Unsupervised Spoken Keyword Spotting and Learning of Acoustically Meaningful Units

by

Yaodong Zhang

B.E. in Computer Science, Shanghai Jiao Tong University (2006)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .....
Department of Electrical Engineering and Computer Science
September 4, 2009

Certified by .....
James R. Glass
Principle Research Scientist
Thesis Supervisor

Accepted by .....
Terry Orlando
Chairman, Department Committee on Graduate Theses

Unsupervised Spoken Keyword Spotting and Learning of Acoustically Meaningful Units

by

Yaodong Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on September 4, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

The problem of keyword spotting in audio data has been explored for many years. Typically researchers use supervised methods to train statistical models to detect keyword instances. However, such supervised methods require large quantities of annotated data that is unlikely to be available for the majority of languages in the world. This thesis addresses this lack-of-annotation problem and presents two completely unsupervised spoken keyword spotting systems that do not require any transcribed data.

In the first system, a Gaussian Mixture Model is trained to label speech frames with a Gaussian posteriorgram, without any transcription information. Given several spoken samples of a keyword, a segmental dynamic time warping is used to compare the Gaussian posteriorgrams between keyword samples and test utterances. The keyword detection result is then obtained by ranking the distortion scores of all the test utterances.

In the second system, to avoid the need for spoken samples, a Joint-Multigram model is used to build a mapping from the keyword text samples to the Gaussian component indices. A keyword instance in the test data can be detected by calculating the similarity score of the Gaussian component index sequences between keyword samples and test utterances.

The proposed two systems are evaluated on the TIMIT and MIT Lecture corpus. The result demonstrates the viability and effectiveness of the two systems. Furthermore, encouraged by the success of using unsupervised methods to perform keyword spotting, we present some preliminary investigation on the unsupervised detection of acoustically meaningful units in speech.

Thesis Supervisor: James R. Glass
Title: Principle Research Scientist

Acknowledgments

I would like to thank my advisor, James Glass, for his encouragement, patience and every discussion that guided me through the research in this thesis. In addition, I would like to thank T. J. Hazen for helpful discussions and insightful comment about this research.

The research experiments in this thesis were made possible by assistantships provided by the entire Spoken Language Systems group.

This research was sponsored in part by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Finally, I would certainly like to thank my parents, who always give constant love and support to me.

Contents

1 Introduction **15**

1.1 Motivation 16

1.2 Overview of Chapters 17

2 Related Work **19**

2.1 Supervised Keyword Spotting Systems 19

2.1.1 HMM Based Methods 20

2.1.2 Lattice Alignment Based Methods 21

2.2 Unsupervised Keyword Spotting Systems 22

2.2.1 Ergodic HMM 22

2.2.2 Segmental GMM 23

2.2.3 Phonetic Posteriorgram Templates 24

2.3 Unsupervised Learning in Speech 25

2.3.1 Detection of Sub-word Units 25

3 Unsupervised Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams **29**

3.1 Introduction 29

3.2 System Design 30

3.2.1 Gaussian Posteriorgram Definition 30

3.2.2 Gaussian Posteriorgram Generation 31

3.2.3 Modified Segmental DTW Search 32

3.2.4 Voting Based Score Merging and Ranking 34

3.3 Evaluation 35
3.3.1 TIMIT Experiments 36
3.3.2 MIT Lecture Experiments 39
3.4 Conclusion 41

4 Unsupervised Keyword Spotting Based on Multigram Modeling 43
4.1 Introduction 43
4.2 System Design 43
4.2.1 Overview 43
4.2.2 Unsupervised GMM Learning and Labeling 44
4.2.3 Clustering 44
4.2.4 Keyword to Symbol Sequence Modeling 46
4.2.5 N-best Decoding 52
4.2.6 Sub-symbol-sequence Matching 53
4.3 Experiments 57
4.3.1 TIMIT Dataset 57
4.3.2 Keyword Spotting Experiments 59
4.3.3 Performance Comparison 62
4.3.4 Error Analysis 63
4.4 Discussion 65

5 Unsupervised Learning of Acoustic Units 67
5.1 Introduction 67
5.2 Phonetic Histogram Analysis 68
5.2.1 GMM Training and Clustering 68
5.2.2 Phonetic Histograms 69
5.3 Unsupervised Broad Acoustic Class Modeling 80
5.3.1 Framework Overview 80
5.3.2 Unsupervised Acoustic Feature Analysis 80
5.3.3 Vowel Center Detection 81
5.3.4 Obstruent Detection 87

5.3.5 Voicing Detection 88
5.3.6 Decision Tree Based Inferring 88
5.4 Experiments 89
5.4.1 Vowel Center Detection Performance 90
5.4.2 Obstruent Detection Example 94
5.4.3 Broad Acoustic Class Modeling Result 94
5.5 Summary 96

6 Conclusions and Future Work **99**
6.1 Summary and Contributions 99
6.2 Future Work 101
6.2.1 Unsupervised Keyword Spotting 101
6.2.2 Unsupervised Learning of Acoustic Units 102

