the given speech utterance. Both components are essential to a recognizer and need to be integrated together in an efficient way. For large vocabulary systems, the search cost is significant and a careful design of the recognizer is necessary.

2.4.1 Lexical Tree

In small vocabulary systems, acoustic models are organized in a flat structure. Each word has its unique acoustic models corresponding to its pronunciations. In the flat structure, the source and target word models of a crossword transition are known at the transition stage, thus it is easy to incorporate a language model into the search. When the vocabulary size increases to certain amount, the recognizer cannot afford to have such flat structures anymore.

In large vocabulary systems, the acoustic models are organized in a phonetic tree structure, called a tree-structured lexicon or *lexical tree* [63][38][69][64], as illustrated in Figure 2.2. In recognition, the search for the most likely word sequence is based on the lexical tree, a prefix pronunciation tree. Each tree node (except the pseudo root node) is a phoneme of some word pronunciations. Words with the same prefix pronunciations share the same acoustic models representing these pronunciations and share the same tree paths. Each leaf node is associated with a word whose pronunciation is represented by the path from the pseudo root node to the leaf node. When a search path reaches a tree leaf node (the word-end pronunciation), the search space is extended by copying the entire lexical tree under that leaf node. The tree-copy is a requirement in the traditional lexical tree search algorithm to form a unique search path (to determine the acoustic model to be evaluated) and to retrieve the final transcription. Since the search paths are frequently created and discarded during the search, the lexical trees are copied and discarded accordingly. These operations occupy a large portion of the total search computation.

The recognition is processed as the search paths pass through the lexical tree. The search is initialized from the pseudo root node. The context triphone is rendered and the corresponding acoustic model is used to calculate the acoustic score. The search path splits at the tree branches so that every possible path will be traversed.

At each leaf node, the language model score is attached. The word associated with the leaf node is recorded as the search history and the search path goes back to the pseudo

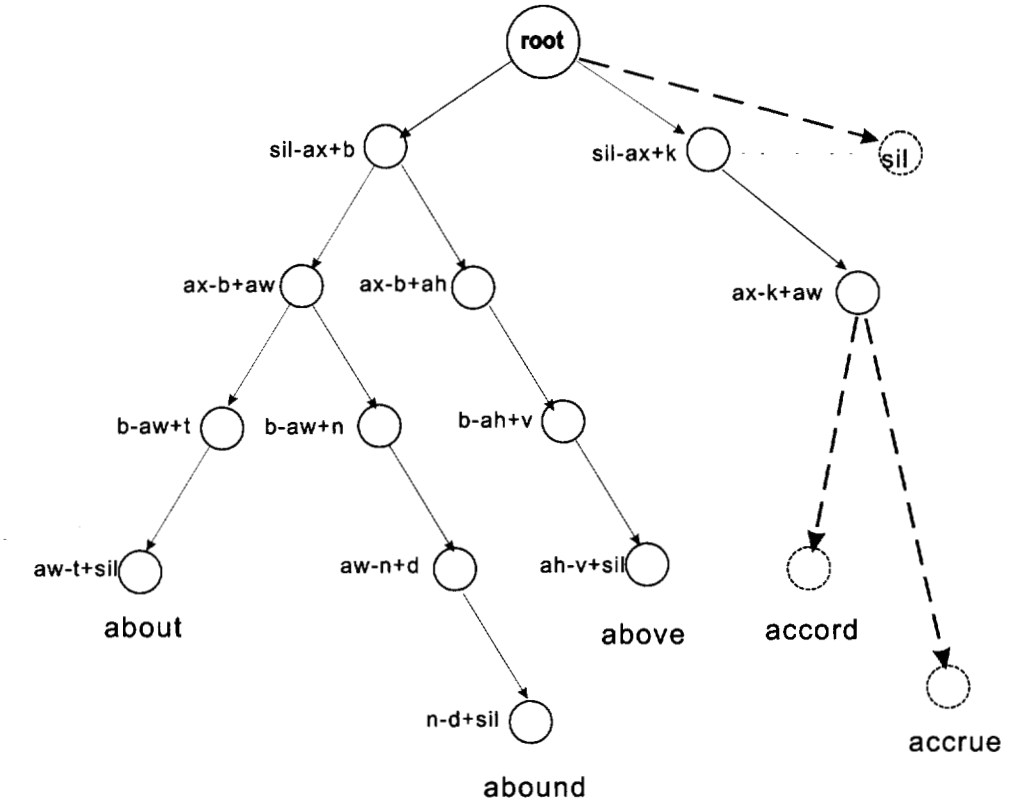


Figure 2.2: An example of Lexical Tree

Words with the same prefix pronunciations share the same tree paths. The search is performed as the search paths passing through the lexical tree. Each tree node is associated with a triphone HMM model.

root node for searching the successive words.

At the end of the search, the search path with the best likelihood score is traced back to retrieve the most likely word sequence and output as the recognition result.

The lexical tree structure has the following advantages [2]:

1. By introducing a high degree of sharing at the root nodes, the lexical tree structure reduces the number of word initial (acoustic) models that need to be evaluated. (Word initial models are the most frequently evaluated models in the flat structure systems.)

2. The tree structure also greatly reduces the number of crossword transitions, which

is again a dominant part of the search.

The problems with a lexical tree occur at the word boundary: (1) The application of a language model has to be delayed because the target word of a crossword transition is unknown at the tree root nodes. (2) When the search paths cross the word boundary, the search space needs to be extended. And the search paths need to be unique after the extension.

The traditional way to solve the problem is to copy the entire lexical tree (or to copy the tree-layers on demand). The tree-copy requirement results in high overhead (of CPU time and memory) for manipulating the lexical tree. The new lexical tree search algorithm in our system solves the problem by decoupling the search and the search space. The lexical tree is re-entered instead of being physically copied.

A typical lexical tree is illustrated in Figure 2.2. Each tree node is associated with an acoustic model (a mono-phone model in the figure). Each path from the pseudo root node to a leaf node (also called a word-end node) represents the pronunciation of a word. The lexical tree is constructed as follows:

1. The pronunciation of a word is represented by a series of mono-phone tree nodes. Acoustic models are associated with the tree nodes. The word is indicated by the corresponding tree leaf node. Tree nodes are shared by words with the same pronunciation prefixes. If a tee-model[1] is inserted as the last pronunciation of a word, the word is also indicated by the mono-phone node right above the tee-model.

2. The sentence-end is represented as a special branch of the lexical tree. The sentence beginning is represented as a stand-alone lexical tree.

[1] A tee-model is an optional silence model with very short duration. It is a special model because it is usually skipped when a left-context (or right-context) mono-phone is considered.

2.4.2 Token Structure

To search through the lexical tree, we need some structures to record the search status and history. The "token passing" [95] concept is used and extended in our algorithm to achieve the lexical tree re-entry. A token is a data structure to represent a (partial) search path at the current time frame. Each search path can be viewed as a token passing through the lexical tree. Additional data structures are carried with the tokens to distinguish the search paths on a tree node (1) with different re-entry times, or (2) re-entering the lexical tree from different words.

To represent a search path, a token has the following elements:

1. **NODE**: a tree node index indicating where the token resides in the lexical tree.
2. **NEXT**: A tree node pointer indicating what is the next move. If **NODE** is a tee-model, **NEXT** is the move after evaluating the tee-model.
3. **TRIPHONE**: the triphone model. If **NODE** is a tee-model, **TRIPHONE** is the last triphone model evaluated. Otherwise, **TRIPHONE** is the current triphone model to be evaluated.
4. **TEE**: A tee-model pointer pointing to the tee-model node if the next evaluating acoustic model is the tee-model. Tee-model is a special phone model that needs special treatment.
5. **Log Likelihood**: The log likelihood value of all the internal states in the triphone model.
6. Pointer to Word Link List (**WLL**): **WLL** is a partial history list containing a list of words that the token passed by. The **WLL** also contains the time t and log likelihood value information. Upon the finish of the recognition, a back trace through the **WLLs** will retrieve the most likely word sequences. It is necessary to build a word graph after the recognition.

The token passing algorithm is implemented in the Viterbi decoder by the following procedures:

1. At time *t*, a token *T* reaches the HMM state *i* of lexical tree instance *h*. *T* represents the current best partial path that starts from time *0* to time *t*, which is the match between the acoustic observation sequence, o_1 to o_t and a sequence of HMMs ending at state *i* of lexical tree instance *h*.

2. At time *t+1*, the token is passed from state *i* to all traversable states on the lexical tree. Each state *j* of these states gets a clone of token *T*.

3. The elements of the cloned token are updated accordingly:

   - **NODE** is updated with the new node index number of the current tree node.
   - **NEXT** is updated according to the topology of the lexical tree.
   - **TRIPHONE** is updated if the current triphone is different from time *t*.
   - **TEE** is updated if the current model is a tee model.
   - **Log Likelihood** is incremented by adding the transition probability and state probability $\log(a_{ij}) + \log(b_j(o_t))$.
   - **WLL** is updated if it just exited from a word end node.

4. For each state *j*, pool and rank all the tokens, discard all tokens except the one with the highest probability.

Since each HMM state *s* was represented by a node in the lexical tree and the lexical tree was re-used by each new word instance, more than one token can propagate to state *s* at time *t*. We pool those tokens together as a token list *TL(t;w;s)*. *w* means a language model state, it is necessary because we have to take account the word history information. At the same time frame, more than one lexical tree representing the same grammar word could be entered. Each one has a different language model state in the search space since

they have different previous words. Each token in *TL(t;w;s)* can be distinguished by its partial path score $\Gamma(t; w; s)$ and partial path history *H(t;w;s)*.

2.4.3 Lexical Tree Search

The search is initiated from the pseudo root node of the sentence-beginning lexical tree. At each time frame, tokens split and move. Usually, a token splits itself into several tokens: one of the tokens remains at the particular tree node and is used to record the model internal transitions; other tokens advance to all the child tree nodes. The advancing tokens will have their internal data structures changed to reflect the move. When a tee model is encountered, a token makes additional splits (in addition to the normal splits mentioned above). The split tokens skip the tee model and go directly to the re-entry process.

When a token reaches a word-end node, the lexical tree is re-entered. The token passing continues. At the end of the speech utterance, the search paths in the special sentence-end branch are sorted to give the transcription.

The search is conducted on the lexical tree by decoupling the search and the search space. The search space is extended by re-entering the lexical tree instead of copying the lexical tree. Because a thorough search is neither affordable or necessary, multiple pruning methods are implemented to control the span of the actual search paths. The surviving search paths are recorded by the tokens, and a back tracing through the tokens can reconstruct the actual search space which is much smaller compared to the original search space. Experimental results show that this lexical tree method takes much less CPU time and memory to achieve the same performance as the traditional lexical tree search algorithm.

2.4.4 Lexical Tree Beam Search

By associating path information with tokens instead of tree nodes, the major drawback of copying the entire lexical tree in the traditional lexical tree search algorithm is avoided, and thus it saves time and memory resources. However, the number of tokens associated with a single tree node increases accordingly, which results in increased computation on each node, especially on the word-end nodes. To address this new problem, pruning is applied to reduce the number of active tokens on each tree node.

Beam Search [55] [63] is one solution used to limit the search space by pruning away the less likely hypotheses. The paths that fall below a certain threshold from a reference path are removed from further propagation. The threshold or the beam width is usually decided by trial and error on a development data set. It is a tradeoff between speed and accuracy; a larger beam width usually means higher recognition accuracy with slower speed.

From the point of view of computer algorithms, Beam Search is an expansion of hill-climbing search: instead of just keeping one state around, several states are kept. Although it can alleviate some local optima problem inherent to hill-climbing, Beam Search is still an incomplete and inadmissible search. It is most useful when the search space is big and the local optima are not dominant. Both are a good fit to lexical tree search.

The standard Beam Search algorithm under our lexical tree search framework has the following steps at each time frame t:

1. Search starts from the root node of the lexical tree and propagates to all entry nodes of the lexical tree.

2. For each active node N (active means there is at least one token with time stamp t that resides on that node):

   (a) Perform Token Passing as outlined in Section 2.4.2.

(b) Find the maximum log likelihood for node N and record it as $MaxLog(N)$.

(c) If $MaxLog(N)$ is larger than the global maximum value $MaxLog$, set $MaxLog$ to $MaxLog(N)$.

3. Perform pruning:

   If ($MaxLog(N) > (MaxLog - BeamWidth)$),

   propagate those tokens to the next frame;

   else

   stop propagating those tokens and deactivate their corresponding nodes.

4. If the token leaves the leaf node and re-enters the lexical tree at the root node, repeat from step 1.

In our system, we have several different pruning steps that are performed every time frame.

- State acoustic pruning: State acoustic pruning is the standard beam pruning approach as detailed above.

- State overall pruning: Similar to the state acoustic pruning, the state overall pruning is used to retain only hypotheses with a score close to the best state hypothesis. The difference is the language model probabilities are added into the score before state overall pruning.

- Word-end pruning: Besides the normal state pruning method, an extended word-end pruning method is implemented. This pruning method is an extension to the traditional word-end pruning. Assuming that acoustic models are well trained, the best path will dominate the search most of the time. Thus, each word-end may allow only certain fan out[2] without performance loss. This concept is extended to every tree node.

[2]Fan out is the number of words that is associated with the successor of the current tree node.

- Histogram pruning: After the previous pruning steps, another round of histogram pruning is performed to control the maximum number of surviving tokens. Histogram pruning is done by examining the total number of active tokens (states) on all tree nodes. If the number exceeds a given threshold ($MaxTokens$), then only the best $MaxTokens$ tokens are allowed to continue.

- Token merge pruning: A token merge step was implemented in our system as described in Section 6.1.1. The token merge beam is used to set a threshold for merging some tokens.

- Other pruning: Language model lookahead pruning and Phoneme lookahead pruning are also used in our system.

2.4.5 Word Graph

As detailed in Section 2.3, the Viterbi algorithm is essentially a dynamic programming algorithm conducting a time synchronous search that processes the input speech one frame at a time. The Viterbi search calculates the best path score at each state at a given time t. It will move on to time t+1 after all states are processed at time t. When it reaches the last frame, a backward trace will generate the most likely phone and word sequence. When we connect those word sequences together, it is an acyclic graph representing the recognition hypotheses. This graph is called a word graph [70], which is a subset of the original search space.

For most ASR tasks, the search space constituted by all possible combined state sequences is too prohibitive to conduct a thorough search. A beam search (Section 2.4) is an approach used to limit the search space by pruning away the less likely partial paths before they reach the end of utterance. Another way to reduce the search cost is a multiple pass approach in which the first pass generates the word graph using simple acoustic and language models, then successive passes re-score the graph using more complex knowledge sources such as long span acoustic and language models. Although the complex

knowledge sources are more accurate, they are too costly to be deployed during the early pass search. Most of the time, the later pass decoding will achieve a higher accuracy with only a relatively small cost compared to the first pass decoding. The overall computation cost of multiple pass decoding is usually much lower compared to single pass decoding.

In the multiple pass approach, the word graph (Figure 2.3) serves as an intermediate recognition output representing the high-ranking sentence hypotheses in the form of a graph whose edges are words. Any path from the sentence-beginning node "S" to the sentence-end node "E" is a valid sentence hypothesis (generated by the recognizer). The horizontal axis shows the time scale and the vertical axis is for the purpose of displaying the hypotheses in parallel. Each node in the word graph represents a word transition time. The edges and nodes of a word graph not only show when the word transitions happen but also their sources and destinations. The word graph can be interpreted as a reduced search space, where the number of possible words is reduced and possible connections are restricted.

Compared to the N-best sentence lists, the word graph has some clear advantages. First, a word graph is a more compact representation because sentence hypotheses can share edges and nodes; the N-best list makes separate entries for every difference along the hypothesis path. Secondly, a word graph is much more powerful and flexible. It contains more information than an N-best list, such as partial hypothesis score, competing paths, and their relationship. Additional information is easier to incorporate into the word graph structure that is essential for integrating new knowledge sources and support multiple sources simultaneously. Most times, the N-best list is extracted from the word graph. Furthermore, the N-best list can be treated as a subset of a word graph.

Applications of Word Graph

Besides serving as an intermediate search space between two sequential searches, the word graph has been used in many other ways:

1. post-recognition approaches:

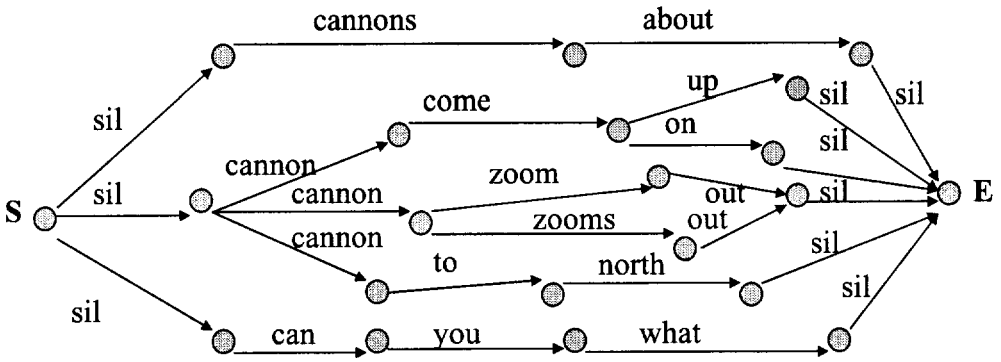


Figure 2.3: A sample diagram of Word Graph

"sil" means non-speech event.
The corresponding utterances are:

- cannons about.
- cannon come up.
- cannon come on.
- cannon zoom out.
- ...
- can you what.

Instead of performing another recognition process, post-recognition approaches manipulate word graph directly. Examples such as Hypothesis Combination is introduced in Section 4.3.2.

2. language model re-scoring:

   Higher order language models or syntactic taggers, which are not able to be directly integrated into the recognizer, can search through a word graph to find an optimal path.

3. Confusion Network:

   Confusion Network [56, 57] is an approach that aims to minimize the WER by post-processing the word graph. It aims to solve the mismatch problem between

the current word-based performance criteria and the standard MAP decoding that is sentence-based. The word graph is clustered into a linear graph called a consensus network (Confusion Network). The final word sequence that minimizes the WER can be found by selecting the word hypothesis with the highest posterior probability from the confusion network.

4. speech understanding system:

   In speech understanding system, the target is to get the essential meaning of the speech rather than to get all the words recognized correctly. Lingustic, syntactic and semantic knowledge of language is normally incorporated into a parser to reach this goal. In dialogue management, if the 1-Best hypothesis is not right, an alternative hypothesis can be dynamically prompted according to a user's response.

5. confidence measurement and word spotting:

   A word graph contains many competing hypotheses at word and sentence level; a number of features can be delivered from a word graph for confidence measurement and word spotting.

Chapter 3

Tasks and Baseline System

This chapter describes the speech recognition tasks that are used throughout this thesis and the essential components of our baseline system. Section 3.1 presents a brief overview of the training and testing corpus. Two corpora, TIMIT and SPeech In Noisy Environments (SPINE), are used to build acoustic models. The test corpora is the DARPA SPINE evaluation set. Section 3.2 gives some introductions on the speech signal processing and the speech feature extractions. Section 3.3 contains the details of building the baseline system used in this thesis work. First, in section 3.3.1, we explain the main components of our in-house LVCSR software package. Then we describe the acoustic model training algorithms and procedures used to obtain different acoustic models for each features. An unique training method called 'Retrain' is briefly introduced (Section 3.3.3). Section 3.3.5 contains details of recognition experiments performed, such as results obtained from different features. In this thesis work, much effort was devoted to improving the performance of our baseline system. A class based Language Model is used to further improve the performance of our baseline system (Section 3.3.6). At the end of this chapter, we obtained a very competitive baseline system, which is about the best system we can get by using traditional techniques.

3.1 Speech Corpus and Tasks

Two databases are used in our experiments: TIMIT corpus and SPeech In Noisy Environments (SPINE) corpus. TIMIT is used only for initial training of acoustic models. The SPINE corpus is one of the latest databases, for DARPA sponsored large-vocabulary continuous speech recognition evaluation conducted in 2000 and 2001.

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. The labels consist of two integers (start and end sample-numbers[1]) followed by an ASCII ARPAbet representation of the standard IPA phonetic symbol.

Orthography:

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

The phonetic transcription is very important for initial training (bootstrapping) of each individual HMM phone model, as we will explain in Section 3.3.

3.1.2 SPeech In Noisy Environments

Recent research efforts have focused on robust speech recognition under noisy environments. The newest deployment of speech technology are relevant to telephone conversation and anywhere environments. One great challenge to such applications is the great

[1]Note that these integers are sample-numbers, not milliseconds, or other units of time.

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#

Note: beginning and ending silence regions are marked with h#

degree of variation of speech signal. The source environments and transmission channels may be noisy or variable, resulting in a distorted speech signal from the origin. Also the noise can put the speaker under stress and so may express a variety of emotions reflected in their speech.

Noise or limited bandwidth channels have provided a real challenge to current speech recognition technology. Lots of effort has been undertaken to tackle the problem but only with limited success. To evaluate the current state of the art in speech recognition under noise, especially military noise, the Naval Research Labs organized the first SPeech In Noisy Environments evaluation (SPINE1) in August 2000. With the success of the first evaluation, the second SPeech In Noisy Environments (SPINE2) evaluation was conducted in November 2001 [NRL, 2001].

The SPINE1 data was collected from 22 pairs of speakers. Each pair of speakers participated in a Milton Bradley™ battleship game. Each pair of speakers worked in a cooperative way to locate and sink ships on a grid. Each conversation session was complicated by the introduction of noise and the confusable grid labels. Each pair of speakers

Table 3.1: Scenarios in SPINE1

	Recording Room 1		Recording Room 2		Channel
Scenario	Noise	Handset	Noise	Handset	
DOD	Quiet	STU-III	Office	STU-III	POTS with STU-III
Navy	Aircraft Carrier	TA840	Office	STU-III	HF
Army	HMMWV	H250	Quiet	STU-III	Satellite Delay
Air Force	E3A AWACS	R215	MCE	EV M87	JTIDS

were located in separate sound recording rooms. Four scenarios were combined by realistic noise, handsets, communication channels and vocoders from the military operations, as shown in Table 3.1.

Twelve different vocoders were applied on the transmissions between booths. A pair of speakers also switched booths and repeated the same session. Overall, with four scenarios, twelve vocoders and two speakers, each speaker pair worked through 96 sessions. Although the speech signals excluded the vocoder effects, the stress of listening to vocoded speech is reflected in the speech of those participants.

The difficulty of an ASR task can be measured by the recognition performance of a certain recognizer, namely the word error rate (WER). On the other hand, we can estimate the difficulty of an ASR task by analysis of its characteristics. Both theoretical and practical studies show that the ASR task becomes more difficult along the following dimensions:

- **Vocabulary Size**

  The larger the vocabulary size, the more confusion the recognizer needs to resolve. However, missing coverage of possible words is more expensive than false inclusion of some non-appearing words. The optimal vocabulary is a balance between reducing OOV (out of vocabulary) words and reducing the total number of words. Current technology requires the vocabulary to be task oriented, which means tailored to the specified application. Generally, the English ASR systems are classified

by their vocabulary size into:

- **small**: less than a few hundreds words; such as voice commands on small devices.
- **medium**: around a few thousands words; such as database management.
- **large**: more than 5,000 words; such as Wall Street Journal, Broad Cast News.
- **super**: at least several 10,000 words; such as dictation systems.

- **Speaker Mode**

  Most state-of-the-art ASR systems are **continuous** which means each speech utterance can contain more than one word. Some ASR systems are still **isolated**, which requires the speaker to speak one word at a time. Generally continuous ASR is more challenging than isolated word recognition.

- **Speaker Dependency**

  ASR systems are speaker dependent or speaker independent. **A Speaker Dependent (SD)** ASR system is trained or adapted on the speech of a specified test speaker. **A Speaker Independent (SI)** system is trained on speech data from various speakers and not targeted to any specified speaker. A speaker dependent system generally performs better than a speaker independent system.

- **Channel and bandwidth**

  Different communication channels have different distortions. Some channels limit the bandwidth, such as telephone line, but some are able to provide high bandwidth. Generally low bandwidth causes the loss of information thus increasing the recognition errors.

- **Acoustic environment**

  The type and degree of background noise can significantly reduce the recognizer performance. Non-stationary noise is more harmful than stationary noise. Background human speech is more confusing than non-human speech.

- **Speaking Style**

  Read speech such as dictation and news report contain less variety. Spontaneous speech such as casual talk is more difficult for ASR system because it contains more non-standard speech. Non-standard speech, such as hesitations, repeats, murmurs, cannot be well modeled by current acoustic and language modeling technologies. Experiments were conducted on identical sentences that varied in speaking style [85]. The Word Error Rate increased from 28.8% for read dictation to 52.6% for spontaneous conversation.

- **Language Model**

  A language model with a constrained grammar is more beneficial for a recognizer than a non-constrained one. A lower perplexity language model generally can improve recognition performance compared to a high perplexity one because it restricts the possible word end fanout.

- **Acoustic Training Data**

  Generally speaking, the more speech data for acoustic training the better because the acoustical model can have a more accurate statistical estimation. Also the task domain data is preferable to out-of-domain data.

- **Language Model Training Data**

  Similar to acoustic training data, the more task oriented data for language model training the better. However, in most cases, the available training data within the task domain is quite limited.

- **Computing Resources**

  The performance of an ASR system is strongly tied to its available computing resources, such as memory size and CPU speed. For applications that require real-time response, some compromises have to be made between recognition accuracy and recognition time. With the same computing resources, the smaller the real time

Table 3.2: Characteristics of SPINE Task

	Easy ⇒ Difficult	SPINE2
Speaking Mode	Isolated ⇒ Continuous	Continuous
Speaking Style	Read ⇒ Spontaneous	Spontaneous
Speaker Enrollment	Dependent ⇒ Independent	Independent
Vocabulary	(Small <1,000 words) ⇒ Large (>5,000 words)	Medium
Noise Level	Low (SNR>30db) ⇒ High(SNR<10db)	SNR: 5-20db
Noise Type	Seen ⇒ Unseen	varies
Channel Type	Close talk Microphone ⇒ Cell Phone	varies

factor, the more difficult the recognition task is.

The SPINE data also contains some unusual phenomenon for speech recognition systems. There are quite a few non-lexemes that appear in SPINE data. Speaker noises such as coughing, laughing and breathing are common. Because the speakers need to use push-to-talk handsets, truncations happen frequently. There are also a large number of words spelled out by the speakers to disambiguate some easily confusable words. Although it is easier for speakers to understand, it is difficult for speech recognition systems. There are also noticeable mispronunciations and unintelligible portions of speech in the SPINE data.

The SPINE1 and SPINE2 corpus consist of several parts:

- **SPINE1**

  SPINE1 training data:

  - 10 speaker pairs, 20 speakers overall
  - 4 environments including quiet, office, HMMWV and AC carrier
  - DRT (Diagnostic Rhyme Test) in 2 noise environments
  - grids were labeled with words from the DRT and quiet

  SPINE1 test data:

Table 3.3: SPINE2 Training and Development Data

Training Data	number of Utterances	Hours
SPINE1 Train	11,973	8.7
SPINE1 Eval.	12,079	7.3
SPINE2 Train	6,129	3.2
SPINE2 Dev.	1,941	1.6

  - 20 speaker pairs, 40 speakers overall
  - 6 environments including two new - E3A and MCE

- **SPINE2**

  SPINE2 training data:

  - 6 noise environments
  - grids were labeled with words from a military vocabulary
  - 2 talker pairs (4 speakers total) with 32 conversations (sessions) per talker pair (64 conversations total).

  SPINE2 development data:

  - 2 talker pairs (4 speakers total) with 16 conversations (sessions) per talker pair (32 conversations total).

  SPINE2 test data:

  - 16 talker pairs (32 speakers total) with 4 conversations (sessions) per talker pair (64 conversations total).
  - 8 environments including two new - E3A and MCE
  - total of 7 hours (423 minutes) of audio data

The test data comprises 128 speaker-environment pairs with 8 different noise environments. The test data has unseen speakers and noise types from the training data, so there will be unavoidable speaker and environment mismatch between the training and test data.

Some utterance samples in the SPINE2 corpus are given in Figure 3.1 together with their corresponding recognition output.

```
REF: SAY IT I CAN'T HEAR YOU
HYP: *** ** * ***** THAT AGAIN

REF: confirmed *** ** OKAY HERE we GO DOING A (RADA-) OH
HYP: conf- GOT OH YEAH DON'T we GOT ONE ALL RIGHT I'LL

REF: okay RUN TEST SERIES ONE TWO THREE
HYP: okay *** **** RENTED RIGHT I THINK

REF: the sweep coordinates FROM THE acoustic sweep are north to north east
HYP: the sweep coordinates **** AN acoustic sweep are north to north east
```

Figure 3.1: Some sample utterances in SPINE task

Lines starting with "REF" are the reference transcription of that utterance. Lines starting with "HYP" are the corresponding recognition hypothesis. The words in capital font are mis-recognized.

3.2 Signal Processing and Feature Extraction

Both the training and testing speech data must first be processed before being used by the speech recognition system. In this thesis work, information fusion is based on the complementary information contained in different features. Thus it is necessary to give some introductions on the speech signal processing and the resulting speech features.

The speech signal is highly redundant because of the strong correlation between adjacent segments. Use of the raw signal is not only too expensive but also unmanageable. 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.

Just as feature extraction is important in any pattern recognitions, it is an important part of a successful speech recognition system. Most speech feature extraction approaches are based on the study of the human auditory system and researchers' intuitions. Over the years, various types of parametric representations 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. Due to the physical limitations of human vocal production, the speech signal can be treated as stationary at that short period of time.

Another argument is that successful feature extraction should be able to retain the useful 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. The short-time analysis is suitable for speech coding but may not be a good candidate for speech recognition. In speech coding, the goal is to preserve perceptual components of the signal, possibly for restoration. However for speech recognition, some of the perceptual components are harmful to retain, such as the communication channel information and the emotion of a speaker.

Although there are many different feature extraction methods, they each have their own advantages and disadvantages. More importantly, they have to be integrated with speech recognition modules. The feature extraction algorithms may have different relative performances under different recognition modules or tasks. The most successful and commonly used acoustic features for recognition purposes are Mel-Frequency Cepstral Coefficients (MFCC) [22] and Perceptual Linear Prediction (PLP) [39]. In this thesis work, the complementary information contained in different features are fused within a

new recognition architecture. Since these different speech features are important components of this thesis work, we detail some of the feature extraction methods that we used.

3.2.1 MFCC

We illustrate the procedure of extracting MFCC features below as it was used in all the experiments in this thesis.

1. Speech sampling: 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. Spectral analysis: A Hamming window of 25 ms is used to perform the short-time analysis. The window is shifted every 10 ms. These windows are overlapped to provide a greater frequency resolution.

3. Pre-emphasis: A pre-emphasis filter $H(z) = 1 - 0.97z^{-1}$ is applied to get rid of the lip effect [58].

4. Fast Fourier Transform (FFT) is applied to obtain the spectral representation, followed by a logarithm conversion.

5. Mel-spaced filterbanks are used to map the spectrum of linear scale in 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. There are several advantages to performing such a conversion. One reason is that cepstral parameters are a more efficient compression and thus provide a more compact representation than filterbank parameters. Secondly, mel-scale filterbank parameters are highly correlated and require a large number of parameters to model their distributions. In the cepstral domain, it is safer to make the assumption that the parameters are independent. In practice, conversion to cepstral domain allows using diagonal covariance

matrices with little performance degradation. This dramatically reduces the computation cost of HMM training and decoding. Recently, researchers also found that in cepstral domain it is much easier to get rid of some channel distortions, using techniques such as Cepstral Mean Subtraction/Normalization.

7. The first 12 coefficients are preserved, which become the base of our target MFCC feature vectors.

Figure 3.2 gives a diagram of the acoustic feature processing for producing MFCC features in our system. Note that energy information is also extracted for every frame and appended to the MFCC feature vector, giving the 13th dimension.

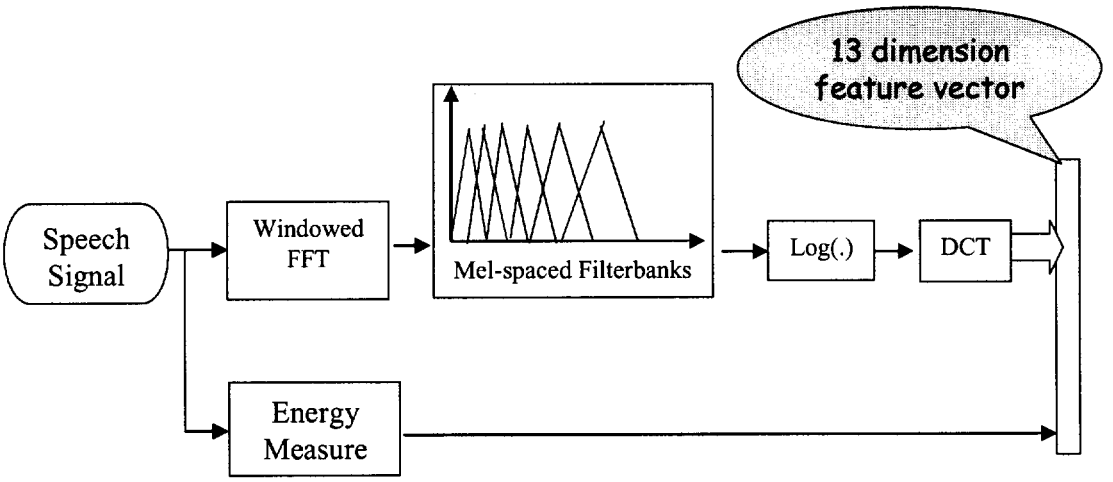


Figure 3.2: Extracting the base acoustic feature vectors of MFCC from speech data

The spectral pattern of a frame only contains local and static information about a sound. Since dynamic characteristics of temporal features play an important role in human perception, it is necessary to use some dynamic features to capture the temporal change. The most common method of obtaining dynamic feature is to estimate the delta and acceleration of the spectral coefficients over a series of consecutive frames, and

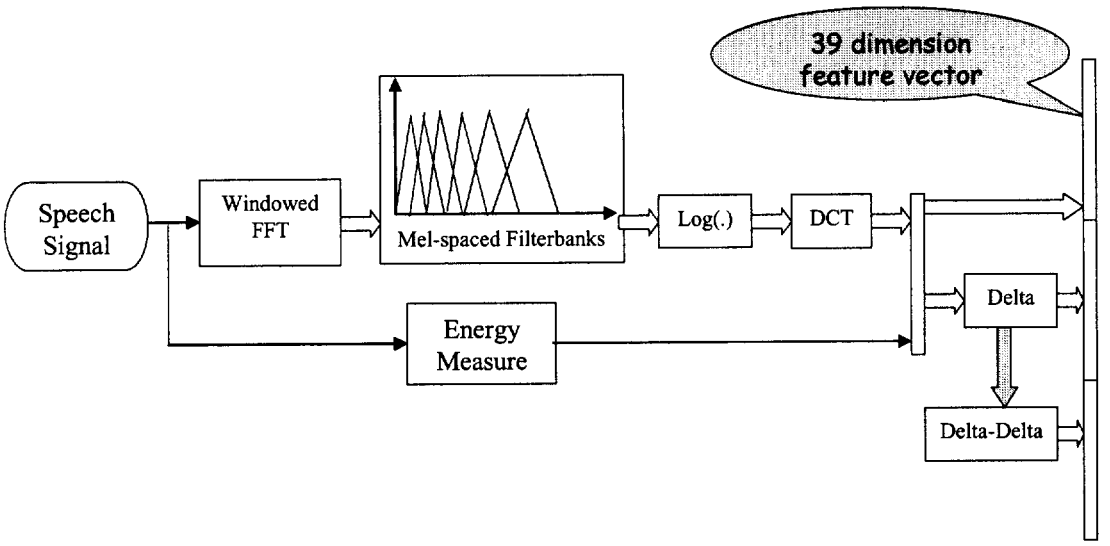


Figure 3.3: Diagram of extracting MFCC acoustic feature vectors from speech data

then append these measurements to the basic static feature vectors [31]. The success of these dynamic features is also due to their complementary nature to HMMs. Because the time independence assumption of HMM assumes each frame is independent of the other frames, dynamic features weaken this unsound assumption by broadening the duration of a frame.

Usually, a linear regression equation is used for the calculation:

$$d_i(t) = \frac{\sum_{k=1}^{N}(c_i(t+k) - c_i(t-k))}{2\sum_{k=1}^{N} k^2} \tag{3.1}$$

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 (Figure 3.3).

3.2.2 TRAPS (TempoRAl Pattern)

MFCC, along with most feature extraction methods in ASR is based on the spectral envelope of speech. An inherent problem with the spectrum of speech is that it is sensitive to many non-linguistic factors. Those factors such as frequency response of communication channels or frequency selective noise, have little effect on human speech understanding.

To reduce this kind of sensitivity, a multi-band paradigm [11] [42] adopts a series of spectral subband classifiers. Each subband only considers a part of the whole spectral envelope. Since only some of the subbands were corrupted by the frequency-selective noise, this paradigm allows noise robustness by reducing the damage caused by those unreliable subbands.

The traditional feature extraction methods also use the short-term (10ms) spectral envelope and process it solely on the frequency domain. The frequency domain features are better at representing features such as formats and provide more accuracy than in the temporal domain. However recent studies have found that the phonetic information of one phoneme is spread across several neighboring phonemes. Some important linguistic characteristics, such as the phoneme and the syllable, are much longer than the conventional short-term spectrum. Modeling syllable-length information requires the feature extraction to be conducted also on the temporal domain.

TRAPS was proposed by Dr. Hermansky's group as an innovative approach to solve some of the problems of traditional feature extractions [41]. It extracts features in the time-frequency domain under the multi-band paradigm. As a radical departure from conventional feature extraction, TRAPS uses a rather longer temporal envelope: 1 sec window (101 frames at 10 ms frame rate). Similar to the Multi-Band approach, there are 15 critical bands across the frequency space. The energy of each single critical band is calculated based on these temporal vectors. It assumes independence among different frequency bands in the early stage of speech communication.

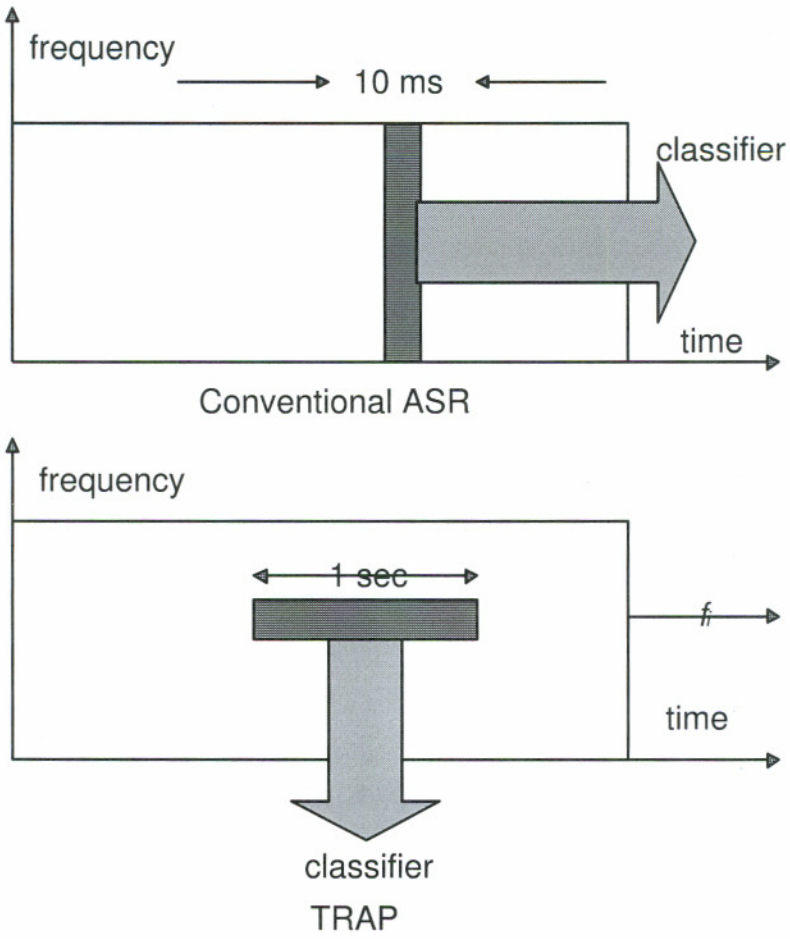


Figure 3.4: Comparison of TRAPS feature with conventional features

The design of TRAPS allows it to apply mean and variance normalization of the critical band spectrum, thus making it more robust to linear filtering of the signal and stationary noise. It is also more robust to frequency selective noises because of its relation to the Multi-Band approach.

The procedure that we used to produce TRAPS features is detailed as follows [66],[41], [91]:

1. Adjacent pairs of rather long (about 1 second) temporal trajectories of critical-band spectral energies formed the vector space for LDA-based projection. The 15 most

relevant 2-d discriminants are used to project 202 point feature vector (101 features from each adjacent critical band) into a 15 dimensional vector space. The projected features are given as input to multi-layered perceptrons (MLP). The output units of MLPs are the acoustic targets defined by 30 phonetic categories. The 56 phonemes which cover the whole SPINE-1 development set are grouped together to obtain the targets. The grouping of phonemes is based on the similarity of Articulatory properties namely Manner, Place, Voicing, and Height.

2. The second (information merging) stage uses MLP to combine the information from the individual frequency-localized classifiers. The output from this MLP represent estimates of posterior probabilities of the underlying phonetic features. To circumvent the skewed distribution of the estimated posterior probabilities, the final softmax nonlinearity in the output layer was removed from the trained MLP. The size of the hidden units is kept at 300 for band-specific MLPs and at 500 for merging MLP. First and second order dynamic features (speed and acceleration) are computed on trajectories of these probability estimates. Whitening transform (Karhunen-Loeve) is used to reduce the dimensionality and de-correlate the output. This results in 39 dimension feature vectors, which form the input to the recognizer. All linear and nonlinear transformations described above are derived on force-aligned SPINE1 development data.

3.2.3 TLDA (Two dimensional Linear Discriminants Analysis)

Studies show that the influence of the current phone extends beyond its boundaries, into its surrounding phones. Conventional feature extraction using a 10 ms time span is too short to cover most phone duration. To incorporate a longer time span that can match the duration of a phone, a wider block of the spectrogram is used. But use of a long time span incurs some problems. First, these features are not independent but highly correlated. Second, the feature dimensions are increased dramatically. It not only requires a huge

amount of training data but also stresses the training procedure.

To solve these problems, Linear Discriminants Analysis (LDA) is used to reduce redundancy and obtain a smaller size feature set. The final feature is projected on the joint time-frequency domain. In the spectral domain, the discriminants are based on short-term spectral energies. In the temporal domain, the discriminants are based on time trajectories of the spectral energies. The joint features are formed by concatenating short-term spectral frames along the temporal span.

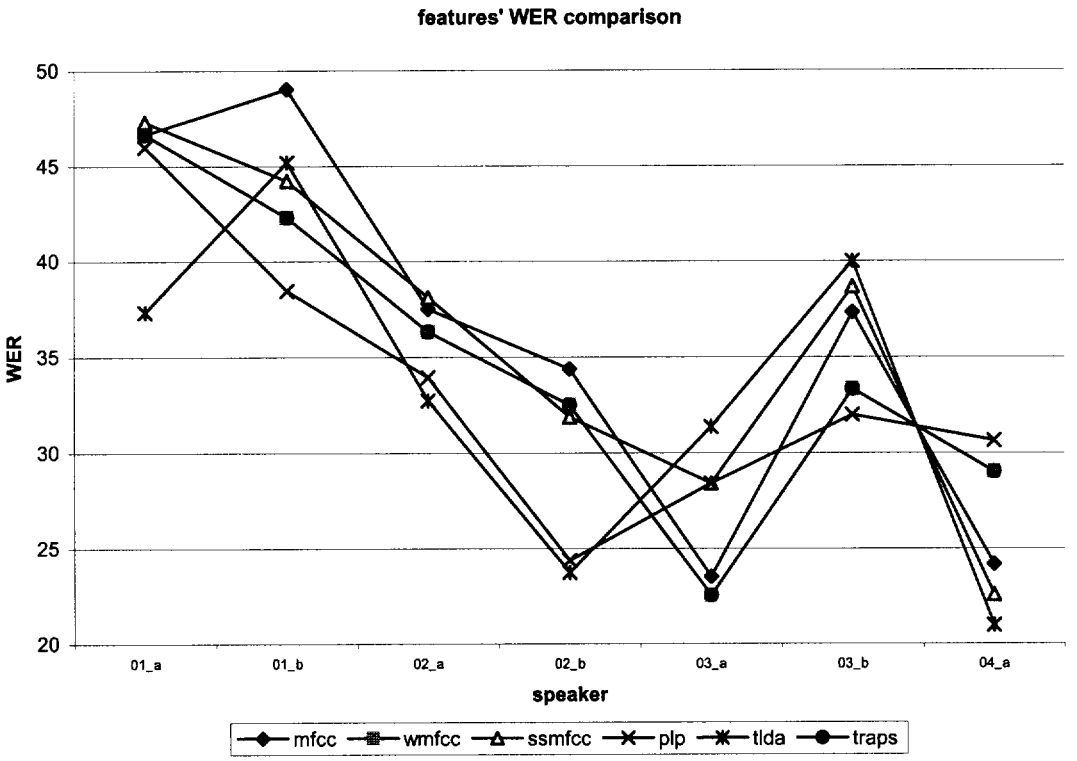


Figure 3.5: Performance comparison of different features

The procedure for obtaining 2-dimensional LDA-derived discriminants in our SPINE work is detailed as follows [66],[48],[91]:

Spectral bases representing the 15 most dominant LDA discriminants of logarithmic spectral energy vectors from 15 Bark filter-bank were derived on force-aligned SPINE1

training database with 56 context-independent phone classes. Temporal discriminants (LDA-RASTA filters) are derived from the OGI Stories hand-labeled database (41 context-independent phones). 15-dimensional logarithmic spectral energy vectors derived on this database are projected on the first spectral base. The three most dominant discriminants from LDA projection of 101-point temporal vectors (representing critical-band spectral energy trajectories, spanning 500 msec into past and 500 msec into future) yield three temporal RASTA filters. Features for the recognizer are generated as follows. The base features are first projected on 13 spectral bases to generate a 13 dimensional feature vector. The time trajectories of these 13 features are then filtered using three RASTA filters to generate (13x3=39) dimensional feature vector. Finally, each element of the feature vector is normalized using its mean and variance computed over the utterance.

Other Features Involved in the Building of Baseline System

There are a few other features that we used during our official evaluation [66],[91]:

1. FeatureNet [80, 27, 40]:

   The initial feature representation is 13 PLP cepstral coefficients (20 ms analysis frame with 10 ms steps, 12th order PLP model) normalized to zero mean and unity variance over the utterance. These features are fed into a 3 layer MLP, with 9 frames of context and 56 phoneme target classes, which results in MLP size 351-1000-56. The MLP is trained on forced aligned SPINE1 development data by back-propagation with minimum cross entropy criterion. The outputs of the trained MLP represent estimates of the phoneme posterior probabilities. These posterior probabilities have a skewed distribution that is difficult to be modeled by Gaussian mixture models. To circumvent the skewed distribution of the posterior probabilities, the final soft-max nonlinearity in the output layer is removed from the trained MLP by the logarithm of the posteriors with a normalization constant. The linear outputs

from MLP are augmented with their first derivative and second derivatives computed over 9 frame window. These features are whitened using global Karhunen-Loeve transform and the first 56 dimensions are retained.

2. SMFCC [66],[91]:

   The minimal value of power spectrum in each Mel-frequency band is used as an estimate of noise in the utterance. This estimate is subtracted from the Mel spectrum prior to cosine transform. The rationale is that the background noise is additive in the power spectral domain. We noticed that this feature by itself has no advantage compared with mean subtracted MFCC. However this feature set contributed to improvement performance of the overall (ROVER-combined) system.

3. WMFCC:

   WMFCC is similar to MFCC but with wide bandwidth filters [82]. The filter overlap is increased to 75% instead of 50% in the conventional MFCC. The average noise distortion is lower for WMFCC than for MFCC, and thus provides more noise robustness.

3.3 Building the Baseline System

3.3.1 OGI LVCSR System

Since 1996, our research group has been actively focused on research and development of large vocabulary continuous speech recognition systems and we have participated in several government sponsored annual evaluations (Broadcast News Transcription (HUB4) 1997 [92] and 1998 [90], Speech In Noise Environment (SPINE) [91], and Language Recognition 2003).

Our research software platform is a large-vocabulary, speaker-independent, continuous speech recognition system. It contains most of the state-of-the-art components of a speech recognition system:

1. Continuous HMM based training and decoding components.
2. A statistical n-gram language model, supports unigram, bigram, trigram, and class-based language models.
3. Complete package for signal processing and feature extraction (the commonly used features such as MFCC, PLP, LPC).
4. Some noise/channel variation reduction techniques, such as Cepstral Mean/Variance Normalization [30].
5. Speaker and channel segmentation.
6. Advanced acoustic model training and speaker adaptation:
   (a) Speaker Adaptive Training (SAT) [3] [2] [4],
   (b) Vocal Tract Length Normalization (VTLN) [26]).
   (c) Maximum A Posteriori (MAP) [97],
   (d) Maximum Likelihood Linear Regression (MLLR) [52]
   (e) Markov Random Field Linear Regression (MRFLR) [89].
7. Flexible decoders:
   (a) Supports either single-pass decoding or two-pass decoding.
   (b) Supports both within-word and cross-word model.

   Real time decoding is made possible with various fast decoding strategies including complex beam pruning, fast gaussian mixture computation, phoneme lookahead and language model lookahead.
8. A portable system that can run on either Linux or Windows platform, that supports parallel training and decoding on either a Linux or Windows cluster.

3.3.2 Acoustic Training

The initial Speaker Independent (SI) model was trained based on SPINE1 and SPIEN2 training data plus SPINE1 evaluation data. We refer to them as the SPINE2-SI-Base training set. The total training data is about 20 hours (Table 3.3).

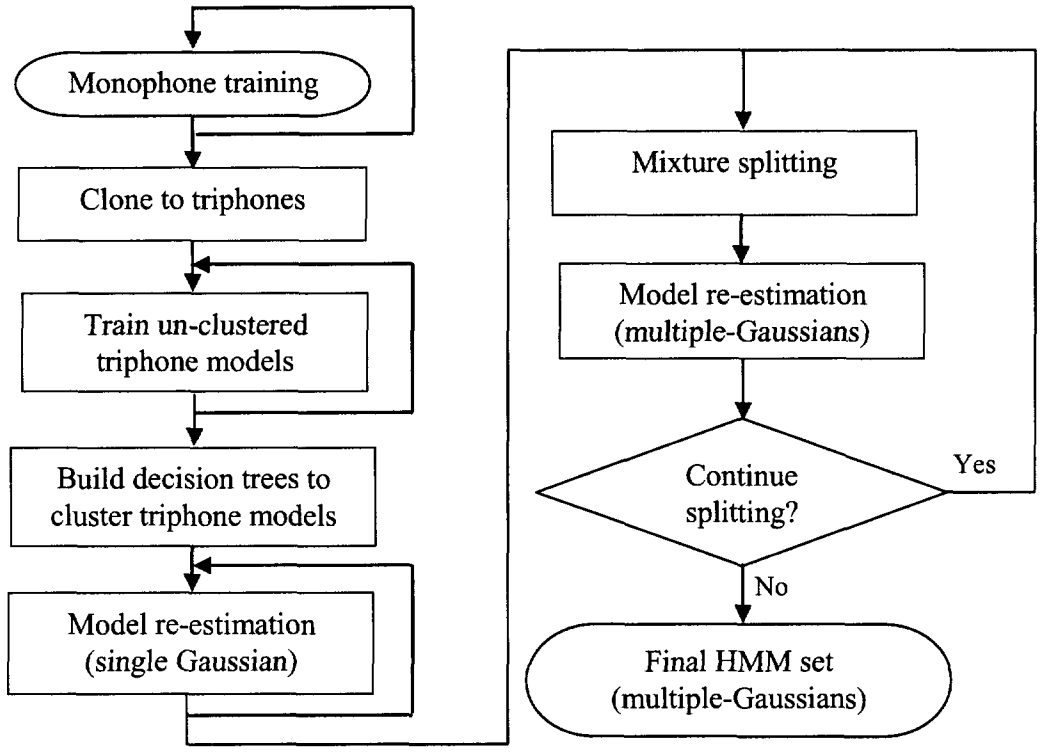


Figure 3.6: The acoustic training procedure

The training procedure is a step by step learning process, starting from building simple acoustic models to gradually increased model complexity. The training procedure is shown in Figure 3.6 and works as follows:

1. Monophone initialization and training (Bootstrapping)

   The first step of acoustic model training is building monophone models. Phonetically labelled speech data is necessary for this step. Generally a carefully hand

labelled database is essential to obtaining accurate monophone models. In English, the TIMIT corpus (Section 3.1.1) is the most widely used one for bootstrapping. Several iterations of Viterbi and Baum-Welch training are performed to obtain the probabilities of the acoustic observations given the current HMM models. If the change of the probabilities between successive iterations falls below a preset threshold, or the number of iterations reaches a preset limit, the procedure stops and a new set of HMM models are obtained.

2. Un-clustered context triphone training

   The monophone models are cloned to their corresponding context triphones (with same central phone). These triphones that are derived from the same monophone have identical HMM models. In this training step, several iterations of the embedded training are performed on the SPINE2-SI-Base training set. Each triphone model is updated after each iteration. Compared to the bootstrapping training, no phone or word boundary information is required. Instead, triphone HMM models are concatenated into a sentence level HMM model and the forward-backward algorithm automatically makes the time alignments. After this step, statistical information is collected for the next clustering step.

3. Clustering and state tying using a phonetic decision tree

   In this step, phonetically similar triphone models are clustered for several reasons:

   (a) A large number of triphone models need to be trained.

   (b) Limited training data.