List of Figures

2-1 HMM Based Keyword Spotting System 20
2-2 The Utterance and Keyword Lattice 22
2-3 HMM Topology Learning 27

3-1 Segmental DTW Algorithm 34
3-2 Effect of Different Smoothing Factors 37
3-3 Effect of Different SDTW Window Sizes 38
3-4 Effect of Different Score Weighting Factors 39
3-5 Effect of different numbers of Gaussian components 40

4-1 Core Components 45
4-2 Cosegmentation Examples 49
4-3 An Example of the JM Model Learning 51
4-4 Region Voting Scheme 56
4-5 Word Statistics on the TIMIT Dataset 58
4-6 Keyword Spotting Result 1 60
4-7 Keyword Spotting Result 2 61
4-8 Performance Comparison 64

5-1 Gaussian Component Cluster 13 69
5-2 Gaussian Component Cluster 41 71
5-3 Gaussian Component Cluster 14 72
5-4 Gaussian Component Cluster 19 73
5-5 Gaussian Component Cluster 26 74

5-6 Gaussian Component Cluster 28 75
5-7 Gaussian Component Cluster 42 76
5-8 Gaussian Component Cluster 45 77
5-9 Gaussian Component Cluster 48 78
5-10 Gaussian Component Cluster 60 79
5-11 Overview 81
5-12 Algorithm Flowchart 82
5-13 Example of Speech-rhythm Based Detection of Syllable Nuclei 86
5-14 Decision Tree for Broad Acoustic Class Inferring 90
5-15 Distribution of Syllable-Nuclei Intervals in the TIMIT and their Corresponding Rhythm-scaled Versions 91
5-16 Example of Estimated Instantaneous Rhythm Periodicity for a Single TIMIT Utterance 92
5-17 Example of Fricative Detection on a TIMIT Utterance 95

List of Tables

3.1 TIMIT 10 Keyword List 36
3.2 MIT Lecture 30 Keyword List 41
3.3 Effect of Different Numbers of Keyword Examples 41
3.4 30 Keywords Ranked by EER 42

4.1 4 Examples of the Word "artists" 47
4.2 5-best Decoding Result of the Word "artists" 53
4.3 The Scoring Matrix of "aaabbb" and "aabbbb" 54
4.4 8 Keyword Set for TIMIT 59
4.5 Top 5 Error List - 1 63
4.6 Top 5 Error List - 2 63

5.1 Syllable Nuclei Detection Comparison on TIMIT 94
5.2 Gaussian Cluster Statistics 96
5.3 Broad Acoustic Class Hypotheses 97
5.4 Ground Truth of Broad Acoustic Classes 98

Chapter 1

Introduction

Automatic speech recognition (ASR) technology typically requires large quantities of language-specific speech and text data in order to train complex statistical acoustic and language models [4]. Unfortunately, such valuable linguistic resources are unlikely to be available for the majority of languages in the world, especially for less frequently used languages. For example, commercial ASR engines typically support 50-100 (or fewer) languages [1]. Despite substantial development efforts to create annotated linguistic resources that can be used to support ASR development [2], the results fall dramatically short of covering the nearly 7,000 human languages spoken around the globe [3]. For this reason, there is a need to explore ASR training methods which require significantly less language-specific data than conventional methods.

The problem of keyword spotting in audio data has been explored for many years, and researchers typically use ASR technology to detect instances of particular keywords in a speech corpus [33]. Although large-vocabulary ASR methods have been shown to be very effective [42], a popular method incorporates parallel filler or background acoustic models to compete with keyword hypotheses [44, 27]. These keyword spotting methods typically require large amounts of transcribed data for training the acoustic model. For instance, the classic filler model requires hundreds of minutes of speech data transcribed at the word level [27], while in the phonetic lattice matching based approaches [39, 21], the training of a phonetic recognizer needs detailed transcription at the phone level. The required annotation work is not only time con-

suming, it also requires linguistic expertise for providing the necessary annotations which can be a barrier to new languages.

In this thesis, we focus on investigating techniques to perform the keyword spotting task without any transcribed data. Two completely unsupervised keyword spotting systems are presented and carefully evaluated on the TIMIT and MIT Lecture corpus. The results demonstrate the feasibility and effectiveness of our unsupervised learning framework for the task of keyword spotting.

1.1 Motivation

As we enter an era where digital media can be created and accumulated at a rate that far exceeds our ability to annotate it, it is natural to question how much can be learned from the speech data alone, without any supervised input. A related question is what techniques can be performed well using unsupervised techniques in comparison to more conventional supervised training methods. These two questions are the fundamental motivation of our research.

Specifically, the idea of investigating unsupervised learning of speech-related tasks is motivated by the recent trends in data driven methods towards unsupervised large-scale speech data processing. As mentioned, the speed of the speech data production is much faster than data transcription can be performed. We need to find new ways of dealing with untranscribed data instead of waiting until enough transcription work is done. Transcription work is not only time consuming, but also requires some linguistic knowledge. Finally, hiring linguistic professionals to perform these tasks can be very expensive.