   (c) The triphone models are not evenly covered by the training data.

   Based on the occupancy statistics of triphone models, a phonetic decision is constructed for every HMM state. Then the phonetic decision-tree algorithm is used to

Table 3.4: Comparison on acoustic model state numbers among different features

Number of states	MFCC	WMFCC	FeatureNet	TLDA	TRAPS	SMFCC
Within-Word	1221	1249	1422	1293	1284	1101
Cross-Word	1894	1837	1870	1879	1843	1832

We built both within-word and cross-word acoustic models for each features. The acoustic models have different HMM state numbers

perform state clustering and then state-tieing to the corresponding mixture parameters (mixture weight, mean vector and covariance matrix). Up to this step there is only one Gaussian mixture component for each HMM state.

4. Clustered triphone training

   The clustered triphone model is again trained with the embedded training algorithm on the SPINE2-SI-Base training set. After every three or four iterations of training, each HMM state in the model is split to more mixture components. Generally the amount of training data affects how many iterations of training and mixture components are necessary. In practice, experiments varying these numbers are performed and the best ones are selected by running a decoder on the development set.

The final clustered HMM states are automatically determined by the decision tree algorithm according to both the training data and decision tree parameters. The features we used in SPINE2 task have different state numbers, ranging from 1101 to 1422 for within-word models (table 3.4). The size of these acoustic models also varies because of that (table 3.5). These acoustic models all use 12 Gaussian mixture components.

3.3.3 Retrain Strategy

Retraining is an approach similar to unsupervised adaptation, except that the acoustic models are re-estimated (as in the normal acoustic model training) using the standard EM algorithm. Similar to unsupervised adaptation, the test data is first decoded and the recognition result was used as the training transcription. Rather than modify the SI models

Table 3.5: Comparison on acoustic model size among different features

Mbyte	MFCC	WMFCC	FeatureNet	TLDA	TRAPS	SMFCC
Within-Word	9.4	9.6	15.3	9.6	10.2	8.5
Cross-Word	14.6	14.2	20	13.6	14.3	14.2

We built both within-word and cross-word acoustic models for each features. Each acoustic model is stored as a binary file on the disk. This table shows the size of these binary files. As shown in the table, the acoustic model size varies from 8.5M to 15.3M for within-word models and from 13.6M to 20M for cross-word models.

by adapting them to the test data, our retraining approach combines the test data with the original training data for a fresh training. A diagram showing the retraining procedure is given in Figure 3.7.

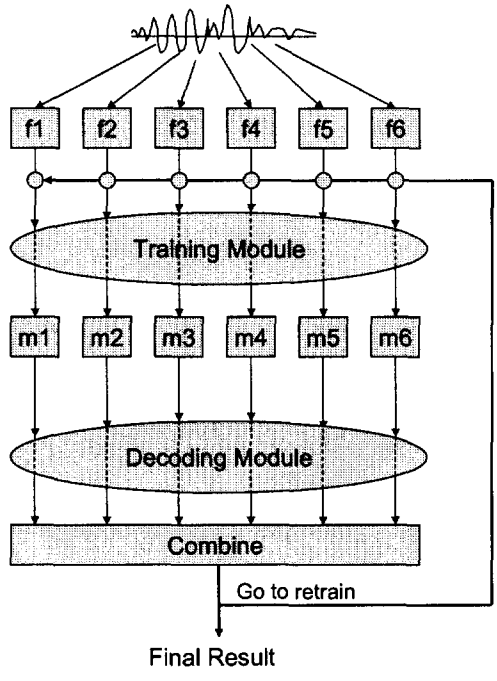


Figure 3.7: The retrain procedure in SPINE2

3.3.4 Lexicon and Language Model

In our official evaluation, we used the common language model provided by CMU to all participant sites.

The SPINE2 lexicon was built from three sources:

- 1759 unique words from all text files of SPINE1 data set but excludes partial words that occurred just once.
- 160 words containing ACE grid labels from the battleship game.
- 5000 most frequently occurring words in the Switchboard corpus. Tests conducted during the construction of SPINE1 language model showed that these were the most useful set of "extra" words to be included in the SPINE vocabulary.

After combining all files mentioned above, we obtained a 5720 word lexicon.

3.3.5 Experimental Results

The development of our SPINE2 system is based on many experiments on both SPINE1 evaluation data and SPINE2 dry run data. We first developed a system according to the official SPINE1 evaluation requirement, the purpose is to build a basic working system and compare its results with other sites. The official evaluation results on SPINE1 are shown in Table 3.6:

We built our rudimentary system and its result is the 2^{nd} best (32.3%) compared with all the participant sites. We further did comparison experiments on MLLR and Retraining. Our retrained system is better than the one which we applied MLLR (Table 3.7).

From Table 3.6, we can see that the best overall system used ROVER [28]. Inspired by this finding, we built 15 different systems, varying their features. The final result after applying ROVER is quite impressive: ***25.0%***.

Based on this setup, we trained a similar system for SPINE2 dry run. The training data includes all SPINE1 data plus SPINE2 training data. The testing data is the SPINE2 dry

Table 3.6: Official Evaluation Results on SPINE1

Systems	WER
Site1 ROVER	25.7%
Site1 Primary	26.5%
Site2	32.8%
Site3	40.8%
Site4	46.3%
Site5	55.1%
Site6	56.2%
Site2	58.1%
Site2	68.7%

Table 3.7: Experimental Results on SPINE1 Evaluation Data: Comparison of MLLR and Retrain on MFCC based systems

Systems	WER
Baseline	32.3%
MLLR	29.6%
Retrain	28.8%

run data. Based on our findings on SPINE1, we used Retrain exclusively in the following experiments (Tabel 3.8). The additional findings are that MLLR is complementary to Retrain, and VTN delivers significant gain. The final system reduces the WER by 27% compared to the baseline.

Table 3.8: Experimental Results on SPINE2 Dry Run Data

Systems	WER
Baseline	44.6%
Retrain Round 1	35.0%
Retrain Round 2	34.5%
Retrain Round 2 + MLLR	34.1%
Retrain Round 2 + MLLR + VTN	32.5%

Based on these experiments, we built our final system for the official evaluation. The features and acoustic model training have been introduced since the beginning of this

chapter. The original plan also includes two rounds of Retrain plus MLLR and VTN. We did not finish that part because of the tight time schedule. The results in the official evaluation are shown in Table 3.9.

Table 3.9: Baseline System Performance on Official SPINE2 Evaluation

Mbyte	MFCC	WMFCC	Feature_Net	TLDA	TRAPS	SMFCC
Baseline	46.7%	47.6%	54.6%	50.6%	55.0%	48.9%
ROVER	41.5%					
Retrain	42.1%	NA	NA	NA	NA	NA
ROVER	40.7%					

Compared to other participant sites, our system ranks 3^{rd} place in using the common language model (Table 3.10).

Table 3.10: Official SPINE2 Evaluation Result: common language model

System name	%totalerror	%correct	%SUB	%DEL	%INS
ATR	72.5	51.2	39.2	09.6	23.8
ATT	42.7	63.7	26.3	10.0	06.4
CMU	38.1	69.1	23.9	06.9	07.2
CU	42.6	63.7	27.4	08.9	06.4
IBM	53.7	52.2	30.5	17.3	05.9
OGI LVCSR	40.7	62.2	14.5	23.3	02.9
SRI	42.1	68.2	23.7	08.1	10.3
SSLI	38.8	66.6	22.6	10.9	05.4
ISIP	56.9	54.6	28.9	16.5	11.5

After the official evaluation, we finished up some of the planned system building and the best result 39.6% was treated as our baseline (Table 3.11).

3.3.6 Some Improvements by Applying Class Based Language Model

In this thesis work, some effort are devoted to improving the performance of our baseline system. In the official SPINE2 evaluation, some sites obtained significant gain by using

Table 3.11: Baseline System Performance on Official SPINE2 Evaluation - Post Evaluation

Mbyte	MFCC	WMFCC	Feature_Net	TLDA	TRAPS	SMFCC
Baseline	46.7%	47.6%	54.6%	50.6%	55.0%	48.9%
ROVER	41.5%					
Retrain	42.1%	42.6%	44.2%	43.6%	45.9%	45.1%
ROVER	39.8%					
Retrain	41.8%	42.2%	43.5%	42.4%	45.4%	43.6%
ROVER	39.6%					

Table 3.12: Official SPINE2 Evaluation Result: special language model

System name	%totalerror	%correct	%SUB	%DEL	%INS
CU	37.5	67.8	24.2	08.0	05.3
IBM	29.3	73.0	13.7	13.3	02.3
SRI	27.5	74.4	13.5	12.1	02.0

special language models, such as class based language models. Because of their success, we implemented a class based language model in our baseline system.

It is usually difficult to obtain enough training corpus for many domain-dependent tasks, such as, the SPINE2 tasks. In such cases, a class based language model is preferred to a general word-based language model. SPINE2 task contains some common words, such as personal names, military acronyms, slang and directions, etc. Each individual word cannot be well trained since each one occurs infrequently in our sparse corpus. In this case, a class-based language model can be employed to gain a more robust model and further reduce the model size.

Before training a class-based language model, we first defined some classes (refer to Appendix A). For example, the DIRECTION class includes words such as east, west, northeast, etc. A tagged sentence from our corpus is shown below:

Original sentence:

East to southeast do you copy Michael.

Tagged sentence:

[DIRECTION: East] to [DIRECTION: southeast] do you copy [Name: Michael].

Given two neighboring words W_{i-1} and W_i, the probability of the word W_i given its preceding word W_{i-1} can be expressed as:

$$P(W_i|W_{i-1}) = P(W_i|C_i)P(C_i|C_{i-1}) \tag{3.2}$$

where W_i and W_{i-1} belong to class C_i and C_{i-1} separately.

To obtain a class based language model from our original trigram model, two additional sets of probabilities need to be estimated:

1. Transition probability $P(C_i|C_{i-2}, C_{i-1})$, which is the probability of the current class C_i given its preceding two classes C_{i-2} and C_{i-1}.

2. Observation probability $P(W_i|C_i)$, which is the probability of the current word W_i given its class C_i.

We use the CMU-Cambridge Statistical Language Modeling Toolkit to obtain the set of transition probabilities. Also, the set of observation probabilities is computed via dividing the number of occurrences of each word in a class by the total number of occurrences of all words in the class. In our computation, a Witten-Bell discounting method is applied to the probabilities of less frequent or unseen events. The improvement after adopting class based language model is impressive (Table 3.13).

Table 3.13: Comparison on the effect of Class based Language Model (CLM) and common language model

Systems	WER
Baseline	39.6%
Baseline + Retrain + MLLR + VTN with common language model	35.8%
Baseline + Retrain + MLLR + VTN with CLM	31.2%

Similar to acoustic model retraining, we retrain the language model by interpolating original training data with decoded testing data. Decoded testing data contains some unseen word sequences in the original language model. And by retraining, the resulting

language model is closer to the testing domain. Some confidence measurement steps were used to remove some error prone utterances from the decoded data [100], [99]. Further improvement was observed after applying the approaches above (Table 3.14).

Table 3.14: Comparison on the effect of language model retrain

Systems	WER
class based language model	31.2%
class based language model + retrain	28.7%

Chapter 4

Overview on Information Fusion in Speech Recognition

This chapter gives a background review on some existing approaches to performing fusion in speech recognition. Information fusion in speech recognition is a relatively new and active research area. It is based on the research findings on human speech recognition (Section 4.1). Current fusion approaches can be roughly classified into two categories: pre-recognition (Section 4.2) and post-recognition combination (Section 4.3). We will review these existing fusion approaches and give their advantages and disadvantages.

4.1 Information Fusion in Speech Recognition

In human speech recognition, various cues (including visual information) are used. The more difficult the speech (such as in noisy environments), the more cues are needed [24]. Fletcher extensively studied how humans process and recognize speech [1, 29]. This work showed that the phones are processed in independent articulation bands and that these independent estimates are "optimally" merged to achieve the recognition results. Recent research activities on multi-stream or multi-band [11, 42, 61] also demonstrated the importance of looking at the data from different angles (different signal processing and features) and fusing the information to improve recognition accuracy. However, both Fletcher and the recent activities did not explicitly conclude **how** different information

should be fused to form the sound-unit recognition in order to achieve human-like performance.

Motivated by how humans recognize speech, in recent years, there has been a strong interest among researchers on how to combine different features in speech recognition [21, 24]. The success of these approaches is partly due to their efficiency in improving recognition accuracy, partly due to their simplicity and ease of deployment.

The existing art can be roughly classified into two categories: pre-recognition and post-recognition combination.

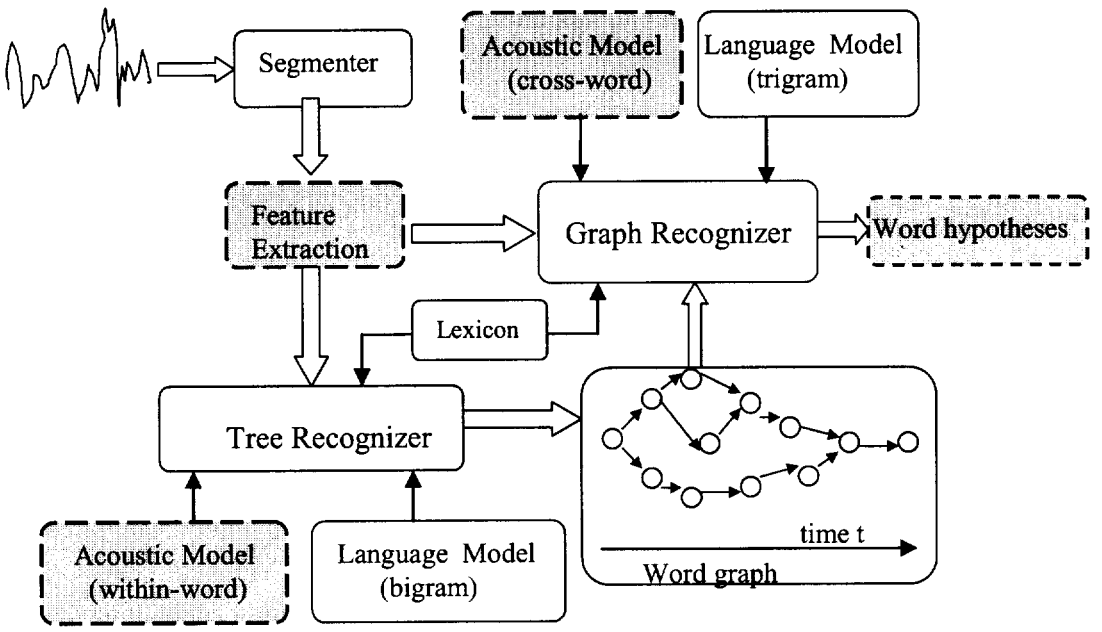


Figure 4.1: Existing fusion approaches in an ASR system

The boxes with dashed lines are the modules related to existing fusion approaches.

4.2 Pre-recognition Combination

The pre-recognition approach combines features or probabilities before conducting decoding. It can be further classified into feature combination and probability combination.

4.2.1 Feature Combination

Feature combination concatenates different features to form a single feature vector before acoustic modeling. The benefit of this approach is that the time dependence of different features is exploited. Successful examples of this approach include concatenating energy and delta features with a spectral representation (such as MFCC [31] as shown in Figure 3.3).

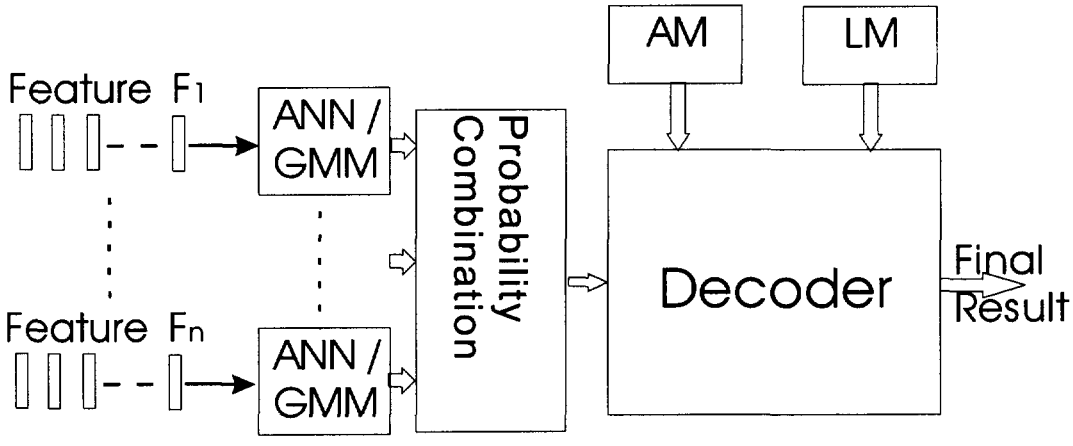


Figure 4.2: Pre-recognition: Feature Combination

The advantage of feature combination is that it is easy to deploy in current HMM systems. The disadvantage is that the combination of several features could lead to a much larger feature vector, and therefore a larger acoustic model. The larger acoustic model requires much more training data, resources and time. Feature combination also incurs many redundancies thus diluting the complementary information. Another disadvantage is that feature combination assumes that the combined features are independent. However,

most existing features have highly correlated information. Thus feature combination requires special selection of individual features. The success of feature combination is still limited to concatenating the energy and delta vector with the baseline feature vector. During our preparation for the SPINE task, we concatenated TRAPS and MFCC feature vectors into a single feature vector. Although the feature space was nearly doubled, the recognition performance was somewhere between the two single feature systems.

4.2.2 Probability Combination

Probability combination is mainly used in HMM/ANN (Artificial Neural Network) hybrid systems such as Multi-Band [11, 61, 18, 17] and Multi-Stream systems [13, 83, 47, 61, 21]. A set of ANNs are trained for each feature and used for probability estimation. The output of these ANNs are combined and input to an HMM decoder.

An example of probability combination is the Multi-Stream approach, which is mainly based on an HMM/MLP (Multi-Layer Perceptron, a kind of ANN) hybrid system that employs several MLP recognizers trained on different features. The outputs of these MLPs are fed into another trained MLP to estimate the phone posterior probability (Figure 4.3). The combined phone posterior probability is input into an HMM decoder for final decoding. Experiments have reported that Multi-Stream systems have better noise robustness than the conventional HMM/MLP hybrid system using single feature.

A variation named a "tandem approach" [40, 27] has been proposed recently. It is different from the conventional Multi-Stream approach in that the outputs of MLPs are used as input features of GMMs (Gaussian Mixture Models) instead of a MLP. The output of the GMMs are likelihood values for different speech units used in the HMM decoder. Tandem approach has reported large WER reduction using context-independent modeling compared to the standard MFCC or PLP features. Nevertheless the improvement cannot carry over to large vocabulary task such as SPINE when context-dependent modeling is necessary [27].

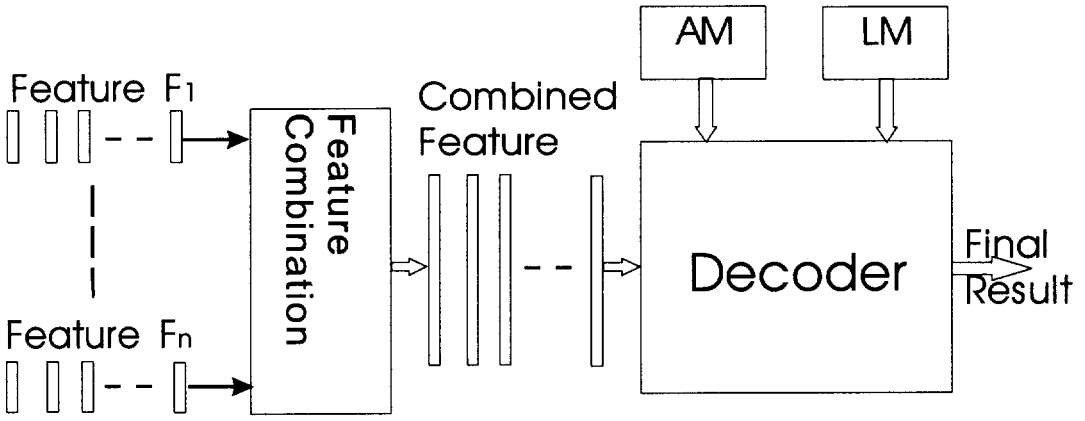


Figure 4.3: Pre-recognition: Probability Combination

The advantage of probability combination is that it can be designed for parallel processing in several small models instead of a single large one. The disadvantage is that the number of ANNs needed to be trained is very large and often prohibitive for a context-dependent phone system. The drawback for both approaches is that only frame-based feature can be incorporated (or only time synchronized features can be incorporated). Segmental based information, such as tones (or pitch patterns), cannot be integrated easily.

4.2.3 HMM Combination

There has been much research in exploring possible extensions to HMMs. These include factorial HMMs [34], 2-D HMM [62] and coupled HMMs [15, 65] among others.

Factorial HMM was first introduced by Ghahramani and Jordan. They attempted to extend HMMs by allowing the modeling of several stochastic random processes loosely coupled. Factorial HMMs can be seen as an extension to HMMs. In the experiments presented in their report, factorial HMMs did not appear to offer any advantage over regular HMMs when traditional feature vectors were used.

To integrate segmental based information, such as tones (or pitch patterns), Mirghafori

and Morgan tried to relax the synchrony constraints in their research by using a 2-D HMM [62]. In their research, sub-band HMMs are combined to form 2-D HMMs. Two approaches were used in their paper to relax synchrony constraints: HMM decomposition/recombination and two-level dynamic programming.

Nock and Young proposed a method called loosely coupled HMMs (Figure 4.4), which is similar to the factorial HMMs. In their approach, two HMMs are coupled together to form a so-called loosely coupled HMM. A coupling matrix is defined for each coupled HMM, which represents the transition and observation probabilities. Coupled HMMs are trained to model two different input streams with asynchrony allowed. Various degrees of synchrony between the two state sequences are also allowed by restricting some state transitions. One major problem for this approach is the computational cost is $O(S^3T)$, instead of $O(S^2T)$ for traditional HMMs. The number of parameters that need to be estimated is also explosive in space under a continuous ASR system.

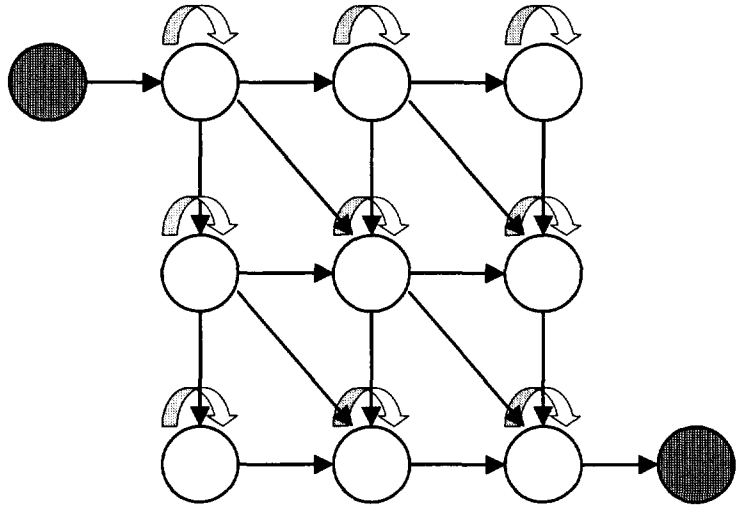


Figure 4.4: Coupled-HMM topology

Although theoretically 2-D HMM and coupled-HMM can be used to address the existing problem, the associated expense is an increased model space (extra states need to

be introduced). Although it is straightforward from an implementation point of view, the tremendous increase in the state space dimension makes it impossible for applying to multiple input streams. Attempts were made in [61] but failed to improve accuracy due to significant increase in free parameters that needed to be estimated.

4.3 Post-recognition Combination

For the post-recognition combination approach, the underlying mathematic assumption is the conditional independence between different features during recognition of each stream. Thus decoding is performed on each stream independently of the decoding on the other features. The benefit of this approach is its simplicity and flexibility in manipulating the final recognition result. Approaches such as ROVER and word graph (or lattice) combination all fit into this category. The time-dependency between different features is completely ignored during the recognition of each stream. There is no interaction between different features during the decoding process, and so the presence of one feature will not affect the course of decoding on the other features. The problem with this approach is that some complementary information among different features is not utilized. The mistakes made in the early decoding stage may not be recoverable at the combination stage since the correct hypothesis may have been pruned away during decoding of each individual streams. As shown in Figure 3-3, recognition is performed independently on each single feature representation and the results are combined in a post recognition manner.

4.3.1 Recognizer Output Voting Error Reduction (ROVER)

ROVER was introduced by J. Fiscus at National Institute of Science and Technology (NIST) and used at the DARPA 1997 LVCSR Hub 5-E evaluation [28]. After combining the results submitted by all participants in the evaluation, the WER is reduced to 39.4% from 44.9% (obtained by the best single system). Since then, ROVER has gained much attention in the speech recognition community. Five of nine participants in the 1998

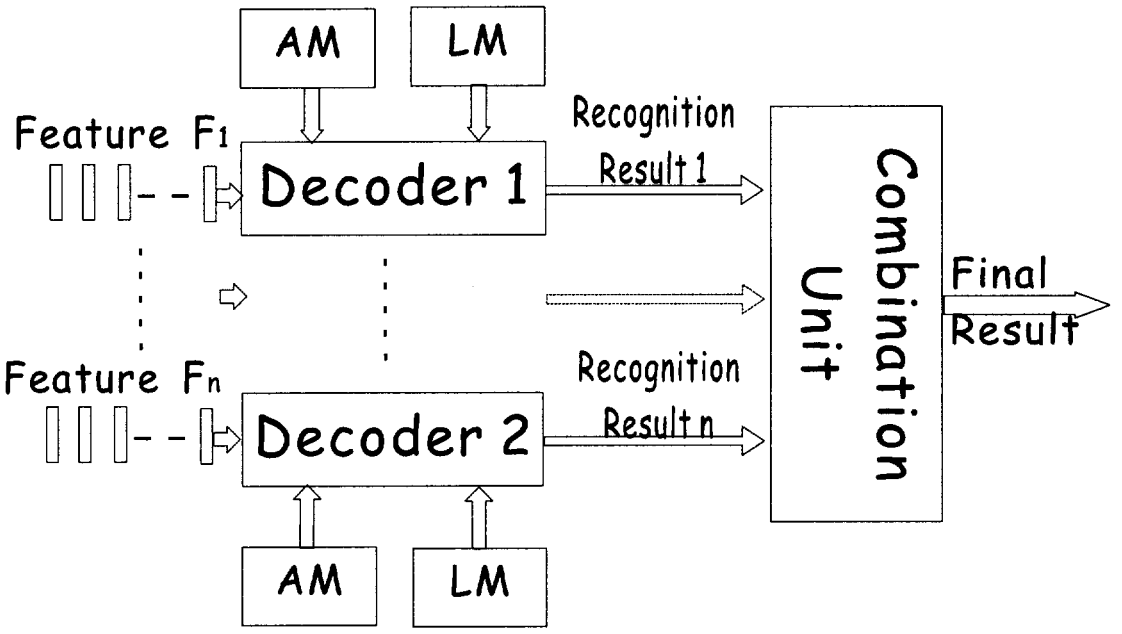


Figure 4.5: Post-recognition: recognition result combination

DARPA Broadcast News evaluation adopted ROVER. Even though their results are the output of a ROVER system, NIST further reduced the WER from 13.5% to 10.6% after performing ROVER on the results of all nine participating systems.