The idea of building an unsupervised keyword spotting system is motivated by the trend towards finding useful information from data in multi-media formats. For example, many state-of-the-art search engines provide keyword search interfaces for videos. But most of them generate the index based on the title of the video or the text information accompanying the video, which might not always reflect the true content of the video. With spoken keyword spotting, we can build an index based on

the true content of the audio in order to provide more accurate search results.

Taking these motivations one step further, we are also interested in the self-learning ability of machines. In the case of unsupervised learning, since there is not enough labeled data to guide the model's behavior, the modeling result may vary dramatically based on the learning strategies. Due to various statistical constraints, it is worth investigating the unsupervised modeling results to see how much machines can learn from data, and whether there are some post-processing techniques that can compensate for the missing information brought by the unsupervised modeling methods.

1.2 Overview of Chapters

The remainder of this thesis is organized as follows:

Chapter 2 gives an overview of the related research, including unsupervised learning methods in speech processing, supervised spoken keyword spotting systems and some recent unsupervised spoken keyword spotting systems.

Chapter 3 presents an unsupervised keyword spotting system using Gaussian posteriorgrams. The evaluation results on the TIMIT and MIT Lecture corpus are reported and discussed.

Chapter 4 gives a detailed description of another unsupervised keyword spotting system that does not require any spoken keyword samples. The evaluation results on the TIMIT corpus and the performance comparison with the previous system are reported and analyzed.

Chapter 5 further explores some preliminary investigation on detecting acoustically meaningful units in the speech data without any supervised help. Experimental results on the TIMIT corpus are presented and discussed.

Chapter 6 concludes with a discussion of the potential improvements of the proposed two keyword spotting systems and the future work needed in the unsupervised discovery of acoustically meaningful units.

Chapter 2

Related Work

In this chapter, we give an overview of related research. The organization is as follows. First, we focus on some conventional supervised keyword spotting systems. We divide these systems into two categories and select several representative systems to discuss. Then, we discuss three recently developed unsupervised keyword spotting systems. Finally, we briefly review some research work in unsupervised learning in speech processing.

2.1 Supervised Keyword Spotting Systems

Supervised keyword spotting systems can be categorized into two classes [35]. One is based on HMM learning [25], focusing on detecting keywords at the model level. The other one is based on the post-processing of recognized results, focusing on detecting keywords at the transcription level. The main difference between these two methods is the training requirement. In the HMM-based methods, since a whole-word or phonetic HMM is built for each keyword, the training data must contain enough examples of each keyword. In an unbalanced dataset, the detection performance may vary depending on the number of keyword instances. In contrast, in the post-processing based methods, a universal phonetic HMM can be trained using a large amount of data in which no keyword examples need to be included. After recognizing the test data, a sophisticated phonetic label matching algorithm is performed to find a match

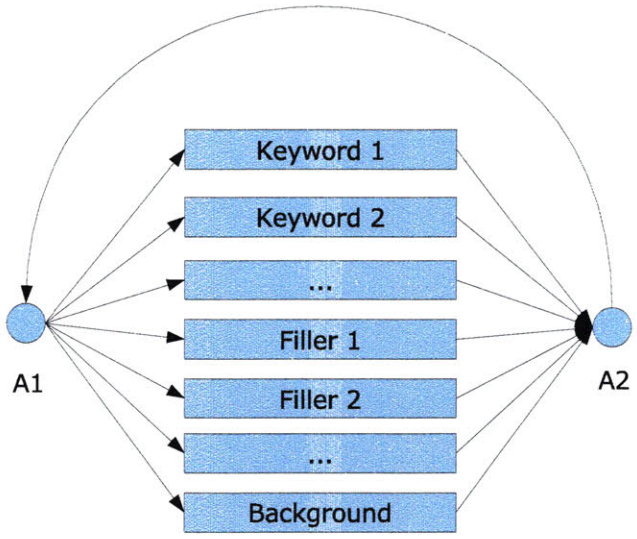

Figure 2-1: This figure illustrates the structure of the HMM-based keyword spotting system. A_1 and A_2 are two auxiliary states in order to provide a self-looping structure. Keyword 1 to N represents keyword HMMs. The filler and background HMMs are used to bypass keyword unrelated speech content and background noise.

for the given keyword. The disadvantage is that since a keyword can potentially be pronounced in many different ways depending on the context, a keyword should be given multiple pronunciation labels in order to let the matching algorithm capture all possible occurrences of that keyword.

2.1.1 HMM Based Methods

Several HMM-based keyword spotting systems have been proposed [27, 43, 37, 26]. The basic architecture of a HMM-based keyword spotting system is shown in Figure 2-1. Each keyword is modeled by one or more HMMs. The filler model is used to cover the speech signal which is unrelated to the keyword. In some systems [45, 44], a background model is used to bypass the possible background noise and silence. A_1 and A_2 denote two auxiliary states that are used for the self-looping structure. In another system [34], a phonetic HMM is used to build the keyword HMM. Since a keyword may have multiple pronunciations, a confusion network based phonetic HMM structure is used to capture the pronunciation variance.

For keyword detection, speech signals are sent into the HMM set and the decoding process is similar to conventional speech recognition. After decoding, a sequence of keyword, filler or background labels is produced, such as

$$\text{Speech Signal} \longrightarrow B_1 B_1 B_2 F_1 F_1 K_1 F_2 F_1 K_2 B_2 B_1 B_1$$

where B_i denotes background labels, F_i denotes filler labels and K_i denotes keyword labels.

2.1.2 Lattice Alignment Based Methods

In post-processing based methods, lattice alignment is the most widely used technique [7, 15, 39, 20, 21]. The basic idea is that for each test utterance, a lattice recognition result is given by the acoustic model. Then, for each keyword, forced-alignment is used to find a small lattice representing only that keyword. The detection decision is made by looking for a sub-matching for the keyword lattice in the utterance lattice. The illustration of these two lattices is shown in Figure 2-2. Based on the keyword lattice, the matching algorithm looks for all possible sub-graph matching in utterance lattices.

Since a lattice is often represented by a directed graph, finding a sub-match in a big directed graph is not an easy problem and can be very time-consuming. But reducing the size of the recognition lattice may lower the resolution of the recognition as well as the keyword detection performance. Therefore, a time alignment method is applied to both the utterance lattice and keyword lattice to first obtain a confusion network. The matching is then performed on the confusion network which is a directed graph with a very constraining structure [20]. Many efficient matching algorithms can be found on this kind of direct graph.

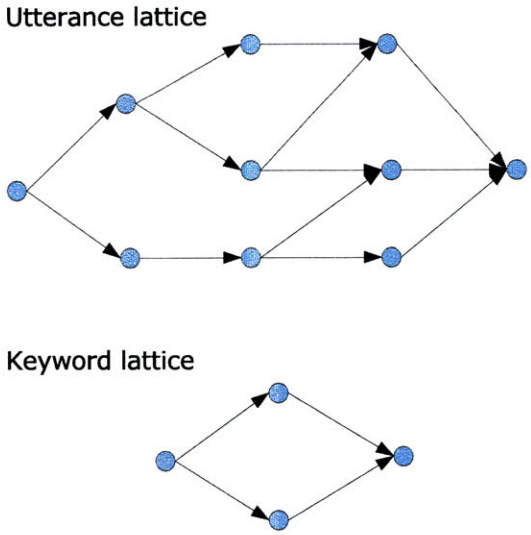

Figure 2-2: This figure illustrates the basic concept of the lattice based keyword spotting system. The top figure is the utterance lattice, while the bottom one is the keyword lattice. Lattice based keyword spotting is performed by looking for a sub-matching of the keyword lattice in the utterance lattice.

2.2 Unsupervised Keyword Spotting Systems

The above two supervised methods can achieve good detection performance when the testing environment is not very different from the training data. But a key problem is that these supervised methods require a large amount of labeled data. It generally is on par with the data requirements for a standard speech recognition system. In other words, good detection performance requires a great amount of human effort. As mentioned previously, when moving to speech processing, unsupervised learning becomes particularly difficult. But keyword spotting is an easier task than speech recognition since it does not require the detection module to understand the entire speech signal. Therefore, it would seem to be a promising task to explore unsupervised methods. We focus on three recent works in the following sections.

2.2.1 Ergodic HMM

Li et al. [19] proposed a keyword spotting system that does not need any manually labeled data for training the acoustic model, and only needs one training instance of a

keyword for detection. The idea is to use a 128-state ergodic HMM with 16 Gaussian mixtures on each state to model speech signals. All of the Gaussian mixtures are initialized by running the K-Means algorithm on the speech feature vectors. After training, similar to supervised keyword spotting systems, the ergodic HMM serves as the background, keyword and filler models.

In the keyword detection stage, they designed a two-pass algorithm. In the first pass, they require a spoken instance of the keyword and use the ergodic HMM to decode this instance into a series of HMM states. Then, they connect these HMM states to form a conventional HMM to act like the keyword model. In the second pass, each test utterance is decoded by the ergodic HMM and this keyword model. If the keyword model gives a higher confidence score, an occurrence of the keyword is found. Their system was evaluated on a Mandarin reading speech corpus. An equal error rate (EER) of 25.9% was obtained on the short keywords and an EER of 7.2% was obtained on long keywords.

2.2.2 Segmental GMM

Another recent work was proposed by Garcia et al. [11]. In this work, the authors had a different modeling strategy for the unsupervised learning of speech signals. Instead of directly modeling the speech signal, the authors first did some segmentation analysis on the signal, and then used a Segmental Gaussian Mixture Model (SGMM) to represent the signal. After modeling, each speech utterance can be decoded into a series of GMM component sequences. By using the Joint Multigram Model (we will discuss details in Chapter 4), a grapheme-to-GMM-component-sequence model can be built to convert a keyword to its corresponding GMM component sequence. Then, a string matching algorithm is used to locate the GMM component sequence of the keyword in the test data.

Specifically, the modeling module consists of three components: the segmenter, the clustering algorithm and the segmental GMM training. The segmenter takes a speech utterance as input and outputs the segmentations based on the occurrence of spectral discontinuities. Then, for each segmentation, a quadratic polynomial function is used

to map the time-varying cepstral features to a fixed length representation. Pair-wised distance is calculated and used to run the clustering algorithm to cluster the similar segmentations. Finally, each group of similar segmentations is modeled by a GMM.