During our development on SPINE1, when we combined outputs from systems with 15 different feature front ends using ROVER, the combined system obtained a WER of 25%; although systems with each individual feature front end had a WER ranging from 32% to 50%.

ROVER is based on the hypothesis that the complementary information from different recognizers output can be used to reduce word error rate. It has two steps:

1. Align outputs of all the recognizers and build a single Word Transcription Network (WTN) by dynamic programming (DP).

2. Select the highest voted word as the best scoring word at each node of the WTN.

The WTN is aligned iteratively in step one by first aligning two output sequences to

form a combined WTN. This WTN is aligned with the third output word sequence, then the fourth and so on. So the final WTN is related to the combination order of all the recognizer outputs. To achieve best ROVER results, the recognizer outputs are ranked by their individual word error rate in an ascending order. However, the word error rates of the individual systems are generally unavailable.

During the second step, ROVER picks the word with the highest number of votes at each node in the WTN. When two or more words have a tie in the voting, the tie is arbitrarily broken, which is a major drawback for this voting scheme. Especially for a ROVER with only two inputs, all the potential correctable words result in a tie and the final ROVER output lies between the original two input systems. So when only two recognizer outputs are available, ROVER has no advantage at all in reducing WER.

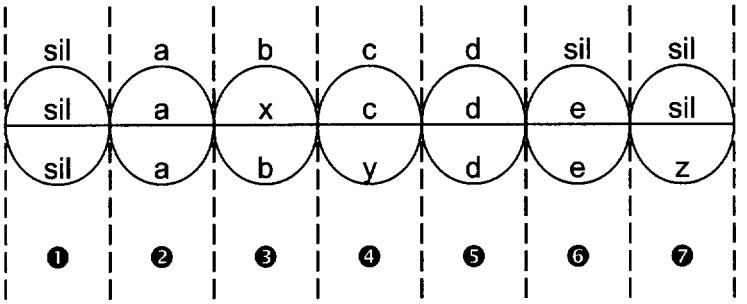


Figure 4.6: A WTN of a ROVER system with three hypotheses as input

There are seven aligned regions in this WTN. Each small capital letter represents a hypothesized word. 'sil' represent a silence region.

In summary, ROVER has the following advantages and disadvantages:

Advantages:

1. Based on a solid assumption that the error patterns of two systems can be dramatically different even though they have a similar recognition error rate.
2. Requires very little run time itself.
3. Works quite well in most cases.

Disadvantages:

1. Based on an unreliable voting decision: The word confidence scores from different systems are not strictly comparable but ROVER assumes they are.

2. Has a limitation in real application: The time and resource cost are linearly increased by a factor of the number of systems.

3. Only uses the first hypothesis (best hypothesis with the largest likelihood) of each system. Complementary information contained in the hypotheses beyond the first one is discarded. Potentially better paths have been pruned during the recognition phrase and are not recoverable.

4. Performance is influenced by the order of the combination with the best system ranked the first. This assumes a prior knowledge about the performance ranking, which is not always available.

5. Cannot guarantee performance improvement. Experiments found it actually hurts performance when combined with systems that have higher word error rates.

4.3.2 Hypotheses Combination

During the 2000 SPINE evaluation, the best system from CMU introduced a parallel hypotheses combination method [82]. The word hypotheses obtained from parallel systems are combined into a word graph. Unlike ROVER, the acoustic score is carried with each node of the word graph. The nodes representing identical words between the same time instants are merged into a single node. For each node pair, if the end time of first node is within 30 ms of the begin time of the second node, a link is added between these two nodes. Finally, a language model is used to score the word graph and find the best path as the final hypothesis.

This approach is similar to ROVER at aligning and building the WTN, but the difference is that it tries to explore more paths than the first hypothesis, so the resulting WTN is

an extended network compared with ROVER. The WER of the best path that can be found in the WTN is believed to be lower than that from ROVER. The problem lies in whether the re-scoring by a language model can select better paths. CMU's result [82] shows that it improves performance compared with the baseline system, but no comparison is done with ROVER.

Since acoustic scores from different systems (recognizers) are not readily comparable, the hypotheses from different systems are infeasible to be cross-linked, which limits the combination only between the same recognizer with different models or features. Because hypothesis combination is the same as ROVER on only combining the single best hypothesis from each systems, it has the same disadvantages as ROVER has.

Chapter 5

Run Time Fusion in Speech Recognition

This chapter presents how we approach the problems in the existing fusion approaches, rooted in their inefficient use of the complementary information. We propose a run time fusion framework to address these problems (Section 5.2). In this thesis work, we mainly discuss our approach at the acoustic level and we fuse the complementary information from the multiple features (Section 5.3). Starting from Chapter 6, we present the detailed design and implementation of our approach. Under the general high level fusion framework, we designed three different fusion approaches. These three approaches are based on the same hypothesis, that by applying complementary information at an earlier stage of the recognition process, the final system will be able to obtain much better accuracy. These three approaches differ from each other at when, where and how the fusion is performed. We investigated these three approaches (or system architectures) in the hope of making the best use of multi-information sources. Experimental results are given after each approach to demonstrate the advantages of our solutions.

5.1 Problems and Motivations

As described in Chapter 4, current approaches to fuse different features have limitations.

1. From the accuracy point of view, pre- and post-recognition do not use the potential benefit of a recognition engine (decoder). Pre-recognition approaches use multiple

information at the feature or acoustic probability level; post-recognition approaches use multiple information at the reduced sentence level. Compared to pre- or post-recognition approaches, our run time fusion has access to all levels of complementary information in the full extent (Figure 5.1). There has not been any work done to perform run-time fusion inside a decoder. A possible reason why this hasn't been done before is the recognizer engines are not readily available to the public till recently, and they are also rather complex to be manipulated. Current pre- and post-recognition undermine the possible improvement on recognition accuracy when complementary information is used.

Our hypothesis: Much more complementary information is available and can be better used during run time in a unified framework. More performance gain can be obtained by performing information fusion at the decoder's run-time compared to the pre- or post-recognition. Run-time fusion is also more robust to different features and noises.

2. From the computation cost point of view, post-recognition approaches, such as ROVER and Hypotheses Combination, require separate recognitions to be performed thus the computation cost is linearly increased. Running separate recognizers not only increases the computation cost but also increases the demand of resources. Since maintaining several recognizers is generally unaffordable, the recognitions are usually performed using the same recognition engine.

   Our hypothesis: Rather than running the same recognizer repeatedly, performing information fusion in a single recognition engine is a feasible and efficient approach. During-recognition fusion can achieve improvement without a large increase in computation cost. Better or comparable performance can be achieved when running the recognizer in a speedup mode (such as narrowing the beam width).

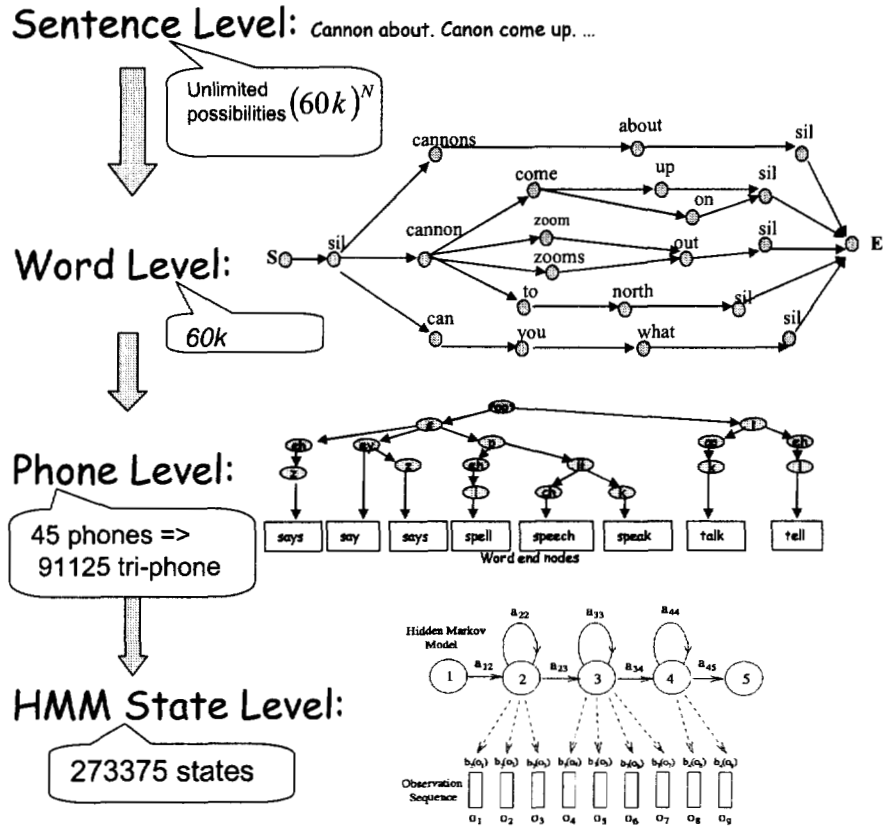


Figure 5.1: Different levels of information within a speech recognition

5.2 Framework of Run Time Information Fusion

We investigated how to effectively fuse different information sources during run-time. The framework for our proposed fusion work is illustrated in Figure 5.2. The novel part of our proposed work is the interaction between different feature streams during recognition. The concept is similar to RAID for storage: using a collection of identical and inexpensive components to form an efficient and better system. The advantages of our proposed work on information fusion include:

Compared with post recognition fusion, complementary information among different feature representations will be exploited during search to avoid un-recoverable errors in post recognition processing and the dependence (or time correlation) of different feature

streams will be preserved. More information can ensure that the recognizer makes fewer errors at run time, and the improvement of recognition on each feature stream will contribute to the over all fused recognition performance.

Compared with a pre-recognition approach, the constraint on frame level synchronization is relaxed in our proposed work and it enables features with different time/frequency resolutions and time spans (such as segmental based features) to be readily integrated. It should be mentioned that our approach is not in opposition with the pre- or post-recognition, and they can coexist in the same system.

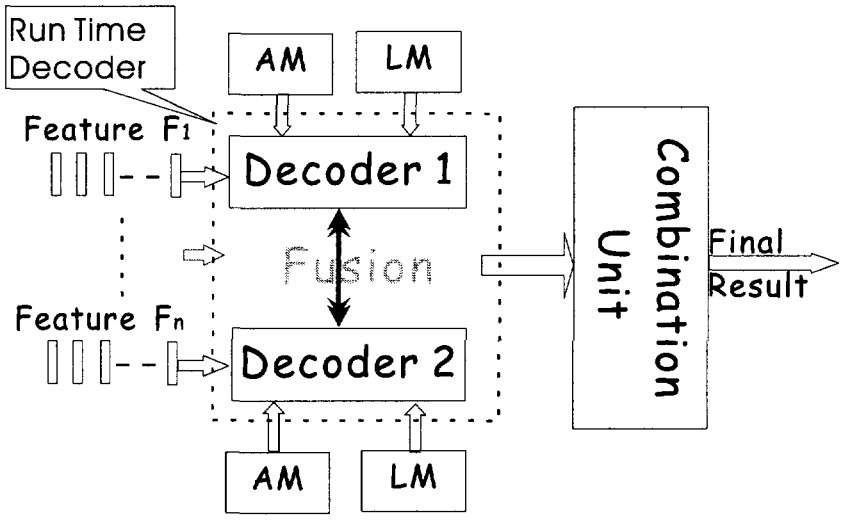


Figure 5.2: Run time fusion framework

The decoder with its underlying search network is the core of any speech recognition system. It can integrate multiple knowledge sources in the same framework at several levels (state, phone and word) and both within and across phones (or words). Rich intermediate information is available for manipulation at these levels. Pre-recognition fusion operates at the frame level and post-recognition is conducted at a reduced word level space. Our hypothesis is that recognition accuracy can be enhanced greatly by utilizing the complementary information contained in different features at different levels during decoding. In this thesis, we investigate fusion methods that lie in between two existing

extreme approaches (either strictly conditionally dependent or completely independent) and show it can provide a more reliable fusion.

The novel part of this framework is the interaction between different features *during* recognition. The potential benefit of this framework are:

1. Complementary information between different feature representations will be exploited during the search to avoid un-recoverable errors as in post recognition processing, and the dependency (or time correlation) of different features is preserved. More information can ensure the recognizer makes fewer errors at run time, and the improvement of recognition on each feature will contribute to the overall fused recognition performance.

2. The constraint on frame level synchronization is relaxed, and it enables features with different time/frequency resolutions and time spans (such as segmental based features) to be readily integrated.

3. Current LVCSR recognizers are quite complex and very few people in the world actually master their art. This is probably one of the main reason why previous work concentrated on ***pre-*** or ***post-*** recognition fusion but not ***during*** the recognition. The advantages of a recognizer make it more attractive to be the center of fusion:

   (a) Integration of multiple knowledge sources: Acoustic and language knowledge sources are already integrated in current LVCSR recognizer. Other innovative knowledge sources such as prosodic information, confidence measurement, noise cancellation, etc. can be used without much trouble.

   (b) Rich statistical information is readily available: As a by-product of decoding, a tremendous amount of data related to the search is produced during recognition. These data can be used to perform fusion under our current statistically based recognizer.

(c) The success of run time fusion can promote researchers in other specialities to experiment with more innovative approaches independent of the recognizer. Such approaches may have had a lack of success under traditional recognition frameworks, which prevented them from further studies.

(d) Our run time fusion approach is not limited to the acoustic level and can be easily extended to other system modules such as language models under a similar methodology. The success of our approach can also help finding complementary knowledge source pairs in a given task.

This thesis work will focus on conducting fusion:

- **When?** - During the decoding (Run time).
- **Where?** - Inside the recognizer.
- **What?** - Fuse the information from several knowledge sources.

This thesis work tries to answers this question: **How to** fuse the information from several knowledge sources inside the recognizer during run time.

5.3 Fusion Based on Multiple Features

Human auditory studies have found that much of the speech signal could be discarded without a significant impact on human's speech recognition process [37]. The nature of human speech contains much redundancy. Phonetic features are signaled by many different cues and distributed in both time and frequency space. This distribution is used to robustly transmit information contained within the speech signal. However, the underlying detail of the distributed representation is still unclear. Furthermore, we still don't have a complete structure representation for the speech signal. The different speech features are based on different assumptions on the structural representation of the speech signal, and they contain different information of the original speech.

For many years, researchers have been devoted to finding a "perfect" feature representation of the speech signal. It seems an endless journey so far, just as we cannot make a "perfect" recognizer that makes no mistakes. Current speech recognizers adopt a single "best" feature set according to the task it is facing and measure their result on a development data set. Also the feature representations have fixed parameters (such as the window and frame size, and the number and shape of the band filters) during feature extraction. For example, MFCC, one of the most popular feature representations, usually has a window size of 25ms with 10ms frame size.

However, different features or features with different parameters can represent the same speech input differently. Information loss is inherent for any feature extraction method because it is a compression process of original speech signal[14]. The remaining information is different for features that are based on different feature extraction methods. It is plausible these differences will result in different recognition results.

Our experiments show that different features have fluctuating performance on different speakers and noise environments. Figure 5.3 gives a clear illustration of the large performance difference among different features. Although the overall performances (WER) of most of these systems that using different features are similar with differences under a few percentages, there are significant differences between there output. The average difference between the highest and lowest WER across all features for each speaker-environment pair is about 20%. If we can successfully select the best-performing feature for every pair, the WER can be reduced significantly compared to the best single feature performance. There is much potential to be explored by using the complementary information from multiple sources (features). The experiments confirm that different feature extraction algorithms can exploit complementary information of the same acoustic signal. Furthermore, the complementary information is exhibited at the outputs of the recognizers.

In this thesis, we explore the use of complementary information within a large vocabulary continuous speech recognition system. Different features will be fused at different

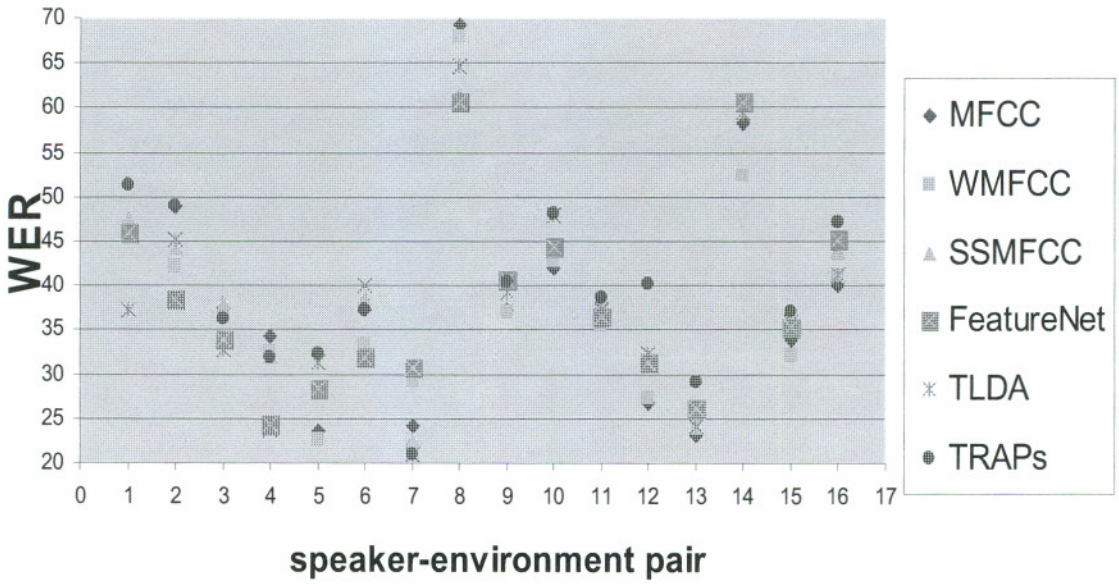


Figure 5.3: WER comparison on 16 speaker-environment pairs

Refer to Section 3.2 for information about each feature showed in this Figure.

levels during the run time of the recognizer.

Chapter 6

Run Time Fusion In Detail

The work described in this thesis has focused on increasing the robustness of the large vocabulary speech recognition. One major problem that speech recognition researchers have to deal with is the robustness issue. The variability of speech is the nature of human speech. Added to the difficulty is the channel and environmental distortions. This thesis work is intended to tackle the problems raised by performing run-time information fusion inside the decoder.

The decoder used in this work is similar to other state-of-the art decoders. The decoder is the most important component of a LVCSR system. It is responsible for integrating various knowledge sources in an efficient way to generate the best word sequence for a given speech signal. There are many requirements for a successful decoder. The basic performance measurement of a decoder is the recognition accuracy. A decoder can also be measured by the demand of computation resources such as memory and disk space. A successful decoder also needs to be fast and robust. Lots of effort has been spent on designing the decoder to cope with the ever increasing demand of a speech recognition system. Current decoders have a very complex design and implementation. The decoder used in this work uses many state-of-the-art technologies such as, time-synchronous beam pruning and lexical tree based Viterbi search. During the decoding, the decoder will generate and manage an extremely complex search space. Thus both the static structure of the decoder and the dynamic decoding space are very complex. Our run-time information fusion was designed to work with these complexities. To achieve this goal, lots of efforts

are related to the details of the implementation. Thus it is necessary to describe many implementation issues in the following sections.

The proposed approach was tested on a large vocabulary speech task which contains eight different environment noises. The experimental results were measured against the results obtained from the conventional approach using WER, the standard criterion used to measure the performance of a speech recognition system. However the WER does not exhibit the power of our run-time fusion. Because the language model of SPINE task restricts the further reduction on WER. And the language model is not the focus of this thesis. Although our run-time fusion can also utilize multiple information from different language models. This thesis is focused on improving the acoustic disambiguation ability. A better evaluation criterion is to decouple acoustic recognition from language model. Word graph provides such an option. When the acoustic ambiguity is higher, there will be more optional words occurred in the word graph. So in this thesis, we also used WGER (Word Graph Error Rate) to evaluate system performance.

Section 5.2 outlines the novel framework of our approach. The fusion is conducted at the run time (**when**) of the decoder (**where**). The statement above distinguishes our approach from other existing fusion solutions. However, it only provides a general architecture 5.2 of our approach. To make our approach work, we will investigate three approaches discussed in the following subsections. These approaches will give a more specified answer to these question:

- **How** to fuse the information from several knowledge sources at the run time of the decoder?
- **When** to perform the fusion?: At what specified time should we perform the run time fusion.
- **Where** to perform the fusion?: At what points of the decoder should we perform the fusion.

Under the above high level framework, we designed three different fusion approaches. We investigated these three approaches (or system architectures) in the hope of making the best use of multi-information sources. These three approaches are based on the same hypothesis, that by applying complementary information at an earlier stage of the recognition process, the final system will be able to obtain much better accuracy. These three approaches differ from each other at the solutions to the three above questions.

We refer to the first approach as *'constraint fusion'*. In constraint fusion, one feature serves as the main feature during the decoding. Decoding of the main feature sets a constraint on the search space for the other features. Other features keep independent search paths but mainly functioned as consultants to the main feature. We refer to the second approach as *'composite fusion'*. The composite fusion is different from the constraint fusion as each feature stream is independent and has its own search space. The synchrony happens at phone or word boundaries. We refer to the third approach as *'rank based fusion'*. Compared to constraint fusion, instead on directly using the log likelihood value from different feature streams, the rank based fusion uses the relative rankings of all the hypotheses.

The successfulness of our approaches is measured by the following:

1. The approach improves recognition accuracy.
2. The approach keeps recognition efficiency.
3. The approach is implementable.

To implement these three approaches, significant work was spent to modify the current decoder architecture. To clearly describe our approaches, and to answer the three questions above, this thesis has to highlight some important implementation details throughout the following sections.

Please note that although we will mainly discuss our approach at the acoustic level, it can be easily generalized to language models and other system components using a similar methodology.

6.1 Constraint Fusion

A post-recognition combination method, such as ROVER, lacks robustness when used with higher error systems; its performance degrads. Since ROVER is a voting procedure, a higher error hypothesis could play a crucial role in the final output decision when there is a tie (or near-tie) among hypotheses. Similar robustness problems exist in the pre-recognition approaches. Because fusion is performed in only one dimension, such as the feature or probability level, noise corrupted features make the same contribution as clean features. To overcome this kind of problem, we proposed a constraint fusion scheme to increase the robustness.

In our constrain fusion scheme, a set of models are trained independently for each feature. During the decoding, recognition on each feature runs independently, except at the designated boundaries (such as phone or word boundaries). The designated boundaries are decided by decoding using only one feature named "main feature", and the main feature can be any feature in the total feature pool. Thus decoding the main feature sets a constraint on the search space for other features. Likelihood from other features with the same state sequence will be added (with a weight estimated under certain criteria such as Minimal Classification Error (MCE) or Maximum A Posteriori (MAP)) before pruning. The likelihood for other features can be estimated using technology similar to forced alignment (the decoded state sequence as the target of the underlying phonetic targets). The purpose is that by using additional (and hopefully complementary) information, the correct path will not be pruned away. The flow chart of this approach is given in Figure 6.1. This process is conducted for every stream of feature representations, and the recognition results from each stream will be fused to give the final recognition output.

The fusion work in this thesis is based on the time synchronous Viterbi search and the general run time fusion architecture is showed in Figure 5.2. The detailed flowchart of our constraint fusion scheme is shown in Figure 6.1.

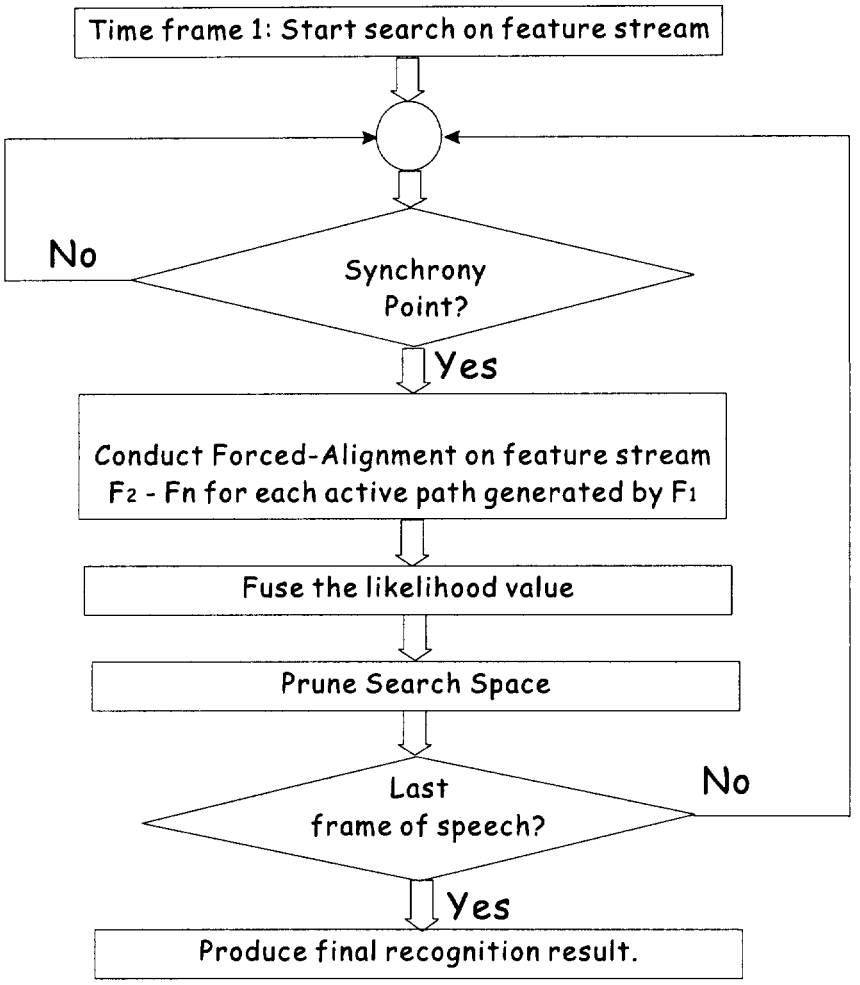


Figure 6.1: Flowchart of Constraint Fusion

6.1.1 Constraint Fusion Implementation - Modification on Token Passing and Token Merge

Experiments show that the state log likelihoods are very small ($<< 10^{-5}$) except for a small number of the most likely states. This characteristic is one reason why beam search is so effective. However, it also introduces search errors. As described in Section 2.4.4, beam search width is determined by the maximum partial path likelihood *MaxLog* at time *t*. The single state that obtains the *MaxLog* will have the determining effect on the pruning decision. Beam search is based on the assumption that the highest partial path

value is from the most likely state sequence. However, the most likely state sequence is not known during the time synchronous search. Thus the beam pruning decision is based on an approximation that may not be accurate enough. The partial path that was in fact part of the most likely path could be pruned away before reaching the end of the utterance. There are many possible reasons that can result in this kind of search errors. For example, at one or several frames, the acoustic event could be poorly modelled by the acoustic models. The *MaxLog* value could be set by an incorrect partial path. For some other partial paths, their state log likelihood values during that period could be very small and the accumulated likelihood is too small to survive the pruning. Although these partial paths are in fact part of the most likely paths. If we keep them alive beyond that short period, they may survive following pruning and result in most likely paths. Unfortunately, we could not know at that point which partial path will be part of the most likely path. Even worse, such a pruning error is unrecoverable, which means there is no way to get that partial path back into the active search space due to the nature of time synchronous search.

Another reason is as follows. Especially in decoding difficult acoustic events, the differentiating ability of the acoustic model is extremely weak. A tremendous amount of tokens could be generated in a matter of several frames. Histogram pruning kicks in during this circumstance to control the total number of active tokens. However, histogram pruning is a compromise to speed up search and more prone to prune promising paths.

To avoid pruning errors, the beam width must be set large enough. However that significantly increases the search effort and therefore is not an ideal solution. In this thesis work, we proposed and implemented some novel pruning methods under our fusion framework. Under constraint fusion, the pruning strategy is governed by two principals:

- **All features have a vote.**

  Under beam pruning, all active tokens at the current frame are subjected to a set of pruning steps guided by some preset thresholds. In the traditional approach, only

one feature is used. A preset threshold is compared with the highest value from that single feature. The pruning decision is deemed by the acoustic model accuracy of that feature. Under our constraint fusion framework, the pruning is no longer decided by one single feature as the traditional approach does. Instead, all features participate in the pruning decision. The pruning accuracy is now related to the accuracies of all acoustic models.

- **Main feature has the final say.**

  Different from the situation above, there are points that only one token in a token pool can be further propagated. For example, under traditional Viterbi decoding, only the "best" token is propagated to the next frame when there are multiple active tokens reside on the same state. Within constraint fusion framework, this "best" token is selected by the main feature.

Extensions on Token Structure

The token concept was introduced in Section 2.4.2. To accommodate our fusion scheme, we need to extend the original token structure. One of the extensions is the log likelihood that is stored in the token is changed from a singleton to an array. There are two kinds of arrays depending on the fusion method we use:

- The array is $(N+1) \times 1$. Log likelihood values of all features plus the fused one are stored in this $N+1$ dimension array. Element $(i, 0)$ in this array is the token's partial path score for feature i.

- The array is $(N+1) \times 2$. Compared to the first kind of array, an additional column is added. Each element $(i, 1)$ in that column records the relative rank of the partial path score among all active tokens for feature i.

Token Merge

Before we detail the implementation of our constraint fusion approach, we first introduce a technology that we used in our system, called Token Merge. The token merge (or recombination) approach is based on the assumption that for each active state in the lexical tree at *t*, all but one token that has the same language model history can be discarded. The language model history can be retrieved from each token, and the range of this history depends on the actual m-gram language model we are using. For a bigram language model, all tokens with the same last word are merged into one token. For a trigram language model, all tokens with the same last two words are merged.

The token merge enables us to incorporate the language model into our lexical tree search without making any tree copies. The token merge is conducted on the token list of each active lexical tree state at frame *t*. Suppose at time frame t, state s has an active token list called $TokenList(t; w; s)$. $TokenList(t; w; s)$ is a linked list structure that has *K* tokens linked together.

The token merge procedure is as follows:

1. Cluster tokens into a set of token lists.

   Start searching $TokenList(t; w; s)$ from head to end, fetch a token k and compare it with corresponding $TokenList\_LM(t; w; s; m)$ according to its m-gram history m. Generate a new $TokenList\_LM(t; w; s; m)$ if it does not exist.

   The pseudocode for this procedure is shown in Figure 6.2:

   After the operation above, all tokens are clustered into a set of token link lists. Every such token link list has a unique m-gram history.

2. Merge the best tokens from all clusters (token link lists).

   The best scored tokens in each cluster have been sorted out in the previous step. We simply concatenate these tokens into a new token link list $TokenList'(t; w; s)$ and attach it back to state s.

If (matched $TokenList\_LM(t; w; s; m)$ does not existed) {
    Create a new linked token list $TokenList\_LM(t; w; s; m)$.
    Put token k on $TokenList\_LM(t; w; s; m)$.
} Else {
    $\hat{\Gamma}_n(t; w; s)$ = score of $TokenList\_LM(t; w; s; m)$.
    If (the score of token k $\Gamma_k(t; w; s) > \hat{\Gamma}_n(t; w; s)$) {
        Replace the token on $TokenList\_LM(t; w; s; m)$ with current token;
    } Else {
        Discard token k.
    }
}

Figure 6.2: Pseudocode for Token Merge

Note: Token merge is decided by the main feature.

After the token merge operation, the pointers to surviving tokens are placed in an one dimensional array. The index of this array is mapped to the m-gram language model. Thus each m-gram language model combination has a unique index in the array. For example, the m^{th} element of this array stores the pointer to a token list named $TokenList\_LM(t; w; s; m)$.

After this merge operation, all tokens in $TokenList'(t; w; s)$ differ in their language model context. The m-gram language model is integrated into the search process without using tree copies (Section 2.4.1).

6.1.2 Constraint Fusion Experiments - Fusion Based Pruning

Constraint fusion is used as the platform for testing our proposed *cross-reference pruning* strategy [101].

In our implementation, once a feature is selected as the main feature, the remaining features will serve as consultants (supporting features) to the selected main feature. During the token pruning, when the decision is to keep a token alive under the traditional pruning strategy (in our implementation, the difference between the maximum token likelihood at current time frame and the likelihood of this particular token in comparison to a

preset threshold), no consultation is requested on the remaining features. If the decision is to prune away this token, a consultation is made via cross-referencing the same path in the search spaces of the supporting features. If the path could survive in the supporting feature spaces, then the path will be kept in the main feature search space.

The first set of experiments were used to measure how well this cross-pruning strategy worked. In these experiments, the main feature was handled as it would be in a standard recognizer except the pruning. The purpose of these experiments was to measure how many most likely paths can be saved by the cross-referencing pruning. In other words, how well the cross-referencing pruning approach compares to conventional beam pruning. This measurement was performed on both the word sequence and the word graph output.

6.1.3 Using Fusion to Improve Word Graph Quality

The word graph quality determines the success of post-recognition approaches and multi-pass decoding. The quality of a word graph can be defined by two measurement:

- Word Graph Error Rate (WGER)[70]: The best WER that can be reached by choosing a path in the graph, so it's the oracle word error rate that can be achieved by extracting a path from the word graph.

- Word Graph Density (WGD): A measurement of word graph size, it is defined as the total number of graph edges divided by the number of actually spoken words. The lower the WGD, the more compact the graph is.

WGER decides the WER lower bound that a second pass decoding or re-scoring can reach. Reducing WGER will improve the performance of other post-recognition approaches [56, 57, 82]. WGD reflects the search cost of generating the word graph, and it also affects the cost of re-searching the graph. Low WGD not only reduces computation cost but also benefits the accuracy because there are fewer incorrect hypotheses in the graph. Generally, we wish to reduce both WGER and WGD. The common way to change WGER or WGD is by adjusting beam widths. However, the common approach

cannot reduce both WGER and WGD at the same time. Increasing beam widths will reduce WGER but also increase WGD. WGER is not guaranteed to be reduced by this method because we cannot prove the new included hypotheses contain better paths. The trick of adjusting the beam widths are based on experience and dry run experiments. A good beam width setting strategy is to reach an optimal WGER and WGD combination.

Compared to tasks with similar baseline WERs,[1], the SPINE task has a much higher WGER; it is one unique characteristic that distinguishes it from other tasks. As reported in [91], our previous constraint fusion experiments show encouraging signs of reducing the WGER by using information fusion during decoding.

Table 6.1: Effect of constraint fusion based pruning on reducing WGER and WER

System	WGD	WGER	WER
TRAPS	100	22.6%	29.9%
TRAPS + MFCC	99.2	20.1%	29.8%
TRAPS + TLDA	102.0	20.6%	29.9%
TRAPS + WMFCC	100.7	20.3%	29.8%

TRAPS + MFCC means TRAPS is the main feature and MFCC serves as the consultant.

The result in Table 6.1 shows the cross reference pruning strategy indeed rescued some most likely paths from being pruned. It shows the fusion based pruning is more accurate than the conventional beam pruning guided by a single feature. The complementary information from multiple feature sources is more reliable for the pruning decision.

The improvement on WGER is not reflected in the WER. We compared the WERs from the same experiments, and there are just small improvements for fusion based systems (Table 6.1). These improvements are not significant.

A reasonable explantation is that although those partial paths were rescued by other consultant features from being pruned, they still can not win out at the end of utterance under the main feature's criterion. In other words, the likelihood differences for the partial

[1]The WGER for Voicemail and Switchboard tasks is about 9% and 9.5% respectively, although the WER for these two tasks (33.7% and 38.5%) are comparable with SPINE task.

paths at time t are too large for them to catch up with the best hypothesis at time T. In more detail, since at every time frame, state log likelihoods are very small ($<< 10^{-5}$) except for a small number of the most likely states. For some partial paths which endured small log likelihoods for several states, their cumulated partial path scores for those states will be much lower than the best partial path. Although we could save them from being pruned. These partial paths have little chance to be the number one hypotheses at the end of utterance. This certainly shows the efficiency of beam pruning for a single feature. However it also shows the weakness of current ASR architecture. Because the failure of acoustic models on a few states can be fatal to the whole recognition.

An encouraging sign in these experiments are that many partial paths caught up with others and were included in the final word graph. Further analysis also shows the WER of NBEST list (best N hypotheses) was also reduced. So if the utterances are long enough to let those partial paths have ample time to catch up, the final WER could be reduced further. This hypothesis requires a detailed error pattern analysis and carefully designed experiment to verify. We will work on it in the future.

6.1.4 Constraint Fusion Experiments - Fusion Based Final Recognition Output

In the experiments above, we implemented the constraint fusion concept into the beam pruning. We observed some improvement on the word graph quality but little gain on the final recognition output. In this Section, as proposed before, we use fused likelihood as the criterion to select the final recognition output. Similar to the implementation above, during the decoding, one feature is assigned as a "main feature", which constrains the possible state sequences and the search space of the following search. The search is conducted as usual but at phone (or word) boundaries, the likelihood for all features were added together. A set of weights α_i are obtained by training on a development data set. Each feature has its own weight and the fused likelihood value is decided by the following equation:

$$\Gamma_{N+1}(t;w;s) = \sum_{i=1}^{N}\{\alpha_i \cdot (\Gamma_i(t-1;w;s) + logP(s_i(t)/s_i(t-1)) + logb(o_i(t)/s_i(t)))\} \tag{6.1}$$

where

$P(s_i(t)/s_i(t-1))$ is HMM state transition probability of feature i,

$b(o_i(t)/s_i(t))$ is HMM state observation probability of feature i,

and $\Gamma_{N+1}(t;w;s)$ is the fused partial path score at time t and state s.

The α_i in the Equation above satisfies the following Equation:

$$\sum_{i=1}^{N} \alpha_i = 1. \tag{6.2}$$

Upon a token first entering into the entry state of a node, we need to add up its factored language model probability:

$$\Gamma_i(t;w;s) = \Gamma_i(t;w;s) + log\omega(s(t)/w) \tag{6.3}$$

where $1 \leq i \leq N+1$, and $\omega(s(t)/w)$ is the factored language model probability.

When a token is finally leaving the last state of a node, we need to remove the factored language model probability after all the pruning:

$$\Gamma_i(t;w;s) = \Gamma_i(t;w;s) - log\omega(s(t)/w) \tag{6.4}$$

Not only different features have different weight value during the likelihood fusion, they play different roles. The search space is pre-defined by the main feature, and the other features take a role in the pruning decision. In Equation 6.1, state s and word w are determined by search path of the main feature. And the selection of best (partial) hypotheses is determined by the fused likelihood $\Gamma_{N+1}(t;w;s)$.

Table 6.2 shows the results when combining two features using the constraint fusion. Compared with the baseline non-fusion system (row "TRAPS" in Table 6.2), fusion systems outperformed it in every case. The improvements are statistically significant.

Table 6.2: Effect on reducing WER by fusing likelihoods from different features.

System	WER
TRAPS	29.9%
TRAPS + MFCC	28.4%
TRAPS + TLDA	28.7%
TRAPS + WMFCC	28.4%

Constraint fusion was applied on two features with TRAPS as the main feature in all experiments. TRAPS + MFCC means TRAPS is the main feature and MFCC serves as the consultant.

Table 6.3: Comparison on WER reduction by using constraint fusion approach with different main feature.

	Main Feature		
	MFCC	TLDA	TRAPS
Baseline	27.7%	28.6%	29.9%
Fusion	26.8%	27.7%	28.4%

Fused likelihoods were applied.

In Table 6.3, the columns "MFCC", "TLDA" and "TRAPS" denote the experiments that the corresponding feature is selected as the main feature. Row "Baseline" denotes systems that did not apply the proposed cross-referencing pruning strategy and the row "Fusion" denotes systems that applied the proposed strategy.

Since our run time fusion approach performed at different stage compared with pre- and post-recognition fusions. These approaches can be combined in a sequential way. For example, we further conducted ROVER on the outputs from both the baseline and fusion systems (Table 6.3). The experiments shows our approach can be successfully combined with post-recognition fusion methods such as ROVER (Table 6.4).

Table 6.4: Further WER reduction by applying ROVER after constraint fusion

System	MFCC	TLDA	TRAPS	ROVER
Baseline	27.7%	28.6%	29.9%	27.3%
Run Time Fusion	26.8%	27.7%	28.4%	26.5%

6.1.5 Fusion with Dynamic Beam Adjustment

In large vocabulary speech recognition, the potential search space is prohibitive for a full search. Beam search is necessary to limit the search space by pruning away the less likely tokens. In the time synchronous search framework, beam search means at every time frame, only the most promising tokens are retained. Beam search is based on the assumption that the highest partial path value is from the most likely state sequence. However, the most likely state sequence is not known during the time synchronous search. Thus the beam pruning decision is based on an approximation that may not be accurate enough. Consequently the beam width is set to a rather large value to ensure that the most likely state sequence is not pruned. The side effect of setting a larger beam width is that more tokens are retained and the computation cost is increased.

Although beam search is very effective to control the active token numbers in most cases, there are periods for which the number of active tokens can be extremely high, because the potential size of the search space is very large. According to Odell [68], the peak number of active models is about 100 times greater than the average number for a WSJ 5k test. This many active tokens consumes a lot of memory and CPU cycles. The acoustic uncertainties, such as noisy speech, make this situation worse. When decoding unintelligible speech segments, no single token has a dominative score over other tokens, the recognizer produces a tremendous amount of tokens because the beam pruning is no longer effective. For tasks containing significant amount of noise or spontaneous speech such as SPINE task, this condition often occurs.

To accelerate decoding under such conditions without any damage to the recognition accuracy, we proposed a dynamic beam adjustment under our run time fusion scheme. In this approach, all features are involved in the pruning decisions. Each feature has its own set of beam pruning values, some of these beam values are dependent on each other, such as total allowed token numbers, but most of them are independent. The tokens are first evaluated by each feature's pruning module, their results are fused to make the

final pruning decision. At each time frame, we gather some statistical information on the beam pruning effect of each feature. For example, at time frame *t*, word-end beam width of feature f_i keeps α_i percentage of tokens active. The word-end beam width for these features at time frame *t+1* is be adjusted according to the following formulas:

$$\varphi = \min_i(\alpha_i) * \beta; \tag{6.5}$$

$$WordEndBeam_i = WordEndBeam_i * (1 - \frac{\alpha_i - \varphi}{\varphi}); \tag{6.6}$$

where β is a weight value used to adjust the threshold φ.

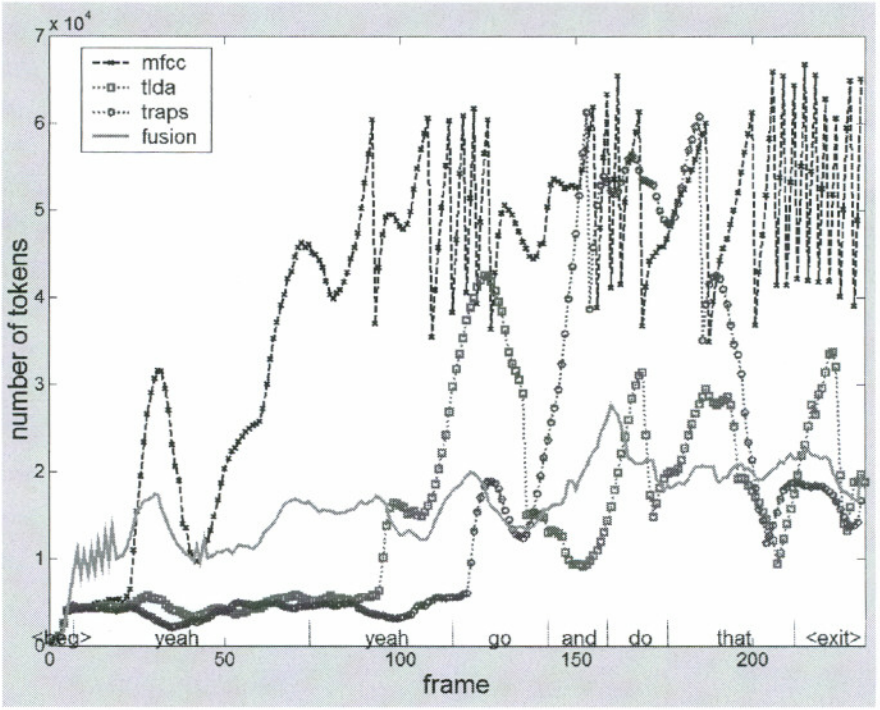


Figure 6.3: The comparison of active token numbers along the time frame during decoding one sentence

When the survived token percentage of feature f_i is over or less than φ, we reduce or

increase its pruning widths according to equation 6.6. This approach is based on the rationale that unintelligible speech segments often occur in some continuous frames. Rather than let lots of unpromising tokens go through each pruning stage till the final histogram pruning, we try to reduce the active tokens in the earlier stage of pruning. Experiments show this approach significantly speeds up the search without any loss of recognition accuracy. Figure 6.3 compares the active token numbers of dynamic beam adjustment with conventional single feature pruning. The curve of the fusion approach is relatively flat and the token numbers are in the range of 10k to 30k. The three features have some peaks over 40k tokens.

6.2 Composite Fusion

From the point of view of HMM topology, the previous proposed constraint fusion scheme relies on one feature as the main feature and needs a final step to fuse the individual results. In the composite fusion approach, the fusion is conducted in the single composite recognizer. Each feature still has its own corresponding HMMs; the search of each feature runs independently in its state space, except that time synchrony is required only at predefined boundaries. The boundaries can be phone boundaries or word boundaries.

The composite HMMs (Figure 6.6) are a set of parallel traditional HMMs which share the same begin and end dummy nodes for the selected units (can be phone or word). Different streams make the transition at the same time for begin and end nodes (synchronizing at segment level rather than frame level) but each stream of HMMs makes their own state transitions independently in the internal states. To ensure global optimization, the EM algorithm can be applied so that HMMs for different streams are trained simultaneously and jointly. Similarly, the existing single stream decoder is extended to ensure the time synchrony jointly at boundaries (the specified begin and end nodes for phone or word).

This scheme is quite different from the multi-stream or multi-band approach, where

different streams/sub-bands are trained and decoded independently by different MLP recognizers. The proposed topology can be viewed as a generalization of existing approaches. The HMM topology of the concatenated approach in pre-recognition fusion is illustrated in Figure 6.4, which is a trivial case of our proposed architecture. In the concatenated approach, multiple streams are merged into one input vector for recognition; only one set of models is needed. While the post-recognition fusion method can be viewed as a composite model at sentence level as shown in Figure 6.2. In the post-recognition fusion approach, multiple streams are searched independently during run time. The underlying state sequences of each path in this diagram are not necessary the same. Furthermore, each path may have different number of states. The only similarity for these paths are they share the same dummy nodes at the sentence beginning and end.

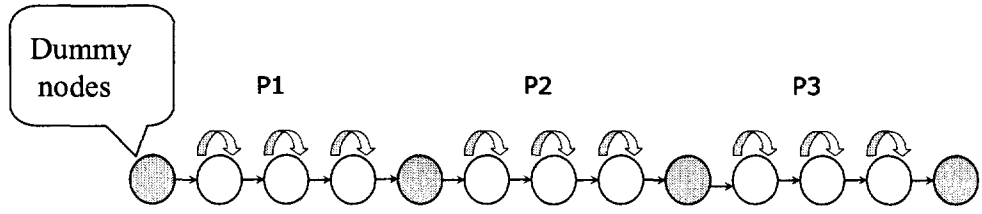


Figure 6.4: Existing art: Concatenated approach in Pre-recognition fusion

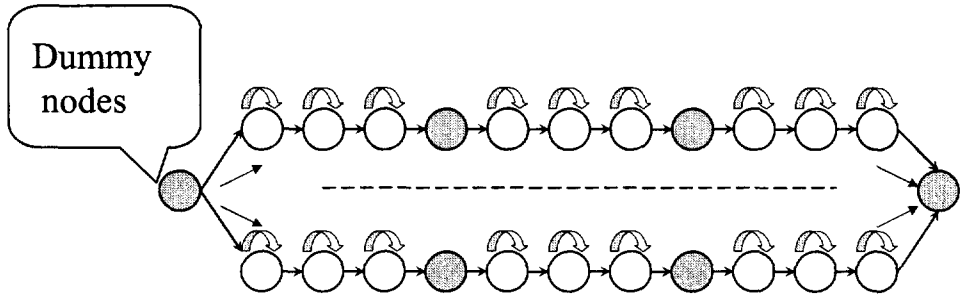


Figure 6.5: Existing art: Post-recognition fusion

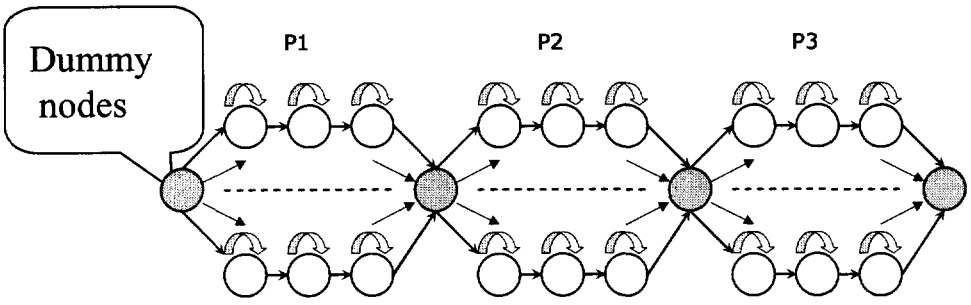


Figure 6.6: Our run time fusion approach

A combination of synchronous and asynchronous search at run time among different feature streams. The underlying state sequences of each partial path that lies between two dummy nodes are not necessary the same. Although these partial paths have the same start and end time in that period.

These figures show a more detailed view on the shortcomings of the existing fusion solutions. The drawback of pre-recognition fusion approach is that only frame-based features can be incorporated (or only time synchronized feature streams can be incorporated). Segmental based information, such as tones (or pitch patterns), cannot be integrated easily into this framework during run time.

The drawback of post-recognition fusion approach is that the time-dependency between different features is completely ignored during the recognition of each stream. In other words, there is no interaction between different features during the decoding process, i.e. the presence of one feature stream does not impact the course of decoding in other feature streams.

The composite fusion approach is a combination of synchronous and asynchronous search at run time among different feature streams. Compared with these existing approaches, the constraint on frame level synchronization will be relaxed in our proposed work, and enables features with different time/frequency resolutions and time spans (such as segmental based features) to be readily integrated.

6.2.1 Fused Viterbi Algorithm

In Section 2.3, we introduced the Viterbi algorithm. Under Viterbi approximation, the "$\sum$" in the following Equation:

$$\hat{W} = \arg\max_{W} P(W)P(O|W) = \arg\max_{W} P(W)\sum_{s_1^T} P(O, s_1^T|W) \tag{6.7}$$

was replaced by "max".

The new search equation becomes:

$$\hat{W} = \arg\max_{W} P(W)P(O|W) = \arg\max_{W} P(W)\max_{s_1^T} P(O, s_1^T|W), \tag{6.8}$$

where s_1^T means all possible state sequences from time *1* to *T*.

The summation in the first equation is for all possible state sequences under the constraint of word sequence W. Summing all possible state sequences requires a thorough search through the whole search space which is unaffordable. And in the second equation, only the most probable state sequence is considered. Under this Viterbi approximation, the most likely word sequence is **approximated** by the most likely state sequence. This approximation works well in practice but is certainly a sub-optimal assumption.

Viterbi approximation assume that the likelihood of the best path can approximate the sum over the likelihood of all paths. It works well when the acoustic and language model ambiguities are low and the best path dominates other paths. This approximation is in doubt when the acoustic and language models can no longer describe the speech signal well. In these difficult situations, the distance between the best and other paths will be narrow and the likelihood of other competing paths are not appropriated to be ignored.

The problem above is addressed under our multiple features fusion framework:

$$\hat{W} = \arg\max_{W} P(W)P(O|W) = \arg\max_{W} P(W)\sum_{i}^{N}\max_{s_1^T(i)} P(O, s_1^T(i)|W), \tag{6.9}$$

where i is the feature index, N is the total number of features. Thus $s_1^T(i)$ means all possible state sequences from time *1* to T under feature i.

In Equation 6.9, each feature has its own Viterbi alignment, and the best word sequence is decided by the fusion of the Viterbi alignments. To accommodate this extension, we need to modify our token passing strategy. For each active state, instead of keeping only one best token as in conventional token passing, we keep the one best token for each feature. So at each time frame t, we need to find the best token for each feature and connect them in a linked list, then attach this linked token list to the active state. We always keep N tokens for each state if any one of them can survive the pruning, which means that duplication is allowed in the N tokens list. Duplication occurs when more than one feature selected the same token as the best token. If that occurs, we will duplicate that token and add the new token to the list. The reason is this agreement can be temporary among these feature, most likely they will go their separate ways in the following search. An illustration of this extension is showed in Figure 6.7.

6.2.2 Fused Token Propagation

The search is implemented by propagating tokens through the lexical tree. The integration of token propagation with a re-entered lexical tree makes it possible to use a single tree instead of making multiple tree copies. Nevertheless, the token propagation is different when it occurs within the tree and across the tree.