There are two potential problems with this modeling strategy. First, the segmentation performance highly affects the following operations. In other words, since the segmenter makes hard decisions for segmentation, it likely brings segmentation errors into the following processing. Second, the segmental GMM may suffer from the data imbalance problem. The clustering algorithm produces similar segmentation groups, but some groups may only have a small number of training instances. As a result, the GMM trained on these groups may be under-trained, which affects the following decoding performance.

2.2.3 Phonetic Posteriorgram Templates

The most recent work by Hazen et al. [14] showed a spoken keyword detection system using phonetic posteriorgram templates. A phonetic posteriorgram is defined by a probability vector representing the posterior probabilities of a set of pre-defined phonetic classes for a speech frame. By using an independently trained phonetic recognizer, each input speech frame can be converted to its corresponding posteriorgram representation. Given a spoken sample of a keyword, the frames belonging to the keyword are converted to a series of phonetic posteriorgrams by a full phonetic recognition. Then, they use dynamic time warping to calculate the distortion scores between the keyword posteriorgrams and the posteriorgrams of the test utterances. The detection result is given by ranking the distortion scores.

Compared to the segmental GMM method, in order to generate phonetic posteriorgram templates, an independently trained phonetic recognizer and several spoken examples of a keyword are required. In addition, while the phonetic recognizer can be trained independently, it should be trained on the same language used by the keyword spotting data. In summary, the ergodic HMM and segmental GMM method require no transcription for the working data. While the ergodic HMM and phonetic posteriorgram template methods require several spoken instances of a keyword, the

segmental GMM method needs a few transcribed word samples to train a multigram model.

2.3 Unsupervised Learning in Speech

Unsupervised learning is a classic machine learning problem. It differs from supervised learning in that only unlabeled examples are given. Unsupervised learning is commonly used to automatically extract hidden patterns from the data and explain key features of the data. In speech processing, unsupervised learning is a relatively new topic for two reasons. First, the basic goal of speech recognition is to translate the speech signal into its corresponding phone and/or word labels. If no labels were given during training, the recognition model would not know the mapping between the learned signal patterns to the phone or word labels. Thus, it is difficult to finish the translation task without supervised feedback. Second, speech contains highly variable hidden patterns. Current supervised methods still suffer from many critical problems, such as noise robustness and out-of-vocabulary words. An unsupervised learning framework usually has a less constrained model structure than supervised methods, which makes it even harder for it to be used for speech processing. Although it is difficult, there has been some recent promising research showing the possibility of using unsupervised methods to do some simple speech related tasks.

2.3.1 Detection of Sub-word Units

While the Hidden Markov Model (HMM) [25] has been widely used in building speaker independent acoustic models in the last several decades, there are still some remaining problems with HMM modeling. One key problem is the HMM topology selection. The current HMM learning algorithm can efficiently estimate HMM parameters once the topology is given. But the initial topology is often defined by a speech scientist, such as the well-known three states left-to-right HMM structure for tri-phones. This predefined topology is based on empirical results that can be language dependent. For example, in English, it is widely reasonable to have three states per phone, while in

Chinese some phones may need five states [4].

Due to the topology selection problem in HMMs, researchers have focused on developing an unsupervised HMM training framework to let the HMM automatically learn its topology from speech data. After learning the topology, the patterns (sub-word units) in the speech data can be represented by the HMM parameters as well as the corresponding state topology. Previous research generally falls into three categories.

Bottom-up. The basic procedure is: 1) train a HMM from one state (or two states); 2) during the training, contextually or temporally split each state into several states; and 3) re-estimate HMM parameters on these new states as well as the whole HMM. Two major bottom-up approaches are the Li-Biswas method [18] and the ML-SSS [31, 36]. The idea is illustrated in Figure 2-3 on the left.

Top-down. The basic procedure is: 1) initialize a HMM with a very large number of states; 2) during the training, merge similar states by using direct comparison of the Kullback-Leiber (KL) divergence [30] or the decision tree based testing [23]; 3) re-train the merged HMM; and 4) re-do the second step until the resultant HMM reaches an acceptable or pre-defined size. The idea is illustrated in Figure 2-3 on the right.

Combing splitting and merging. Recent research [40] [29] proposed a combined version of the top-down and bottom-up methods. The training is divided into two stages. One stage is the parameter estimation in which a simultaneous temporal and contextual state splitting is performed to expand the previously learned HMM topology. The other stage is the model selection in which a maximum likelihood or Bayesian Information criterion is used to choose a current best splitting topology. Then, after the model selection, the model merges the similar states.