The algorithm for within-tree token propagation in our fusion framework is:

1. First calculate the new partial path scores for each feature i:

$$\Gamma_i(t; w; s(t)) = \Gamma_i(t-1; w; s(t-1)) + logP(s_i(t)/s_i(t-1)) + logb(o_i(t)/s_i(t)) \tag{6.10}$$

   where

   $P(s(t)/s(t-1))$ is HMM state transition probability

   $b(o(t)/s(t))$ is HMM state observation probability

2. Update language model lookahead value:

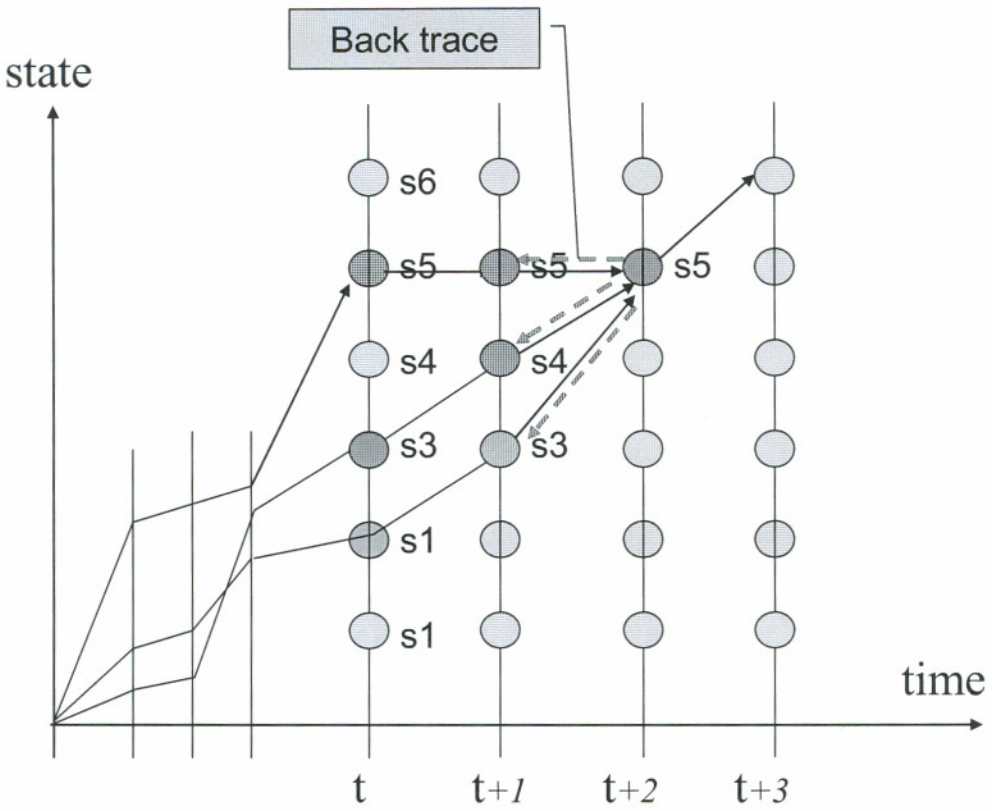


Figure 6.7: An example of our extended Viterbi search

If state s is the entry state of a node, and it is the first time this token enters into s, add its factored language model probability:

$$\Gamma_i(t; w; s) = \Gamma_i(t; w; s) + log\omega(s(t)/w) \tag{6.11}$$

where $1 \leq i \leq N$, and $\omega(s(t)/w)$ is the factored language model probability.

If state s is the exit state of a node, and it is the last time this token stays in s, we remove that factored language model probability before it leaves state s:

$$\Gamma_i(t; w; s) = \Gamma_i(t; w; s) - log\omega(s(t)/w) \tag{6.12}$$

3. For all tokens in frame t, find their best score $\hat{\Gamma}$ for all features

$$\hat{\Gamma}_i = \max_{w;s} \Gamma_i(t; w; s) \tag{6.13}$$

4. (a) If HMM state is the synchrony point, perform fusion here:

if ($\forall\ \Gamma_i(t; w; s) > \tau_i * \hat{\Gamma}_i$) { [2]

i. Clone current token to a new token and update its information for all features.

ii. According to equation 6.9, calculate the fused partial scores by summing the best partial scores over all features:

$$\Gamma_{fused} = \sum_{i=1}^{N} \hat{\Gamma}_i. \tag{6.14}$$

iii. Attach this new token to the linked token list of state s.

} else{

i. Discard this token.

ii. Exit this routine.

}

(b) If HMM state is not the synchrony point, perform normal token propagation:

if ($\forall\ \Gamma_1(t; w; s) > \tau_1 * \hat{\Gamma}_1$) {

i. Clone a new token and update its information;

ii. Attach this new token to the linked token list of state s.

} else{

i. Discard this token.

ii. Exit this routine.

}

[2] τ_i is the beam pruning threshold for feature i

The algorithm for cross-tree token propagation is more complex than the within-tree propagation for several reasons. The cross-tree token propagation starts at the leaf nodes of the lexical tree, and ends at the first level triphone nodes:

1. When a token leaves the last state of a leaf node, we are already able to identify the newly generated word. Thus the actual language model probability is added to the path score:

$$\Gamma_i(t; w; s) = \Gamma_i(t-1; w; s_{t-1}) + logP(w) \tag{6.15}$$

where $logP(w)$ is the language model probability for word w and in the trigram case: $logP(w) = logP(w/w_1, w_2)$

2. Clone a new token and update its information including the path history.

3. For all tokens in frame t, find their best score $\hat{\Gamma}_i$ for all features

$$\hat{\Gamma}_i = \max_{w;s} \Gamma_i(t; w; s) \tag{6.16}$$

4. (a) If word is the synchrony point, perform fusion at this point:

   if $(\Gamma_i(t; w; s) > \tau' * \hat{\Gamma}_i)$

   i. Clone a new token and update its information;

   ii. According to equation 6.9, calculate the fused partial scores by summing the best partial scores over all features:

$$\Gamma_{fused} = \sum_{i=1}^{N} \hat{\Gamma}_i. \tag{6.17}$$

   iii. Attach this new token to the linked token list of state s.

   else

   i. Discard this token.

   ii. Exit this routine.

   where τ' is the **word-end** beam pruning threshold

(b) If word is not the synchrony point, perform normal propagation:

if $(\Gamma_1(t;w;s) > \tau_1' * \hat{\Gamma}_1)$

i. Clone a new token and update its information;

ii. Attach this new token to the linked token list of state s.

else

i. Discard this token.

ii. Exit this routine.

5. Propagate the tokens from leaf nodes to the first level tree nodes:

Suppose the leaf node represents triphone "a-b+c", and its successor nodes can be described as "c-*+*". For each of first level node "c-*+*" that are reachable from the word end node "a-b+c", we need to calculate their new scores:

$$\Gamma_i(t;w;s) = \Gamma_i(t;w;s) + logP(s_i = 0/s_i = -1) + logb(O_i(t)/s_i = 0) \quad (6.18)$$

where $\Gamma(t;w;s)$ on the left hand of the equation is from Equation 6.15.

6. For all tokens in frame t, find their best score $\hat{\Gamma}_i$

$$\hat{\Gamma}_i = \max_{w;s} \Gamma_i(t;w;s) \quad (6.19)$$

7. (a) If word is the synchrony point, perform fusion as follows:

if $\forall\ (\Gamma_i(t;w;s) > \tau_i * \hat{\Gamma}_i)$

i. Clone a new token and update its information according to $\Gamma(t;w;s)$ for all features;

ii. According to equation 6.9, calculate the fused partial scores by sum the best partial scores over all features:

$$\Gamma_{fused} = \sum_{i=1}^{N} \hat{\Gamma}_i. \quad (6.20)$$

iii. Attach this new token to the linked token list of state s (state s is the first state of node "c-*+*").

(b) If word is not the synchrony point, perform normal pruning:

if $\forall\ (\Gamma_i(t;w;s) > \tau_i * \hat{\Gamma}_i)$

i. Clone a new token and update its information according to $\Gamma(t;w;s)$ for all features;

ii. Attach this new token to the linked token list of state s (state s is the first state of node "c-*+*").

Modification to Token Merge

In Section 6.2.1, we stated what was necessary for an extension to the token structure. For each active state, instead of keeping only one best token as in conventional token passing, we keep one best token for each feature. In implementation, the extension is executed on the token merge (Section 6.1.1) module. For each active state in the lexical tree at t, cluster all tokens that have the same language model history, and keep the best token for each feature and discard the rest.

At each time frame t, we need to find the best token for each feature and connect them in a linked list, then attach this token linked list to the active state. Figure 6.8 shows such a token link list.

Suppose at time frame t and state s, there is an active token list with language model history m, and this token list is named as $TokenList(t;w;s)$. $TokenList(t;w;s)$ has K tokens organized as a linked list. The new token merge procedure is as follows:

1. Start search $TokenList(t;w;s)$ from head to end, fetch a token k and compare it with corresponding $TokenList\_LM(t;w;s;m)$ according to its language model history m. Generate a new $TokenList\_LM(t;w;s;m)$ if it does not exist. The pseudocode for this procedure is shown in Figure 6.9.

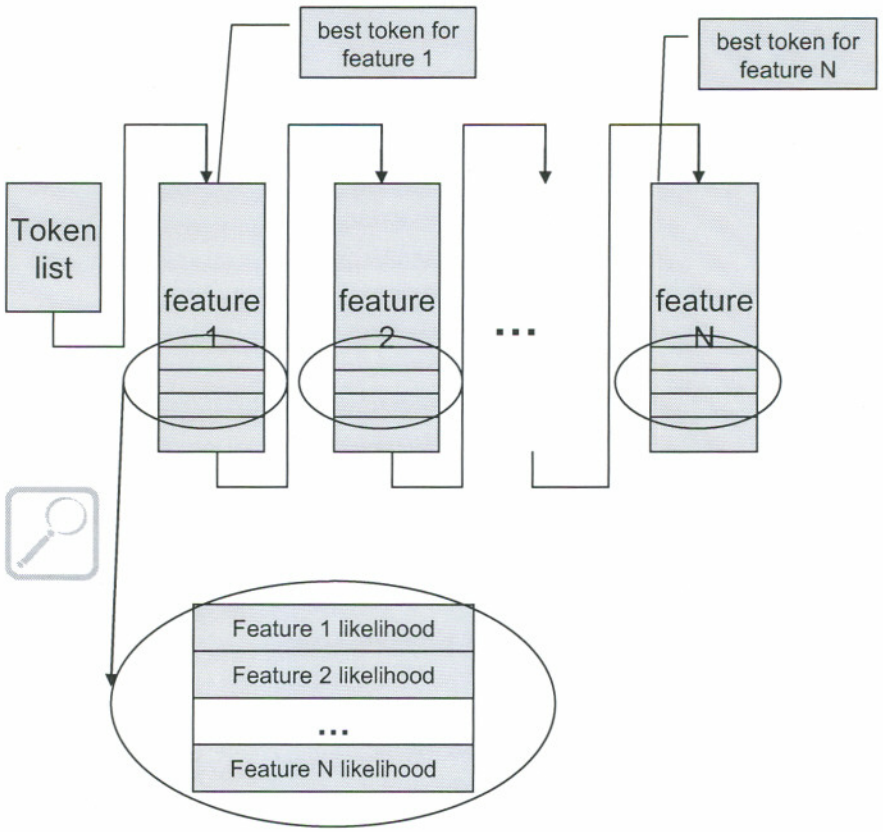


Figure 6.8: An example of Token Link List used in extended Token Merge

The token list contains more than one token, and each token is placed according to the order of the features.

After the operation above, all tokens are clustered into M token link lists with unique m-gram history.

2. Merge the best tokens from each cluster.

   The best scoring tokens in each cluster are already sorted by the previous step. We simply concatenate these tokens into a new linked token list $TokenList'(t; w; s)$, and attach it to the tree node s. Because these tokens are in an ordered list, after this merge operation, all N tokens of each $TokenList'(t; w; s)$ will have the same LM context.

If (matched $TokenList\_LM(t; w; s; m)$ does not exist)
  Create $TokenList\_LM(t; w; s; m)$.
  Generate N copies of token k.
  Link these new tokens onto $TokenList\_LM(t; w; s; m)$.
Else
  matched $TokenList\_LM(t; w; s; m)$ is found;
  For $(i = 1; i < N; i++)\{$
    If $(\Gamma_k(t; w; s; i) > \hat{\Gamma}_m(t; w; s; i \times i))$
      Replace the i^{th} token of $TokenList\_LM(t; w; s; m)$ with a copy of token k;
    Else
      Discard current token if $i == N$.
  $\}$

$\hat{\Gamma}_m(t; w; s; i \times i)$ is the feature i score from the i^{th} token of $TokenList\_LM(t; w; s; m)$.
$\Gamma_k(t; w; s; i)$ is the score for feature i of token k.

Note: Token merge was conducted on all features.

Figure 6.9: Pseudocode for Token Merge

For an active state of the lexical tree, we keep the best token from each feature. These N tokens from N features have the same m-gram history and reside on the same state at the same time. Thus their scores are comparable. Different from previous implementations, in which the fused likelihood was calculated from one single token, in this approach, the likelihood values used in fusion are from different tokens:

$$Score(m) = \sum_{i=1}^{N} \alpha_i \hat{\Gamma}_m(t; w; s; i \times i) \tag{6.21}$$

In sentence level fusion, the final recognition output was decided by the following procedure:

- At the sentence end, search the maximum $Score(m)$ according to Equation 6.21.

- There are N tokens in token group m, and they have the same language model history. That means these N tokens may have different partial word sequences except the last L-1 words in an L-gram case. However, we can only select one token for final recognition output. The approach we used here is to calculate a confidence score for each token:

$$C_m(i) = \hat{\Gamma}_m(t; w; s; i \times i)/\Gamma_{max}(t; s; i), \tag{6.22}$$

  where $\Gamma_{max}(t; s; i)$ is the best score for feature i at time t.

- Find the maximum $C_m(i)$ and select token i of token group m.

- Back-trace token i and generate final recognition output.

The experimental result is shown in the following Table 6.5 and the result is superior to constraint fusion.

Table 6.5: WER reduction by using composite fusion with extended Viterbi

System	WER
MFCC	27.7%
TLDA	28.6%
TRAPS	29.9%
Composite fusion with extended Viterbi	25.3%

6.2.3 Composite Fusion - Improve Word Graph Quality

In Section 6.1.3, we reduced WGER from 24.0% to 20.9% by using the constraint fusion method, but it is still not satisfying. We further reduced the WGER by increasing the search beam widths at different levels. Although the WGER was reduced, we soon reached the limit of our computation power, especially the memory, by increasing beam widths. After a certain point, the WGER decreased very slowly by increasing the beam widths while the WGD increased rapidly. The improvement in WGER is at the expense

of a tremendous increase in word graph size, which lowers the word graph quality and diminishes the purpose of using a word graph.

Table 6.6: The effect of beam width pruning on word graph size and its accuracy

WGER	WGD	Real Time Factor
20.9%	95.3	3.5
19.7%	231.5	6
16.6%	255.2	7.5
14.1%	290.2	10

Note: Beam widths in these experiments are carefully tuned to give the best results.

It is important to improve word graph quality without significant increase in computation cost. A solution is proposed here that integrates the fusion concept into the lexical tree based time synchronous search (Section 2.4.3) and token merge (Section 6.1.1).

At every time frame (10 ms), a token merge operation is conducted on all tokens. For non-leaf nodes of the lexical tree, all tokens having the same language model history but different tree entering times are merged. For leaf nodes of the lexical tree, the new word w is known, and all tokens having the same preceding words W are merged into one token. These W are contained in the final word graph if they survive the following propagation. But tokens must pass various pruning stages before entering into the merge process. For example, right before token merge, we have a pruning step on all tokens from the active state *HMMs*.

$$\Gamma(t; w; s) > \tau * \hat{\Gamma} \tag{6.23}$$

$\hat{\Gamma}$ is the maximum likelihood of all tokens from *HMMs*.

τ is the beam pruning threshold.

Experiments show increasing τ is the most efficient method to reduce WGER and the results of Table 6.6 were obtained in this way. But as stated before, this approach is unsatisfactory. Part of our approach is that rather than simply increase τ, we perform a

fusion based pruning:

$$\forall(\Gamma_i(t;w;s) > \tau_i * \hat{\Gamma}_i) \tag{6.24}$$

$\hat{\Gamma}_i$ is the maximum likelihood value at feature i for all tokens from *HMMs*.

τ_i is the beam pruning threshold for feature i.

Suppose the τ_i have the same value as τ, then by increasing τ in Equation 6.23, we leave many unlikely tokens alive, thus hurting the quality of the word graph. But in Equation 6.24, only the most likely tokens considered by each individual feature are let in. The analysis above was verified by our experiments: at a similar WGD, significant WGER reduction by about 43% is achieved (Table 6.7). Additional optimizations, such as dynamic beam adjustment of τ_i, were also implemented by fusion statistics from various features. We will not elaborate the lengthy detail here because they are similar to the principal of fusion.

Table 6.7: Graph Word Error Rate reduction by cross-reference pruning coupled with dynamic beam width fusion

System	WGER	WGD
Baseline: Increase Beam approach	16.6%	200.2
Fused Token Pruning approach	9.4%	200.6

Second pass cross-word decodings were performed on the word graphs above using the same set of acoustic and language models. The improved word graph (Table 6.7) produced by the first pass decoding gave a 9.7% WER reduction in the second pass decoding (Table 6.8). Significance test (Table 6.9) shows this improvement is statistically significant and the difference is greater that 99.9%.

Table 6.8: The effect of improved WGER on the 2nd pass decoding.

System	WGER	WER
Baseline	16.6%	26.9%
Fused Token Pruning	9.4%	24.3%

Best quality word graph by fusion approach is beneficial to 2nd pass decoding.

6.3 Rank Based Fusion

A great advantage of our run time fusion over other fusion methods is that the hypotheses are automatically aligned across different features. At time frame t, all (partial) hypotheses are generated from the same part of speech, thus their scores for different features can be directly compared. This comparison can be conducted at all levels of a speech recognizer: state, phone, word, and sentence.

In the constraint and composite fusion approaches, we directly use likelihood from other feature representations to participate in the pruning stage during the search. In rank based fusion, we exploit the relative ranking in active search paths. Similar to the constraint and composite fusion approaches, the final result will be given by fusion of the recognition results from each feature representations. The difference is that the likelihood values are transformed into a series of rankings. Each feature has its own ranking order for the recognition hypotheses. The rankings for all features have the same range, such as $1..N$ for N hypotheses. Thus these rankings are comparable among different features. In contrast, the likelihood values from different features for the same hypothesis are not comparable because each of them has a different dynamic range.

6.3.1 Rank Based Fusion in SPINE Task

If we sort the hypotheses according to their likelihood values, each feature will give a different result. An approach we proposed is a rank based re-sort of the recognition hypotheses.

Table 6.9: Significance test result on the WER of 2nd pass decoding

	Baseline	Fused Token Pruning	
Matched Pair Sentence Segment		< 0.001	***
Signed Paired Comparison		< 0.001	***
Wilcoxon Signed Rank		< 0.001	***
McNemar		< 0.001	***
ANOVAR		< 0.001	***

- These significance tests are all two-tailed tests with the null hypothesis that there is no performance difference between the two systems.
- The first column indicates if the test finds a significant difference at the level of p=0.05. It consists of ' ' if no difference is found at this significance level. If a difference at this level is found, this column indicates the system with the higher value on the performance statistic utilized by the particular test.
- The second column specifies the minimum value of p for which the test finds a significant difference at the level of p.
- The third column indicates if the test finds a significant difference at the level of p=0.001 ("***"), at the level of p=0.01, but not p=0.001 ("**"), or at the level of p=0.05, but not p=0.01 ("*"). A test finds significance at level p if, assuming the null hypothesis, the probability of the test statistic having a value at least as extreme as that actually found, is no more than p.

1. For N best hypotheses, find the maximum and minimum likelihood value (denoted as $maxP$ and $minP$) for each feature i:

$$maxP_i = \max_{n=1}^{N} P_i(n) \tag{6.25}$$

$$minP_i = \min_{n=1}^{N} P_i(n). \tag{6.26}$$

2. Calculate the difference of $maxP_i$ and $minP_i$:

$$varP_i = maxP_i - minP_i. \tag{6.27}$$

3. For each NBEST hypothesis, calculate feature i's likelihood value difference with $minP_i$ and divide that by $varP_i$:

$$\Theta_i(n) = \frac{P_i(n) - minP_i}{varP_i}. \tag{6.28}$$

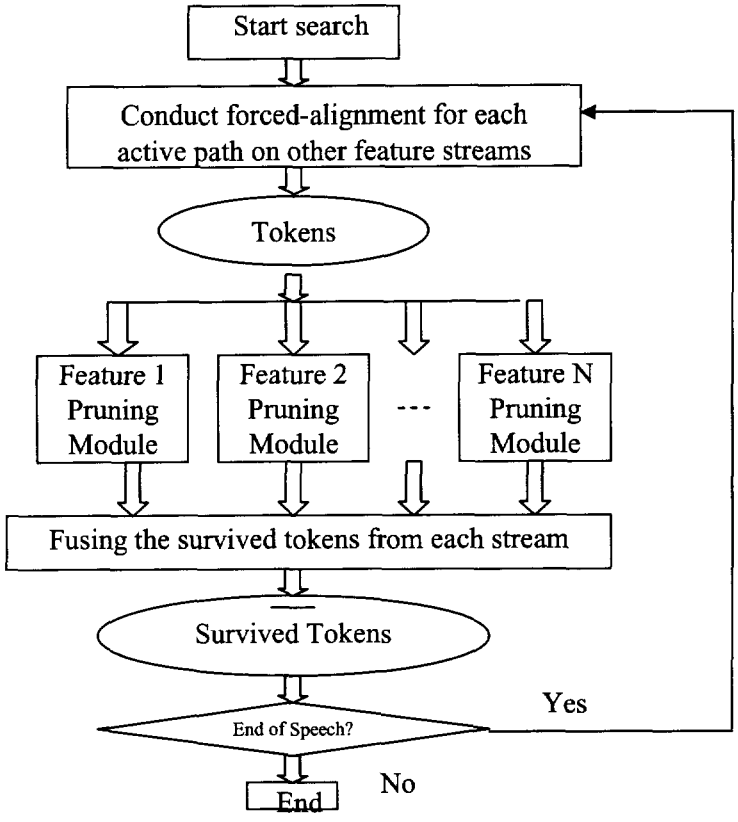


Figure 6.10: Run time fusion with Rank Based Token Pruning

$\Theta_i(n)$ reflects the relative likelihood value difference between hypothesis n and the worst hypothesis under feature i. It is a normalized value in the range of 0 to 1.

After performing the operation above, we have a set of Θ_i for each hypothesis. The Θ_i shows the confidence measurement for that hypothesis under feature i. We sum Θ_i for all M features:

$$\Theta(n) = \sum_{i=1}^{M} \alpha_i \Theta_i(n), \tag{6.29}$$

α_i is the weight value of each feature i.

We re-rank the N hypotheses according to the combined value $\Theta(n)$ in ascending order. The one that has the highest $\Theta(n)$ value is the new best hypothesis. As shown in Table 6.10, a 4.7% relative WER reduction was obtained in the rank based fusion.

Table 6.10: WER reduction by using rank based fusion

System	WER
MFCC	27.7%
TLDA	28.6%
TRAPS	29.9%
Rank Based Fusion	26.4%

α_i is 1, 0.9 and 0.8 respectively for MFCC, TLDA and TRAPS. Maximum value of M is 500 in our experiments.

In the experiment above, we used all available hypotheses up to the best 500. We did observe in some experiments that increasing the maximum M from 500 to 1000 hurts performance. One possible explanation is that hypotheses beyond 500 are not accurately modelled by all features, and their inclusion introduces some noise into out formula. How to optimize M at a global or sentence level still needs to be studied.

The rank based fusion has not achieved the best result among these three proposed fusion approaches. However it still has some potential to be explored. One possible direction is to implement the rank based fusion at a smaller unit level. In the current implementation, only sentence level hypotheses are used. Thus the rankings are influenced by the language model scores. One possible solution is to remove the language model scores before performing rank based fusion. Another possibility is to perform rank based fusion on the state, phone or word level, where the language model effect is smaller.

Chapter 7

Fusion in Speech Segmentation

In this chapter, we focus on using fusion on a different part of the speech recognition system, that of segmenting continuous speech into utterances (sentences) before generating the features. We presents a new speech segmentation approach based on two different level fusion. The first level of fusion applies to the spectral sub-bands and fuses multiple filter bank coefficients. This new approach takes advantage of current feature extraction procedures, with little additional computation cost. Another level of fusion was performed by fusing the results from several segmentation systems. Evaluation was conducted on the SPINE2 task. Experiments show our fusion based approaches significantly reduced the WER compare to two classifier-based approaches. Compared to the manual segmentation, our approach only has 0.3% WER increase. This new approach is our first try to explore multiple information sources at different stages in the recognition process.

7.1 Speech Segmentation Overview

The input speech stream to an ASR system is a continuous flow of speech signal without any type of boundary information. For recognition efficiency, the speech stream is first transformed into a sequence of audio segments. The basic task of speech segmentation is chopping long periods of speech into short ones and removing non-speech events at the same time. Additional tasks of a speech segmenter may also include segmenting and clustering the speech stream according to speaker identities, environmental and channel

conditions. In this thesis, we only focus on the basic task, segmenting the speech stream at the boundaries of speech/non-speech events. The segmentation process is also commonly referred to as endpoint (or silence) detection. The resulting segments are called utterances or sentencess, which is by no means linguistically accurate (these denominations do not strictly correspond to their linguistic counterparts).

Speech segmentation is necessary for an LVCSR system due to memory and speed restrictions of speech recognition (Section 1.1). Speech segmentation serves the following purpose in a continuous speech recognition system:

1. Segmentation reduces the network load in a client/server or cluster type ASR system. Separating the original long stretch of speech into short segments reduces the average feature file size.

2. Segmentation reduces the computational load of the decoder. The search space increases linearly with the length of speech. The longer the speech stream, the more words it may contain. And the potential search space would be increased correspondingly to a degree that the decoder can no longer afford. So it is necessary to constraint the length of input speech due to the memory and speed restrictions.

3. Segmentation increases the robustness of an ASR system. The continuous speech contains speech and non-speech parts. Non-speech parts carry no linguistic information and include silence, background noise, laugh, etc. These non-speech events appear between actual spoken words at random time for an unfixed period. Short periods of non-speech events are modelled by one or several special models in an ASR system. In spite of that, current technology still performs poorly when facing non-speech events especially when there is strong background noise or the duration is long. An ASR system often mis-recognizes non-speech events as words, causing insertion errors. Another type of errors caused by non-speech events is the substitution error, when the non-speech events corrupt neighboring speech events, causing the recognizer mis-recognized both regions.

4. Segmentation can improve the accuracy of some acoustic modeling methods such as Cepstrum Mean Subtraction (CMS). CMS is an approach used in feature generation to normalize the speaker variations. It is desirable to minimize the influence of channel information in CMS calculation. Long silence is the easiest channel information that can be identified with a reasonable accuracy. We found that the recognition performance can be improved by deleting most long silence from both training and decoding data [90].

Since manual segmentation of speech is time consuming and unrealistic in most conditions, various approaches on automatic speech segmentation have been proposed. According to [19, 76], these approaches can be categorized as follows:

1. Metric-based segmentation. This approach is based on the acoustic distance measurement between every two contiguous windows along the speech signal. The maximum distances are detected as potential segmentation points, and the final segmentation decision is based on some thresholds.

2. Classifier-based segmentation. This approach builds separate models for speech and non-speech events. The segmentation problem becomes a classification task. Gaussian mixture models or HMM are trained to model each class, and the final segmentation decision is based on the change point of classes. Another type of classifier-based approach runs the decoder on the speech stream to generate phoneme or word sequences. The final segmentation decision is based on the silence locations generated by the decoder.

Generally the metric-based approaches cannot compete with the classifier-based approaches on segmentation accuracy. However, classifier-based segmentations require complex computation and cause large latency, thus are not suitable for a real time application and are mostly used in off-line systems.

In this thesis, we propose a novel fusion-based segmentation approach that is highly accurate and demands little computation. We compare our approach with two classifier-based segmentation methods, which will be introduced in Section 7.1.1 and Section 7.1.2.

Segmentation is important in the SPINE task because there is lots of noise. Failing to exclude long periods of non-speech noise not only causes a large amount of insertion errors but also disrupts the search continuance. Current acoustic and language modeling techniques act awkwardly when facing such an interruption, and the damage usually spans to its neighboring speech. The raw speech data files of the SPINE task contain several minutes of conversation. The task of segmentation is to separate them into small segments and discard non-speech ones. Speech segmentation is the major interest in the SPINE1 workshop and remains an important topic in the SPINE2 workshop and following conferences. Almost all nine participants in the SPINE2 evaluation used classifier-based segmentation [49, 32, 66, 67, 73, 76, 98].

7.1.1 TRAPS Based Segmentation

The segmentation that we used for the official SPINE2 evaluation is a TRAPS based approach proposed by Dr. Hermansky's group [49, 41, 66].

The TRAPS based segmentation is based on two main processing steps. In the first step, we learned the distribution of the temporal patterns of speech/non-speech present in each critical band independently. This was performed by training a Multi-Layer Perceptron (MLP) in each critical band. The input to the MLPs is a one second long (two syllables long) temporal trajectory of critical band energy. The temporal trajectories were mean subtracted, variance normalized and hamming windowed before given as input to the MLPs. The output layer of the MLP consists of two nodes targeting speech/non-speech respectively. In the second step, we combined the outputs from each band-specific MLP and trained another MLP as a merging classifier. The output layer of this MLP again targets speech/non-speech events. The size of the hidden units is kept at 300 for band-specific MLPs and at 50 for the merging MLP. The output from this merging MLP

was then median filtered to give final speech/non-speech decisions.

7.1.2 Gaussian Mixture Classifier (GMC) Based Segmentation

We also obtained a segmentation from Dr. Richard Stern and Dr. Rita Singh of CMU.

> "A two-class speech/non-speech Gaussian mixture classifier was trained with KLT features from the SPINE2 development data. To train the classifier, the training data were segmented using Viterbi alignment. Feature vectors from segments corresponding to speech events (i.e. words and filled pauses) were used to train the speech distributions. All segments not corresponding to speech were used to train the non-speech distributions. Each of the distributions was a mixture of 32 Gaussians.
>
> During segmentation, the likelihood of each of the two classes was computed over a sliding window corresponding to 0.5 seconds of speech, where the window was advanced in steps of 20 ms. Histograms of the difference in the likelihoods of the classes were derived and the inflexion points between the modes representing speech and non-speech events were located. The value of the likelihood difference at the inflexion point was used as the threshold for the likelihood difference that separated speech from non-speech." [66, 76, 82]

7.2 Proposed Segmentation Approaches

7.2.1 Segmentation using Filter Bank (Subbands) Based Fusion

The third segmentation approach originates from my work on developing the SMFCC feature. It is a simple and efficient method based on fusion.

Filter bank calculation is a necessary step in many feature extraction algorithms. A filter bank is a set of band-pass filters that span the whole frequency spectrum. Each filter

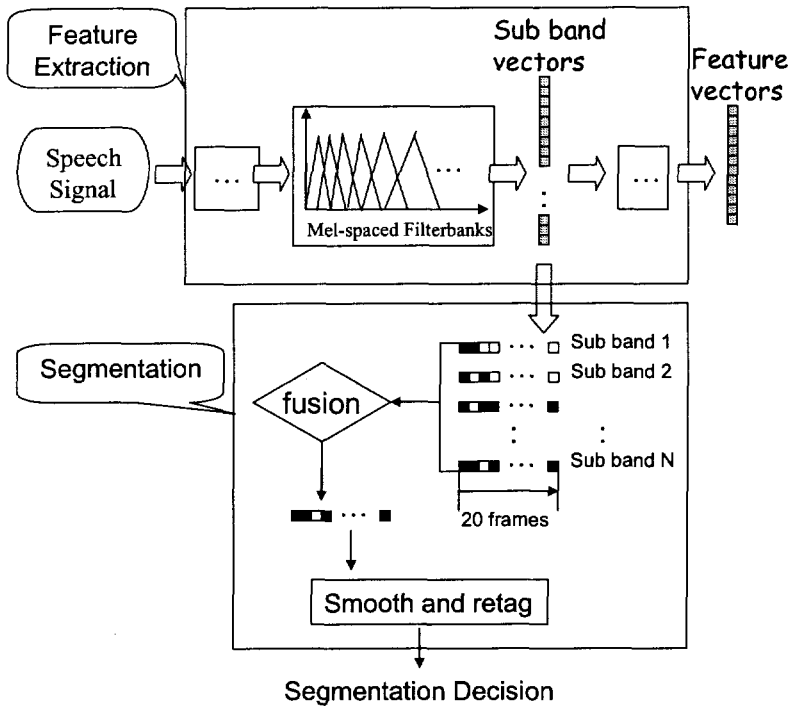


Figure 7.1: Subbands fusion based segmentation

bank corresponds to a subband of the speech spectrum. In the MFCC case, a certain number (we used 24 in our system) of mel scale triangular filters cover the whole frequency analysis spectrum. The filters have 50% overlap with their neighboring filters to obtain a smoothed frequency estimation. The magnitude coefficients in the SFFT spectrum are transformed into mel scale by correlating with these filter banks. The mel scale adopted in our system is

$$mel(f) = 1127 \cdot log(1 + \frac{f}{700}), \tag{7.1}$$

which is designed to normalize 1000Hz correspond to 1000 mels.

According to the equation above, 24 mel scale filter bank coefficients were calculated, which represent a weighted sum of the spectral magnitude in that subband.[1] These coefficients were combined into a single feature vector, and each coefficient describes part

[1] 1 subband $\Rightarrow$ 6 filter banks $\Rightarrow$ 3 MFCC coefficients. These coefficients are not independent, and some methods to decorrelate them may necessary.

of the information carried by the speech signal. In a traditional ASR system, the entire feature vector is used as one entity for training and classification. In our work, however, we treated each filter bank coefficient independently.

Using filter bank coefficients in speech segmentation has several advantages over the traditional energy based approach for detecting non-speech:

1. Each coefficient comes from the short-term spectral vector and represents the energy of the speech signal in a given frequency subband. The noise may corrupt some frequency bands but the majority of them are still useable. Based on this assumption, when majority filter bank coefficients drop to a local minimum, we can assume it is a possible non-speech frame.

2. The noise in SPINE task varies in type and distribution. Some are spread through the whole conversation but some only appear in speech or non-speech segments. They are also not just a simple additive noise that can be removed by spectral subtraction. The traditional energy based method treats the entire feature vector as a single entity; thus, noise is no different from speech in their contribution on energy calculation. Even a single noise corrupted subband spectral can falsely signal a non-speech event as a speech event.

Based on the analysis above, we designed a fusion based segmentation approach. The basic algorithm is as follows:

1. For frame t, obtain N filter bank coefficients from the normal feature extraction routine.

2. Form a fusion window l, which covers T consecutive frames and ends at frame t. Find the minimum filter bank coefficients for each filter bank i $(1 \leq i \leq N)$ within that window:

$$min_l(i) = \min_{t'=t-T+1}^{t} mel_{t'}(i) \tag{7.2}$$

Note: This is similar to the minimum statistic algorithm proposed by Martin [59], in which the minimum of smoothed power within a finite length window is used to estimate the noise power.

3. Fuse the statistical information of all the filter bank coefficients within window l for each frame $t^{'}$ $(t - T + 1 \leq t^{'} \leq t)$:

$$x_{t'}(i)(= \begin{cases} 1 & \text{if } (mel_{t'}(i) = min_l(i)) \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

$$num_{min}(t^{'}) = \sum_{i=1}^{N} x_{t'}(i) \tag{7.4}$$

Note: N is the number of filter banks and $num_{min}(t^{'})$ is the number of minimum filter bank coefficients occurring at frame $t^{'}$.

4. Tag each frame as speech or non-speech. First, compare $num_{min}(t^{'})$ with threshold Θ $(0 \leq \Theta \leq N)$, if

$$num_{min}(t^{'}) \geq \Theta, \tag{7.5}$$

then propose this frame $t^{'}$ as a potential non-speech point. It still has several possibilities considering its relative location to the speech segments:

   (a) Frame $t^{'}$ is the last frame of this l frames window. $\Rightarrow$ It is the start point of a non-speech segment following a speech segment.

   (b) Frame $t^{'}$ is the first frame of this l frames window. $\Rightarrow$ It is located at the end of a non-speech segment and is followed by a speech segment.

   (c) It could be located anywhere within the l frames window. $\Rightarrow$ It is a non-speech frame surrounded by other non-speech frames. It occurs when low level noise appeared inside a high level noise period.

We are more interested in conditions (a) and (b) because they separate the speech from non-speech segments. Condition (c) is less important because we have no intent to separate different noise events.

Scan from the first frame and tag each frame as speech or non-speech according to the neighboring tagging status and value of $num_{min}(t^{'})$.

5. Scan the tagging information from the first frame and re-tag each frame as speech or non-speech. In this step, the "speech/non-speech" tag of current frame t is re-determined by the neighboring tagging information and value of $num_{mel\_min}(t^{'})$.

6. In the following two steps, two re-scans are performed to smooth the segmentation decisions. Because both only involve the tagging information, the cost of performing both re-scans is negligible.

7. Re-scan the tagging information from the first frame and connect the neighboring short periods of speech together. Because in continuous speech, there are always small periods of non-speech parts existing between spoken words. We do not want to segment the whole input speech into individual words but rather separate it into utterances/sentences. Word level segmentation is not only more error prone but also loses the benefit of language model constraints.

   Similarly we connect the closely located non-speech parts together in this step.

8. Scan again starting from the first frame and produce segmented speech files according to the tagging result. We found our method is quite accurate at detecting the change of speech/non-speech events. We extend the speech duration by certain number of frames on both directions to append some silence. We would rather include a short period of silence than lose part of the speech events.

The raw audio data of SPINE2 evaluation has a total of 7 hours of (423 minutes) speech comprised of 128 unsegmented conversations with an average duration of 200 seconds. The overall number of words is 24,015.

The TRAPS segmentation was used in our official evaluation. After the evaluation, we tried CMU's segmentation and reported some results at the following SPINE2 workshop.

After the official evaluation, we conducted a series of experiments to compare these three segmentation algorithms. This is the first time that our fusion based approach is reported. Table 7.1 shows the number of files generated after the segmentations and their total file size.

The best way to evaluate a speech segmentation algorithm for a LVCSR task is to use its standard measurement: Word Error Rate (WER). In our experiments, we use the same recognizer with MFCC feature for all segmentation approaches. Table 7.2 measures the performance of segmentations by compare their corresponding WER. Our fusion based approach generates the least amount of speech data but results in the best recognition performance (Table 7.2). Further analysis shows the performance gain comes from:

1. Reduction on insertion error caused by noise. It measures the accuracy of excluding non-speech (including silence, noise, etc.) events.

2. Reduction on deletion errors caused by discarded speech. It measures the accuracy of tagging speech event.

Table 7.1: Comparison of three segmentation approaches on SPINE2 task: number of files and the overall files size

Segmentation	Number of files	Total files size
RAW[2]	64	779M
TRAPS	4591	315M
Gaussian Mixture	5682	315M
Subbands	5563	305M

The TRAPS based segmentation is also using multiple bands of the speech spectrum. However, it did not perform as well as our subband based approach. We speculate the following differences may be the reason. First, the MLPs used in the TRAPS approach were trained on all kinds of noise conditions; thus it is less accurate to a specific noise

[2]The raw speech data files contain both channels of conversation. Since there is only one participant is supposed to speak at one time, most of the time only one channel contains speech data. So roughly only half of the 779M data contains speech.

condition especially for the unseen testing noise conditions. Second, the TRAPS approach has a large amount of parameters in the MLPs which require sufficient training data and careful optimization. Third, TRAPS approach uses 15 critical bands on a down-sampled 8kHz speech while we use 24 bands on the original 16kHz speech. Another difference is that the TRAPS approach uses 101-frame window compared to our 8-frame window.

7.2.2 Fusion on Several Segmentations

Another level of fusion is achieved by combining the result from these three segmentations (Figure 7.2).

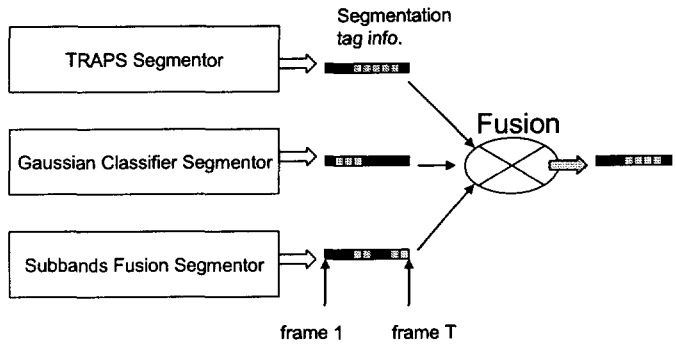


Figure 7.2: Fusion based segmentation: fusion across several segmentations

We tried several fusion methods here:

1. Majority Vote: The speech/non-speech tag of each frame is decided by the majority of segmentations.

2. Weighted Combination: A set of weights α_i and threshold τ are obtained from a development data set. The final tag is decided by comparing the threshold with the following value:

$$\sum_{i}^{N}(\alpha_i \cdot P_i(t)) \tag{7.6}$$

α_i is the weight value for segmentation i, $P_i(t)$ is the probability of frame t is speech estimated by segmentation i, in our case, due to lack of data from the other

two segmentations, $P_i(t)$ is a value of 0/1.

Table 7.2: Comparison of Segmentation Approaches on SPINE2 Task: Performance Measured by WER

Segmentation Approaches	WER
TRAPS	41.6%
Gaussian Mixture Classifier	39.3%
Subbands Fusion	38.4%
Majority Vote Fusion	38.2%
Weighted Combination Fusion	38.1%
Manual	37.8%

The effects of our fusion approaches are quite obvious. The weighted combination fusion achieves the lowest WER among automatic segmentations, which is only 0.3% higher than we obtained from a manual segmentation. Significance tests show these two systems have no statistical difference at the level of p=0.05, and they are both significantly better (at level p=0.001) than the TRAPS and Gaussian Mixture Classifier systems. As noted before, due to lack of data from the other two segmentations, $P_i(t)$ has a value of 0 or 1. Ideally a probability or confidence score can give a more reliable combined score.

Our fusion based segmentation approach has a clear advantage over others due to its simplicity and fast execution. The filter bank coefficients were already available from feature extraction and the additional calculation is negligible, so our approach can be easily integrated into the front end of an ASR system. The segmentation can be performed on-the-fly and the features can be generated right after it. However, TRAPS and Gaussian Mixture Classifier approaches, both require a recognition process and are much more complex and time demanding. They are fine with evaluation type tasks but not suitable for real applications that require a fast response.

Chapter 8

Conclusions and Future Work

In this thesis, we presented a run time fusion scheme to use different knowledge sources within a speech recognition system. In particular, we investigated the problems posed by the inefficient use of complementary information in speech recognition. Three methods of performing run time fusion were developed in conjunction with careful design and implementation. The techniques developed have been applied to a large vocabulary continuous speech recognition task and the performance analyzed both in terms of the computational complexity and the recognition accuracy.

8.1 Review of the Work

The traditional approach of fusion decouples the search stage with the fusion process; thus complementary information from multiple sources are either lost or used inefficiently by the decoder. The pre-recognition fusion methods, such as feature combination, require the multiple fusion sources to be independent which is generally not true. It is also difficult to identify complementary information directly from these sources. The combination at this level also causes the individual complementary information to be diluted by the more dominant redundant information. For post-recognition fusion, the complementary information is represented at the high level of a recognition architecture, such as the word or sentence level. The available information at these levels is much smaller compared to the whole search space. The complementary information from the original sources is

either lost or moved to a higher level. We have proposed a number of ways to address these problems. Specifically, our contributions are as follows.

- Our approach provides a novel way of fusing multiple information into an LVCSR system. Different feature representations are integrated into the decoding module of an LVCSR system. It provides the foundation for performing fusion at the run time of a recognizer.

  The decoding module is the kernel or CPU of a recognition system. Various types of information are presented and integrated at different levels within this module. Thus the decoding module is an ideal choice for information fusion.

- Implemented constraint fusion. We investigated the contribution of supporting features on reducing beam pruning. The result shows that conventional beam pruning works well for single feature decoding, while integrating several features into the pruning decision can improve the word graph quality. However, a likelihood value fusion is necessary to reduce WERs. We also proposed a dynamic beam adjustment approach: by comparing pruning effects across features, we were able to set a tight beam width without sacrificing accuracy.

- Implemented composite fusion. Different from constraint fusion, all features in the composite fusion approach have the same role on the search decision. The fusion concept is further integrated into the pruning decision. Significant improvement on word graph quality was observed and this improvement was also reflected in the WER reduction of second round decoding. Research shows evidence that different knowledge sources are not strictly time synchronous. To conduct asynchronous fusion across different features, we extended the token structure and proposed a modified Viterbi algorithm. By rewriting the token passing and merge procedure, we were able to perform fusion at selected levels (state, word and sentence). Experimental result shows the composite fusion achieved the best result among our fusion approaches.

- A great advantage of our run time fusion over other fusion methods is that the hypotheses are automatically aligned across different features. At time frame t, all (partial) hypotheses are generated from the same stretch of speech, thus the scores for different features can be directly compared. This comparison can be conducted at all levels of a speech recognizer: state, word, and sentence.

All experiments using our run-time fusion showed that the fusion based system outperforms single feature systems. The experiments proved that information fusion during the decoding phase not only can reduce pruning errors but is also able to select the better path. It was rather robust in all of our experiments and outperformed the baseline system in all cases.

8.2 Future Work

While the run time fusion approach described in this thesis gives significant improvement, there is room for further refinement. We think the following research directions might be useful in pursuing a robust speech recognition system under our run time fusion framework:

Finding canonical and complementary sets of information sources

This thesis work is mainly on investigating ways to use different signal processing and feature extraction techniques. Further work can be done on finding canonical and complementary sets of information sources. The possible complementary candidates include: processing techniques for additive and convolutional noises, features based on segments [Kingsbury, 1998] and frames, features from the spectral domains (such as MFCC) and time domains (such as pitch, duration, and stress). This work can be further extended to use fusion to guide the feature generation.

Detailed analysis of the complementary characteristic

In this thesis work, we selected some features for fusion based on their potential complementary natures. However this judgement is based on the analysis of feature design and experiment. More detailed analysis of the complementary characteristic of various features is necessary, such as the independence on the feature level and recognition error distribution on the state or phone level.

Extend this thesis work beyond acoustic information

1. Semantic decoding: how to incorporate semantic information into decoder run time to improve system performance in spoken dialogue systems.

2. Feature selection: for the operating environment of a given application, how to automatically select a set of complementary feature representations.

3. Multi-modal applications: how to exploit complementary information contained in different modalities during decoder run time.

Utilize aligned scores from multiple sources

Under the run time fusion framework, the hypotheses are automatically aligned across different features. At time frame t, all (partial) hypotheses are generated from the same stretch of speech, thus the scores for different features can be directly compared. This comparison can be conducted at all levels of a speech recognizer: state, word, and sentence. We obtained some success on using the rank information for fusion decision. The future work is to explore efficient statistical algorithms to use these aligned scores. The scores or ranks from multiple features are comparable under our run-time fusion framework, thus are suitable for confidence measuring and word spotting.

Appendix A

Classes used in our Class Based Language Model

Table A.1: Class Language Model: x-axis of ACE Grid Labels in SPINE2 Lexicon

Word Name	Probability in the Language Model Train Text
alpha	4.087714e-04
alfa	3.891240e-02
bravo	6.699896e-02
charlie	5.091183e-02
delta	3.363793e-02
echo	4.168150e-02
foxtrot	2.638554e-02
golf	1.702335e-02
hotel	2.282527e-02
india	3.311049e-02
juliett	2.308899e-02
kaybeck	3.930799e-02
kilo	1.425426e-02
lima	3.205559e-02
mike	3.020953e-02
november	4.247267e-02
oscar	5.249416e-02
papa	4.537363e-02
romeo	4.497804e-02
sierra	4.853831e-02
tango	6.317497e-02
uniform	2.928650e-02
victor	3.746193e-02
whiskey	2.084734e-02
x-ray	3.218745e-02
yankee	4.946135e-02
zulu	6.291124e-02

Table A.2: Class Language Model: y-axis of ACE Grid Labels in SPINE2 Lexicon

Word Name	Probability in the Language Model Train Text
abort	9.102723e-03
above	3.060230e-03
affirm	6.633747e-03
aft	6.958612e-03
anchor	3.644987e-03
away	9.102723e-03
bingo	3.255149e-03
blast	1.370931e-03
bogey	2.345527e-03
bow	8.128127e-03
break	7.088558e-03
broken	7.413424e-03
buster	4.294718e-03
clear	3.255149e-03
code	9.362615e-03
copy	3.255149e-03
crew	1.240985e-03
danger	5.399259e-03
deck	6.113963e-03
decoy	6.958612e-03
ditch	6.308882e-03
divert	3.255149e-03
drive	6.048989e-03
engine	5.854071e-03
foe	6.503801e-03
...	...
wave	5.789097e-03

Table A.3: Class Language Model: Class of Directions in SPINE2 Lexicon

Word Name	Probability in the Language Model Train Text
north	1.339549e-01
south	1.436749e-01
east	1.548348e-01
west	1.051552e-01
northwest	1.011952e-01
northeast	1.058751e-01
southwest	1.137951e-01
southeast	1.415149e-01

Table A.4: Class Language Model: Class of Person Name in SPINE2 Lexicon

Word Name	Probability in the Language Model Train Text
angela	1.831069e-02
brandon	3.603071e-02
david	1.069108e-01
dawn	1.831069e-02
fox	1.187242e-01
gloria	3.603071e-02
houston	3.603071e-02
liz	1.069108e-01
maggie	1.831069e-02
maverick	5.375074e-02
michael	3.431778e-01
pam	3.603071e-02
sarah	7.147076e-02

Table A.5: Class Language Model: Class of Partial Words in SPINE2 Lexicon

Word Name	Probability in the Language Model Train Text
a-	3.348169e-02
ac-	3.135748e-03
acou-	3.135748e-03
acoust-	3.135748e-03
an-	1.527413e-02
ar-	6.170342e-03
b-	1.223953e-02
ba-	1.011531e-04
bo-	1.011531e-04
c-	4.663160e-02
ca-	1.527413e-02
ch-	8.193404e-03
co-	1.325106e-02
con-	4.460854e-02
conf-	2.134331e-02
confi-	1.011531e-04
confir-	1.830872e-02
coo-	1.011531e-04
coor-	6.170342e-03
coord-	3.135748e-03
cor-	1.011531e-04
...	...
z-	6.170342e-03

Appendix B

Significance Test

In the speech recognition community, it is necessary to compare the performance of two recognition systems when reporting any experimental results. Researchers need to make comparison tests to claim workable, novel, or improved algorithms. Most publications still use a single criterion to measure the performance. This criterion, Word Error Rate (WER), measures the overall word level agreement between the recognized words and the reference by aligning each sentence pair using a dynamic programming (DP) algorithm. The WER is calculated by averaging the overall substitutions (*SUB*), deletions (*DEL*) and insertions (*INS*) over the total number of reference words:

$$WER = \frac{SUB + DEL + INS}{Num\_of\_Words} \tag{B.1}$$

The difference of two experimental results is further measured by the absolute WER difference

$$\Delta WER = WER_{new} - WER_{baseline} \tag{B.2}$$

and relative WER difference

$$\Delta WER\% = \frac{WER_{new} - WER_{baseline}}{WER_{baseline}} \tag{B.3}$$

WER is a good performance measurement of a ASR system but generally it is not sufficient. For an extreme example, if a test set consists of only one reference word, the system recognized correctly has a 0% WER but another system which made a mis-recognition has a WER of 100%. The results on this tiny one word test set by no means

give any sureness on the performance of a ASR system. The shifting from 0% to 100% WERs can well be the result of variability and uncertainly of the test data. WER, is not sufficient to back any difference in performances as statistically significant. To measure the efficiency of algorithms and to compare the performance of different ASR systems, it is important to measure the statistical significance of the difference in experimental results.

A significance test is a test for determining the probability that a given result could not have occurred by chance. 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.

A set of tools for performing significant tests was originally developed by Pallett, et al. at NIST for DARPA speech recognition benchmark tests [71] and is now included in the NIST standard scoring package. Some of the significance tests we used in this thesis are briefly explained below.

B.1 Signed Pair Comparison Test

The Signed Pair Comparison Test or sign test, is a test comparing word error rates on:

- different speakers,
- different conversation sides,
- some pre-specified subsets of a test set.

It measures which system performs better on each such subset. If there is systematic evidence of differences in a consistent direction, this may prove to be significant even if the magnitudes of the differences are small.

The null hypothesis:

The number of speakers for which the differences is positive equals the number of speakers for which the differences is negative.

The alternative hypothesis:

The number of speakers for which the differences is positive is NOT equal to the number of speakers for which the difference is negative.

Signed Pair Comparison Test is based on the following assumptions that the distribution of positive and negative differences follows the binomial distribution for N fair coin tosses. It measures the systematic evidence of differences in a consistent direction and ignores the magnitudes of these differences.

The sign test is simple and easy to use, and has been regularly used by NIST in its organized evaluations since 1992. The disadvantage of sign test is it is less powerful compared to other tests.[1]

B.2 Wilcoxon Signed Rank Test

The Wilcoxon signed rank test applies in similar evaluation situations as the sign test and is generally more powerful. The Wilcoxon signed rank test is a non-parametric test[2] that utilizes information on both the signs and the magnitudes of the performance differences in two systems. The implementation used in this thesis 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]The power of a statistical hypothesis test measures the test's ability to reject the null hypothesis when it is actually false - that is, to make a correct decision.

[2]Non-Parametric tests are often used in place of their parametric counterparts when certain assumptions about the underlying population are questionable. Non-Parametric tests may be, and often are, more powerful in detecting population differences when certain assumptions are not satisfied. All tests involving ranked data, i.e. data that can be put in order, are non-parametric.