All of these methods were tested on a small speech dataset without transcription and promising results were obtained [40]. Since there is no transcription, the learned HMM is just a mapping from the speech signal to a set of HMM state sequences. By providing some example utterances with transcription, we can use forced-alignment techniques to build a mapping between the similar HMM state sequences to the

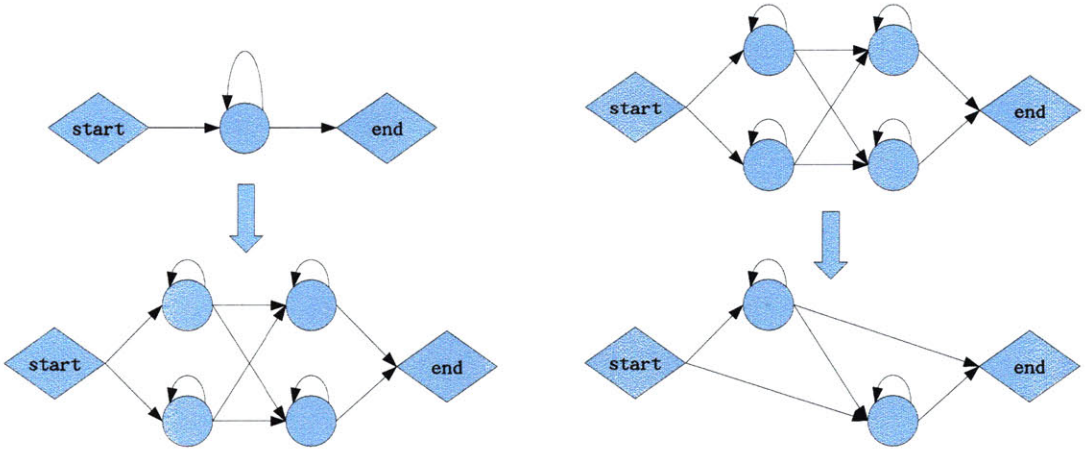

Figure 2-3: These figures illustrate two methods used to learn an HMM topology. The method on the left used a bottom-up approach by iteratively splitting states. The method on the right used a top-down approach by iteratively merging similar states.

sub-word or phone labels in the transcription. In other words, acoustic meaningful sub-word units can be found by grouping similar HMM state sequences. The results were promising because unsupervised HMM learning shows the ability of extracting sub-word units using only a few example utterances with transcription. If the example utterances cover the entire sub-word or phone inventory, it becomes possible to build a complete mapping between learned sub-word units and the actual phone or word labels so that conventional speech recognition can be performed.

Chapter 3

Unsupervised Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams

3.1 Introduction

In this chapter, we present an unsupervised keyword spotting system using Gaussian posteriorgrams. Without any transcription information, a Gaussian Mixture Model is trained to label speech frames with a Gaussian posteriorgram. Given one or more spoken examples of a keyword, we use segmental dynamic time warping to compare the Gaussian posteriorgrams between keyword examples and test utterances. The keyword detection result is then obtained by ranking the distortion scores of all the test utterances. We examine the TIMIT corpus as a development set to tune the parameters in our system, and the MIT Lecture corpus for more substantial evaluation. The results demonstrate the viability and effectiveness of this unsupervised learning framework on the keyword spotting task.

The keyword spotting scenario discussed in this chapter assumes that a small number of audio examples of a keyword are known by the user. The system then searches the test utterances based on these known audio examples. In other words,

an example of the keyword query must appear in the training data.

The remainder of this chapter is organized as follows: the following section briefly reviews three previous unsupervised keyword spotting systems; Section 3.3 describes the detailed design of our system; experiments on the TIMIT and the MIT Lecture corpus are presented in Section 3.4; and our work is summarized in Section 3.5.

3.2 System Design

Our approach is most similar to the research explored by Hazen et al. [14]. However, instead of using an independently trained phonetic recognizer, we directly model the speech using a GMM without any supervision. As a result, the phonetic posteriorgram effectively becomes a Gaussian posteriorgram. Given spoken examples of a keyword, we apply the segmental dynamic time warping (SDTW) that we have explored previously [24] to compare the Gaussian posteriorgrams between keyword examples and test utterances. We output the keyword detection result by ranking the distortion scores of the most reliable warping paths. We give a detailed description of each procedure in the following sections.

3.2.1 Gaussian Posteriorgram Definition

Posterior features have been widely used in template-based speech recognition [5, 6]. In a manner similar to the definition of the phonetic posteriorgram [14], a Gaussian posteriorgram is a probability vector representing the posterior probabilities of a set of Gaussian components for a speech frame. Formally, if we denote a speech utterance with n frames as

$$S = (s_1, s_2, \cdots, s_n)$$

then the Gaussian posteriorgram (GP) is defined by:

$$GP(S) = (q_1, q_2, \cdots, q_n) \tag{3.1}$$

Each q_i vector can be calculated by

$$q_i = (P(C_1|s_i), P(C_2|s_i), \cdots, P(C_m|s_i))$$

where C_i represents i-th Gaussian component of a GMM and m denotes the number of Gaussian components. For example, if a GMM consists of 50 Gaussian components, the dimension of the Gaussian posteriorgram vector should be 50. If the speech utterance contains 200 frames, the Gaussian posteriorgram matrix should be 200x50.

3.2.2 Gaussian Posteriorgram Generation

The generation of a Gaussian posteriorgram is divided into two phases. In the first phase, we train a GMM on all the training data and use this GMM to produce a raw Gaussian posteriorgram vector for each speech frame. In the second phase, a discounting based smoothing technique is applied to each posteriorgram vector.