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

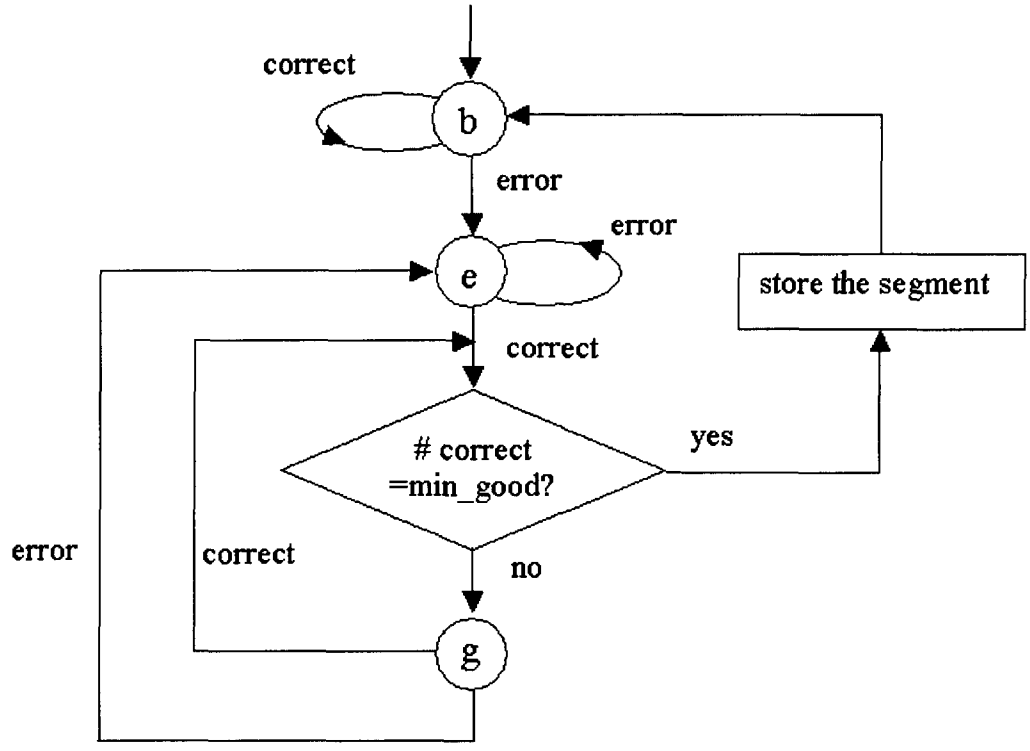
$$z = \frac{T_+ - \mu}{\sigma}$$

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 $|z| > 1.96$.

B.3 MAPSSWE Test

Matched Pairs Sentence Segment Word Error (MAPSSWE) test, sometimes simply called matched-pairs test, is a parametric test that looks at the numbers of errors occurring in units that are larger than single words and smaller than entire utterances [35] [36]. 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 in Figure B.3.



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 B.1: 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.

An example of detected sentence segments is shown below. There are four segments

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

	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

For each segment i, define d_i as the difference of the number of mis-recognized 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, $\bar{d} = \sum_{i=1}^{n} d_i/n$, and 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 [60] 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.

B.4 McNemar (Sentence Error) Test

McNemar Test applies to discrete items which are either correct or incorrect. Similar as matched-pairs test, the McNemar test requires those items be independent. For continuous

speech, utterances can be viewed as either correct or incorrect and they do qualify the independence test. But the words in the utterances are not a suitable items for McNemar Test because the violation of independent assumption. The reasons are:

1. Language model (such as bigram, trigram) applied among words.
2. Current speech recognition algorithm performs an optimization at sentence level.
3. The scoring program use DP based algorithm for string comparison.

The errors are therefore highly inter-dependent at the word level. But if each spoken phrase is reasonably short (a few words), it can be treated as an independent item, thus the McNemar Test can be applied.

The McNemar Test is based on an Error Matrix (Table B.1).

Table B.1: McNemar Test Error Matrix: Counts of correct and incorrect items for two systems.

#Items	System B Correct	System B Incorrect
System A Correct	N_{00}	N_{01}
System A Incorrect	N_{10}	N_{11}

The assumptions applied to the Table B.1 are:

- There is little information in the numbers of instances for which: Both systems under consideration get correct results (N_{00}). Both get an incorrect results (N_{11}).
- N_{00} and N_{11} are due to excessively easy or excessively difficult items.

Based on these assumptions, only N_{01} and N_{10} are actually used in McNemar test. The hypotheses of the test are as follows. **The null hypothesis:**

$$H_0^1 : q_{01} = q10 \tag{B.4}$$

$$q = \frac{q_{01}}{(q_{01} + q10)} \tag{B.5}$$

The alternative hypothesis:

$$H_0^2 : q = \frac{1}{2} \tag{B.6}$$

The q_{01} represents the conditional probability that System B will make an error on an utterance given that only one of the two algorithms makes an error.

The test statistic used in this thesis is distributed approximately as chi-squared with 1 degree of freedom

$$Z = \frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}} \tag{B.7}$$

Based on a 95% confidence interval, the Z-score should be greater than $\chi^2_{1,0.95} = 3.842$.

The McNemar Test can be viewed as the sign test applied at the utterance level. It can be applied without making many unsound assumptions. For example, for some experiments with small error rates, there are often too few relevant observations to apply parametric tests.

B.5 Significant Test Summary

Some of the assumptions (Table B.2) required for these tests (e.g. independence of errors and the availability of sufficient errors to justify assumptions about distributions) may not be satisfied for some experiments. We may choose to use the sign test over a more powerful competitor because of its ease of application (Table B.4). If null hypothesis was rejected, we are done, otherwise, we may use a more powerful test or increase the sample size.

Choosing between Parametric and Non-parametric tests (Table B.4) depends on the

Table B.2: Significance Tests Comparison: Test Assumptions

Test Name	Assumptions
Signed Paired Test	None other than H_0
Wilcoxon Signed Rank Test	Difference has symmetric distribution
McNemar	Independence
Matched Pair Test	Independence; normal distribution

Table B.3: Significance Tests Comparison: Test Units

Test Name	Test Units
Signed Paired Test	Speaker Word Accuracy Rate
Wilcoxon Signed Rank Test	Speaker Word Accuracy Rate
McNemar	Sentence Error
Matched Pair Test	Word Error

sample size. For large sample size, both tests are powerful enough if their assumptions are (at least approximately) satisfied. When not enough samples available, both tests may be inaccurate, especially non-parametric tests, which lack statistical power.

Table B.4: Significance Tests Comparison: Whether it is a parametric test and its relative power

Test Name	Parametric Test	Power
Signed Paired Test	no	small
Wilcoxon Signed Rank Test	no	large
McNemar	no	large
Matched Pair Test	yes	largest

Bibliography

[1] ALLEN, J. B. How do humans process and recognize speech? *IEEE Transactions on Acoustics, Speech, and Signal Processing, 2*, 4 (1994), 567–577.

[2] 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* (Munich, Germany, Apr. 1997), pp. 1043–1046.

[3] ANASTASAKOS, T., MCDONOUGH, J., SCHWARZ, R., AND MAKHOUL, J. A compact model for speaker adaptive training. In *Proceedings of Fourth International Conference on Spoken Language Processing* (Philadelphia, PA, Jan. 1996), pp. 1137–1140.

[4] AUBERT, X., AND THELEN, E. Speaker adaptive training applied to continuous mixture density modeling. In *Proceedings of Fifth European Conference on Speech Communication and Technology* (Rhodes, Greece, Sep. 1997), pp. 1851–1854.

[5] AUBERT, X. L. An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech and Language 16* (2002), 89–114.

[6] BAHL, L. R., BROWN, P. F., 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.

[7] 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.

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

[9] 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.

[10] BESAG, J. Spatial interaction and the statistical analysis of lattice systems. *Journal of Royal Statistical Society 36* (1974), 192–236.

[11] BOURLARD, H., AND DUPONT, S. A new *ASR* approach based on independent processing and recombination of partial frequency bands. In *Proceedings of Fourth International Conference on Spoken Language Processing* (Philadelphia, PA, Jan. 1996), pp. 426–429.

[12] BOURLARD, H., AND DUPONT, S. Subband-based speech recognition. In *Proceedings of the 1997 International Conference on Acoustics, Speech, and Signal Processing* (Munich, Germany, Apr. 1997), vol. 2, pp. 1251–1254.

[13] BOURLARD, H., DUPONT, S., AND RIS, C. Multi-stream speech recognition. Idiap-rr 96-07, IDIAP, 1996.

[14] BOURLARD, H., HERMANSKY, H., AND MORGAN, N. Towards increasing speech recognition error rates. *Speech Communication 18* (1996), 205–231.

[15] BRAND, M., OLIVER, N., AND PENTLAND, A. Coupled hidden Markov models for complex action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (Puerto Rico, Jun. 1997), pp. 994–999.

[16] BREIMAN, L., FRIEDMAN, J., OLSHEN, R., AND STONE, C. *Classification and Regression Trees*. Wadsworth & Brooks, 1984.

[17] CERISARA, C., AND FOHR, D. Multi-band automatic speech recognition. *Computer Speech and Language 15*, 2 (2001), 151–174.

[18] CERISARA, C., HATON, J. P., MARI, J. F., AND FOHR, D. A recombination model for multi-band speech recognition. In *Proceedings of the 1998 International Conference on Acoustics, Speech, and Signal Processing* (Seattle, Washington, May 1998), pp. 717–720.

[19] CHEN, S., AND GOPALAKRISHNAN, P. S. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Broadcast News Transcription and Understanding Workshop* (Lansdowne, Virginia, Feb. 1998), pp. 127–132.

[20] CHOU, K., WILLSKY, A., AND BENVENISTE, A. Multiscale recursive estimation, data fusion, and regularization. *IEEE Transactions on Automatic Control, 39* (1994), 464–491.

[21] CHRISTENSEN, H., LINDBERG, B., AND ETC. Employing heterogeneous information in a multi-stream framework. In *Proceedings of the 2000 International Conference on Acoustics, Speech, and Signal Processing* (Istanbul, Turkey, Jun. 2000), vol. 3, pp. 1571–1574.

[22] 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.

[23] DEMPSTER, A. D., LAIRD, N. M., AND RUBIN, D. B. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of Royal Statistics Society, Series B 39* (1977), 1–38.

[24] DENES, P. D., AND PINSON, E. N. *The Speech Chain: The physics and biology of spoken language*. W.H. Freeman and Company, New York, 1993.

[25] 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.

[26] EIDEN, E., AND GISH, H. A parametric approach to vocal tract length normalization. In *Proceedings of the 1996 International Conference on Acoustics, Speech, and Signal Processing* (Atlanta,Georgia, May 1996), pp. 346–349.

[27] ELLIS, D. W. P., SINGH, R., AND SIVADAS, S. Tandem acoustic modeling in large-vocabulary recognition. In *Proceedings of the 2001 International Conference on Acoustics, Speech, and Signal Processing* (Salt Lake City, Utah, May 2001), vol. 1, pp. 517–520.

[28] FISCUS, J. G. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction *(ROVER)*. In *IEEE Workshop on Automatic Speech Recognition and Understanding* (Santa Barbara, California, Dec. 1997), pp. 347–354.

[29] FLETCHER, H. *Speech and Hearing in Communication*. Robert E. Krieger Publishing Company, Huntington, New York, 1953.

[30] FURUI, S. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing 29*, 2 (1981), 254–272.

[31] FURUI, S. Speaker-independent isolated word recognition using dynamic features of speech spectrum. In *Proceedings of the 1986 International Conference on Acoustics, Speech, and Signal Processing* (Tokyo, Japan, Apr. 1986), pp. 52–59.

[32] GADDE, R., STOLCKE, A., VERGYRI, D., ZHENG, J., SONMEZ, K., AND VENKATARAMAN, A. Building an *ASR* system for noisy environments:*SRI*'s 2001 *SPINE* evaluation system. In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver, Colorado, Sep. 2002), vol. 3, pp. 1577–1580.

[33] 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.

[34] GHAHRAMANI, Z., AND JORDAN, M. Factorial hidden Markov models. In *Advances in Neural Information Processing Systems* (Denver, Colorado, Apr. 1995), vol. 8, pp. 472–478.

[35] GILLICK, L., AND COX, S. J. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing* (Glasgow, Scotland, May 1989), pp. 532–535.

[36] 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* (Alburquerque, New Mexico, Apr. 1990), pp. 97–100.

[37] GREENBERG, S. Understanding speech understanding: Towards a unified theory of speech perception. In *ESCA Workshop on The Auditory Basis of Speech Perception* (Keele University, UK, Jul. 1996), pp. 1–8.

[38] HAEB-UMBACH, R., AND NEY, H. Improvements in time-synchronous beam search for 10,000 word continuous speech recognition. *IEEE Transactions on Speech and Audio Processing 2* (Apr. 1994), 353–356.

[39] HERMANSKY, H. Perceptuanl linear predictive (*PLP*) analysis of speech. *Journal of the Acoustical Society of America, 87* (1990), 1738–1752.

[40] HERMANSKY, H., ELLIS, D. P. W., AND SHARMA, S. Tandem connectionist feature extraction for conventional *HMM* systems. In *Proceedings of the 2000 International Conference on Acoustics, Speech, and Signal Processing* (Istanbul, Turkey, Jun. 2000), vol. 3, pp. 1635–1638.

[41] HERMANSKY, H., AND SHARMA, S. *TRAPS* - classifiers of temporal patterns. In *Proceedings of Fifth International Conference on Spoken Language Processing* (Sydney, Australia, Nov. 1998), vol. 3, pp. 1003–1006.

[42] HERMANSKY, H., TIBREWALA, S., AND PAVEL, M. Towards *ASR* on partially corrupted speech. In *Proceedings of Fourth International Conference on Spoken Language Processing* (Philadephia, PA, Oct. 1996), pp. 458–461.

[43] HON, H. W., AND LEE, K. F. Recent progress in robust vocabulary-independent speech recognition. In *Fourth DARPA Speech and Natural Language Workshop* (Pacific Grove, CA, Feb. 1991), pp. 258–263.

[44] HUANG, X., ACERO, A., AND HON, H.-W. *Spoken Language Processing: a guide to theory, algorithm, and system development.* Prentice Hall PTR, Upper Saddle River, N.J., 2001.

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

[46] HWANG, M. Y., AND HUANG, X. Shared distribution hidden markov models for speech recognition. *IEEE Transactions on Speech and Audio Processing 1* (1993), 414–420.

[47] JANIN, A., ELLIS, D., AND MORGAN, N. Multi-stream speech recognition: Ready for prime time? In *Proceedings of Sixth European Conference on Speech Communication and Technology* (Budapest, Hungary, Sep. 1999), vol. 2, pp. 591–594.

[48] KAJAREKAR, S., YEGNANARAYANA, B., AND HERMANSKY, H. A study of two dimensional linear discriminants for *ASR*. In *Proceedings of the 2001 International Conference on Acoustics, Speech, and Signal Processing* (Salt Lake City, Utah, May 2001), pp. 137–140.

[49] KINGSBURY, B., JAIN, P., AND ADAMI, A. A hybrid *HMM/TRAPS* model for robust voice activity detection. In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver, Colorado, Sep. 2002), pp. 1073–1076.

[50] 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* (Rhodes, Greece, Sep. 1997), pp. 19–22.

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

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

[53] LIPPMANN, R. P. Speech recognition by machines and humans. *Speech Communication 22*, 1 (1997), 1–16.

[54] LIU, C. *Toward More Effective Acoustic Model Clustering by More Efficient Use of Data in Speech Recognition*. PhD thesis, OGI School of Science & Engineering at Oregon Health & Science University, 2002.

[55] LOWERRE, B., AND REDDY, R. The harpy speech understanding system. In *Trends in Speech Recognition*. Prentice Hall, 1980, pp. 340–360.

[56] MANGU, L., BRILL, E., AND STOLCKE, A. Finding consensus among words: Lattice-based word error minimization. In *Proceedings of Sixth European Conference on Speech Communication and Technology* (Budapest, Hungary, Sep. 1999), pp. 495–498.

[57] MANGU, L. L. *Finding consensus in speech recognition*. PhD thesis, Johns Hopkins University, 2000.

[58] MARKEL, J. D., AND GRAY, A. H. *Linear Prediction of Speech*. Springer-Verlag, 1976.

[59] MARTIN, R. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Trans. Speech and Audio Processing Vol. 9*, 5 (Jul. 2001), 504–512.

[60] MASON, R. L., GUNST, R. F., AND HESS, J. L. *Statistical Design and Analysis of Experiments*. John Wiley & Sons Inc, 1989.

[61] MIRGHAFORI, N., AND MORGAN, N. Combining connectionist multi-band and full-band probability streams for speech recognition of natural numbers. In *Proceedings of Fifth International Conference on Spoken Language Processing* (Sydney, Australia, Nov. 1998), pp. 743–746.

[62] MIRGHAFORI, N., AND MORGAN, N. Sooner or later: Exploring asynchrony in multi-band speech recognition. In *Proceedings of Sixth European Conference on Speech Communication and Technology* (Budapest, Hungary, Sep. 1999), vol. 2, pp. 595–598.

[63] NEY, H., HAEB-UMBACH, R., TRAN, B. H., AND OERDER, M. Improvement in beam search for 10,000 word continuous speech recognition. In *Proceedings of the 1992 International Conference on Acoustics, Speech, and Signal Processing* (San Francisco, California, Apr. 1992), pp. 9–12.

[64] NGUYEN, L., AND SCHWARTZ, R. The bbn singlephonetic-tree fast-match algorithm. In *Proceedings of Fifth International Conference on Spoken Language Processing* (Sydney, Australia, Nov. 1998), pp. 1827–1830.

[65] NOCK, H. J., AND OSTENDORF, M. Parameter reduction schemes for loosely coupled hmms. *Computer Speech and Language 17*, 2-3 (2003), 233–262.

[66] NRL. http://elazar.itd.nrl.navy.mil/spine. In *The Second Speech in Noisy Environments Evaluation and Workshop* (2001).

[67] Ö.ÇETIN, NOCK, H., KIRCHHOFF, K., BILMES, J., , AND OSTENDORF, M. The 2001 *GMTK*-based *SPINE ASR* system. In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver, Colorado, Sep. 2002), pp. 1037–1040.

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

[69] ODELL, J. J., VALTCHEV, V., WOODLAND, P. C., AND YOUNG, S. J. A one pass decoder design for large vocabulary recognition. In *Proceedings of the 1993 International Conference on Acoustics, Speech, and Signal Processing* (Minneapolis, Minnesota, Apr. 1993), pp. 405–410.

[70] ORTMANNS, S., AND NEY, H. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language 11* (1997), 43–72.

[71] PALLETT, D., FISCUS, J., FISHER, W., AND GAROFOLO, J. Benchmark tests for the *DARPA* spoken language program. In *Human Language Technology Workshop* (Princeton, N.J., Mar. 1993), M.Bates, Ed., pp. 7–18.

[72] PAUL, D. B. Experience with a stack decoder-based *HMM CSR* and back-off n-gram language models. In *Fourth DARPA Speech and Natural Language Workshop* (Pacific Grove, California, Feb. 1991), pp. 284–288.

[73] PELLOM, B., AND HACIOGLU, K. Recent improvements in the *CU SONIC ASR* system for noisy speech: The *SPINE* task. In *Proceedings of the 2003 International Conference on Acoustics, Speech, and Signal Processing* (HongKong, China, Apr. 2003), vol. 1, pp. 4–7.

[74] POLYMENAKOS, L., OLSEN, P., KANVESKY, D., GOPINATH, R. A., GOPALAKRISHNAN, P. S., 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* (Seattle, Washington, May 1998), pp. 901–904.

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

[76] RAJ, B., AND SINGH, R. Classifier-based non-linear projection for adaptive end-pointing of continuous speech. *Computer Speech and Language 17*, 1 (2003), 5–26.

[77] RAVISHANKAR, M. K. *Efficient Algorithms for Speech Recognition*. PhD thesis, Carnegie Mellon University, 1996.

[78] SAVAGE, L. J. *The Foundations of Statistical Inference*. New York: Weley (Methuen & Co. London), 1962.

[79] 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* (San Diego, California, Mar. 1984), pp. 35.6.1–35.6.4.

[80] SILVADAS, S., AND HERMANSKY, H. Hierarchical tandem feature extraction. In *Proceedings of the 2002 International Conference on Acoustics, Speech, and Signal Processing* (Orlando, Florida, May 2002), pp. 809–812.

[81] SINGH, R., RAJ, B., AND STERN, R. M. 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* (Phoenix, Arizona, Mar. 1999), pp. 117–120.

[82] SINGH, R., SELTZER, M. L., RAJ, B., AND STERN, R. M. Speech in noisy environment: Robust automatic segmentation, feature extraction and hypothesis combination. In *Proceedings of the 2001 International Conference on Acoustics, Speech, and Signal Processing* (Salt Lake City, Utah, May 2001), vol. 1, pp. 273–276.

[83] TIBREWALA, S., AND HERMANSKY, H. Sub-band based recognition of noisy speech. In *Proceedings of the 1997 International Conference on Acoustics, Speech, and Signal Processing* (Munich, Germany, Apr. 1997), vol. 2, pp. 1255–1258.

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

[85] WEINTRAUB, M., TAUSSIG, K., HUNICKE-SMITH, K., AND SNODGRASS, A. Effect of speaking style on *LVCSR* performance. In *Proceedings of Fourth International Conference on Spoken Language Processing* (Philadephia, PA, Oct. 1996), pp. 16–19.

[86] WILEY, J., AND SONS. *Bayesian Statistics: Principles, Models, and Applications*. John Wiley & Sons, Inc., 1989.

[87] WOODLAND, P. C., LEGGETTER, C. J., ODELL, J. J., VALTCHEV, V., AND YONG, S. J. The 1994 *HTK* large vocabulary speech recognition system. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing* (Detroit, Michigan, Apr. 1995), pp. 73–76.

[88] 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* (Adelaide, Australia, Apr. 1994), pp. 125–128.

[89] WU, X. *Knowledge Constraints in Speaker Adaptation*. PhD thesis, Oregon Graduate Institute of Science and Technology, 2000.

[90] WU, X., LIU, C., YAN, Y., KIM, D., CAMERON, S., AND PARR, R. The 1998 *OGI-FONIX* broadcast news transcription system. In *Broadcast News Transcription and Understanding Workshop* (Herndon, Virginia, Feb. 1999).

[91] YAN, Y., LIU, C., AND ZHENG, C. A multiple feature front-end approach to speech in noise. In *International Conference on Signal and Image Processing 2002* (Kauai, Hawaii, Aug. 2002).

[92] YAN, Y., WU, X., SCHALKWYK, J., AND COLE, R. Development of *CSLU LVCSR*: the 1997 *DARPA Hub4* evaluation system. In *Broadcast News Transcription and Understanding Workshop* (Lansdowne, Virginia, Feb. 1998).

[93] YAN, Y., ZHENG, C., ZHANG, J., PAN, J., HAN, J., AND LIU, J. A dynamic cross-reference pruning strategy for multiple feature fusion at decoder run time.

In *Proceedings of Eighth European Conference on Speech Communication and Technology* (Geneva, Switzerland, Sep. 2003), pp. 1177–1180.

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

[95] YOUNG, S., RUSSEL, N., AND THORNTON, J. Token passing: A simple conceptual model for connected speech recognition systems. Tech. Rep. TR38, Cambridge University Engineering Department, 1989.

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

[97] ZAVALIAGKOS, G. *Maximum A Posteriori Adaptation Techniques for Speech Recognition.* PhD thesis, Northeastern University, 1995.

[98] ZHANG, J., AND NAKAMURA, S. Modeling varying pauses to develop robust acoustic models for recognizing noisy conversational speech. In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver, Colorado, Sep. 2002), vol. 4, pp. 2601–2604.

[99] ZHENG, C., AND YAN, Y. Efficiently using speaker adaptation data. In *Proceedings of Sixth International Conference on Spoken Language Processing* (BeiJing, China, Oct. 2000), vol. 4, pp. 358–361.

[100] ZHENG, C., AND YAN, Y. Improving speaker adaptation by adjusting the adaptation data set. In *2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems* (Honolulu, Hawaii, Nov. 2000), pp. 718–721.

[101] ZHENG, C., AND YAN, Y. Run time information fusion in speech recognition. In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver, Colorado, Sep. 2002), pp. 1077–1080.

Biographical Note

Chengyi Zheng was born in ZheJiang, China on June 16th, 1973. He attended Shanghai JiaoTong University (SJTU) from 1991 to 1995, and obtained his Bachelor Degree in Information System and Instrument Engineering in 1995. He graduated with outstanding student honor and was waived from the national graduate entrance exam. He selected another prestigious university in China, Fudan University, for his graduate study. He started working on speech recognition as a graduate student of the computer science department. He obtained his Master Degree of Computer Science in 1998.

From 1998 to 2004, Chengyi Zheng pursued his Ph.D. degree in Computer Science and Engineering Department at the OGI School of Science & Engineering, which is part of the Oregon Health & Science University (formerly Oregon Graduate Institute of Science and Technology). His research focus was speech recognition. As part of a small team led by Dr. Yonghong Yan, Chengyi Zheng participated in several DARPA sponsored worldwide competitions. Beginning with the 1998 DARPA Broadcast News, in 2001 the team obtained the 3rd place on the NIST SPeech In Noise Environments test (SPINE) under the common language model track. In 2003 the team obtained the 3rd place in NIST Language Recognition Evaluation. His major contributions lie in improving the performance of current Large Vocabulary Continuous Speech Recognition system. He has worked on various aspects of the speech recognition system, from speech segmentation, feature extraction, acoustic and language model training and speaker adaptation to search engine optimization.

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

List of Publications:

- ZHENG, C., LIU, X. AND LI, Z., A Chinese Speech Database for Network Service, In *Proceedings of ORIENTAL COCOSDA Workshop 1998* (Tsukuba, Japan, May 1998).
- ZHENG, C., XU, Z., Sign Language Recognition System Using Image Processing, In Journal of Computer Engineering & Application (1998, China), 1998 annual edition.
- XU, Z., ZHENG, C., YE, Z., XIE, M., Complex-valued Multistate Bidirectional Associative Memory, In *Journal of Acta Electronica Sinica*, (1999), Vol 27, 118–120.
- ZHENG, C., AND YAN, Y., Efficiently Using Speaker Adaptation Data, In *Proceedings of Sixth International Conference on Spoken Language Processing* (BeiJing, China, Oct 2000).
- ZHENG, C., AND YAN, Y., Improving Speaking Adaptation by Adjusting the Adaptation Data Set, In *Proceedings of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems* (Honolulu, Hawaii, Nov 2000).
- YAN, Y., LIU, C., AND ZHENG, C., A Multiple Feature Front-end Approach to Speech in Noise, In *Proceedings of International Conference on Signal and Image Processing 2002* (Kauai, Hawaii, Aug 2002).
- ZHENG, C., AND YAN, Y., Run Time Information Fusion in Speech Recognition, In *Proceedings of Seventh International Conference on Spoken Language Processing* (Denver,Colorado, Sep 2002).
- YAN, Y., ZHENG, C., AND ETC., A Dynamic Cross-reference Pruning Strategy for Multiple Feature Fusion at Decoder Run Time, In *Proceedings of Eighth European*

Conference on Speech Communication and Technology (Geneva, Switzerland, Sep 2003).

- ZHENG, C., AND YAN, Y., Fusion Based Speech Segmentation in DARPA SPINE2 Task, In *Proceedings of the 2004 International Conference on Acoustics, Speech, and Signal Processing* (Montreal, Canada, May 2004).

- ZHENG, C., AND YAN, Y., Information Fusion in Large Vocabulary Continuous Speech Recognition, *Submitted for publication*.