The GMM training in the first phase is a critical step, since without any guidance, it is easy to generate an unbalanced GMM. Specifically, it is possible to have a GMM with a small number of Gaussian components that dominate the probability space, with the remainder of the Gaussian components representing only a small number of training examples. We found this to be particularly problematic for speech in the presence of noise and other non-speech artifacts, due to their large variance. The unfortunate result of such a condition was a posteriorgram that did not discriminate well between phonetic units. For example, some dimensions in the generated posteriorgrams always control the larger probability mass (95%), while the remaining dimensions only have a very small probability mass (5%). Approximation errors would greatly affect the result in discriminating these posteriorgrams. Our initial solution to this problem was to apply a speech/non-speech detector to extract speech segments, and to only train the GMM on these segments.

After a GMM is trained, we use Equation (3.1) to calculate a raw Gaussian posteriorgram vector for each speech frame and the given spoken keyword examples. To avoid approximation errors, a probability floor threshold P_{min} is set to eliminate

dimensions (i.e., set them to zero) with posterior probabilities less than P_{min}. The vector is re-normalized to set the summation of each dimension to one. Since this threshold would create many zeros in the Gaussian posteriorgram vectors, we apply a discounting based smoothing strategy to move a small portion of probability mass from non-zero dimensions to zero dimensions. Formally, for each Gaussian posteriorgram vector q, each zero dimension z_i is assigned by

$$z_i = \frac{\lambda \cdot 1}{Count(z)}$$

where $Count(z)$ denotes the number of zero dimensions. Each non-zero dimension v_i is changed to

$$v_i = (1 - \lambda)v_i$$

3.2.3 Modified Segmental DTW Search

After extracting the Gaussian posteriorgram representation of the keyword examples and all the test utterances, we perform a simplified version of the segmental dynamic time warping (SDTW) to locate the possible occurrences of the keyword in the test utterances.

SDTW has demonstrated its success in unsupervised word acquisition [24]. To apply SDTW, we first define the difference between two Gaussian posterior vectors p and q:

$$D(p, q) = -\log(p \cdot q)$$

Since both p and q are probability vectors, the dot product gives the probability of these two vectors drawing from the same underlying distribution [14].

SDTW defines two constraints on the DTW search. The first one is the commonly used adjustment window condition [28]. In our case, formally, suppose we have two Gaussian posteriorgram $GP_i = (p_1, p_2, \cdots, p_m)$ and $GP_j = (q_1, q_2, \cdots, q_n)$,

the warping function $w(\cdot)$ defined on a $m \times n$ timing difference matrix is given as $w(k) = (i_k, j_k)$ where i_k and j_k denote the k-th coordinate of the warping path. Due to the assumption that the duration fluctuation is usually small in speech [28], the adjustment window condition requires that

$$|i_k - j_k| \leq R$$

This constraint prevents the warping path from getting too far ahead or behind in either GP_i or GP_j.

The second constraint is the step length of the start coordinates of the DTW search. It is clear that if we fix the start coordinate of a warping path, the adjustment window condition restricts not only the shape but also the ending coordinate of the warping path. For example, if $i_1 = 1$ and $j_1 = 1$, the ending coordinate will be $i_{end} = m$ and $j_{end} \in (1 + m - R, 1 + m + R)$. As a result, by applying different start coordinates of the warping process, the difference matrix can be naturally divided into several continuous diagonal regions with width $2R + 1$, shown in Figure 3-1. In order to avoid the redundant computation of the warping function as well as taking into account warping paths across segmentation boundaries, we use an overlapped sliding window moving strategy for the start coordinates ($s1$ and $s2$ in the figure). Specifically, with the adjustment window size R, every time we move R steps forward for a new DTW search. Since the width of each segmentation is $2R+1$, the overlapping rate is 50%.

Note that in our case, since the keyword example is fixed, we only need to consider the segment regions in the test utterances. For example, if GP_i represents the keyword posteriorgram vector and GP_j is the test utterance, we only need to consider the regions along the j axis. Formally, given the adjustment window size R and the length of the test utterance n, the start coordinate is

$$(1, (k-1) \cdot R + 1),\ 1 \leq k \leq \left\lfloor \frac{n-1}{R} \right\rfloor$$

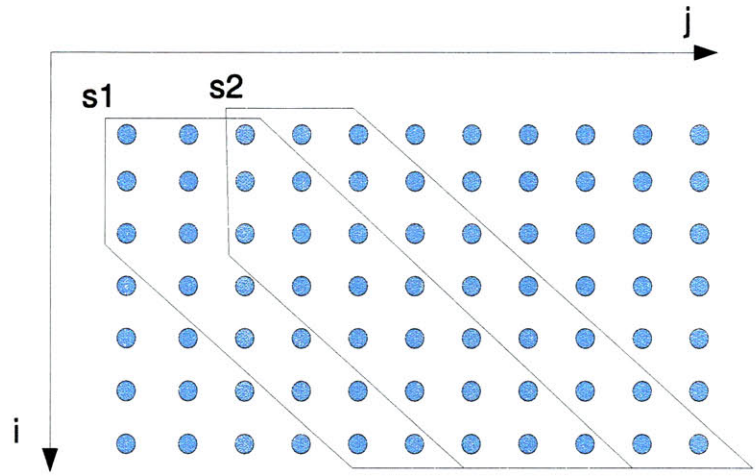

Figure 3-1: This figure illustrates the basic idea of the segmental DTW algorithm. s_1 and s_2 are the first two start coordinates of the warping path with $R = 2$. The pentagons are the segmentation determined by the corresponding start coordinate s_i and the adjustment window condition R.

As we keep moving the start coordinate, for each keyword, we will have a total of $\left\lfloor \frac{n-1}{R} \right\rfloor$ warping paths, each of which represents a warping between the entire keyword example and a portion of the test utterance.

3.2.4 Voting Based Score Merging and Ranking

After collecting all warping paths with their corresponding distortion scores for each test utterance, we simply choose the warping region with the minimum distortion score as the candidate region of the keyword occurrence for that utterance. However, if multiple keyword examples are provided and each example provides a candidate region with a distortion score, we need a scoring strategy to calculate a final score for each test utterance, taking into account the contribution of all keyword examples.

In contrast to the direct merging method used [14], we considered the reliability of each warping region on the test utterance. Given multiple keyword examples and a test utterance, a reliable warping region on the test utterance is the region where most of the minimum distortion warping paths of the keyword examples are aligned. In this way a region with a smaller number of alignments to keyword examples is considered

to be less reliable than a region with a larger number of alignments. Therefore, for each test utterance, we only take into account the warping paths pointing to a region that contains alignments to multiple keyword examples.

An efficient binary range tree is used to count the number of overlapped alignment regions on a test utterance. After counting, we consider all regions with only one keyword example alignment to be unreliable, and thus the corresponding distortion scores are discarded. We are then left with regions having two or more keyword examples aligned. We then apply the same score fusion method [14]. Formally, if we have $k \geq 2$ keyword examples s_i aligned to a region r_j, the final distortion score for this region is:

$$S(r_j) = -\frac{1}{\alpha} \log \frac{1}{k} \sum_{i=1}^{k} \exp(-\alpha S(s_i)) \tag{3.2}$$

where varying α between 0 and 1 changes the averaging function from a geometric mean to an arithmetic mean. Note that since one test utterance may have several regions having more than two keyword alignments, we choose the one with the smallest average distortion score. An extreme case is that some utterances may have no warping regions with more than one keyword alignment (all regions are unreliable). In this case we simply set the distortion score to a very big value.

After merging the scores, every test utterance should have a distortion score for the given keyword. We rank all the test utterances by their distortion scores and output the ranked list as the keyword spotting result.

3.3 Evaluation

We have evaluated this unsupervised keyword spotting framework on two different corpora. We initially used the TIMIT corpus for developing and testing the ideas we have described in the previous section. Once we were satisfied with the basic framework, we performed more thorough large vocabulary keyword spotting experiments on the MIT Lecture corpus [13]. In addition, we compared several keyword spotting experiments against the learned acoustic units based system in Chapter 4.

Table 3.1: TIMIT 10 keyword list. "word" (a, b) represents the keyword "word" occurs a times in the training set and b times in the test set.

age(3:8)	warm(10:5)	year(11:5)	money(19:9)
artists(7:6)	problem(22:13)	children(18:10)	surface(3:8)
development(9:8)	organizations(7:6)		

The evaluation metrics that we report follow those suggested by [14]: 1) P@10 : the average precision for the top 10 hits; 2) P@N : the average precision of the top N hits, where N is equal to the number of occurrences of each keyword in the test data; 3) EER : the average equal error rate at which the false acceptance rate is equal to the false rejection rate. Note that we define a putative hit to be correct if the system proposes a keyword that occurs somewhere in an utterance transcript.

3.3.1 TIMIT Experiments

The TIMIT experiment was conducted on the standard 462 speaker training set of 3,696 utterances and the common 118 speaker test set of 944 utterances. The total size of the vocabulary was 5,851 words. Each utterance was segmented into a series of 25 ms frames with a 10 ms frame rate (i.e., centi-second analysis); each frame was represented by 13 Mel-Frequency Cepstral Coefficients (MFCCs). Since the TIMIT data consists of read speech in quiet environments, we did not apply the speech detection module for the TIMIT experiments. All MFCC frames in the training set were used to train a GMM with 50 components. We then used the GMM to decode both training and test frames to produce the Gaussian posteriorgram representation. For testing, we randomly generated a 10-keyword set and made sure that they contained a variety of numbers of syllables. Table 3.1 shows the 10 keywords and their number of occurrences in both training and test sets (# training : # test).

We first examined the effect of changing the smoothing factor, λ in the posteriorgram representation when fixing the SDTW window size to 6 and the score weighting factor α to 0.5, as illustrated in Figure 3-2. Since the smoothing factor ranges from 0.1 to 0.00001, we use a log scale on the x axis. Note that we do not plot the value

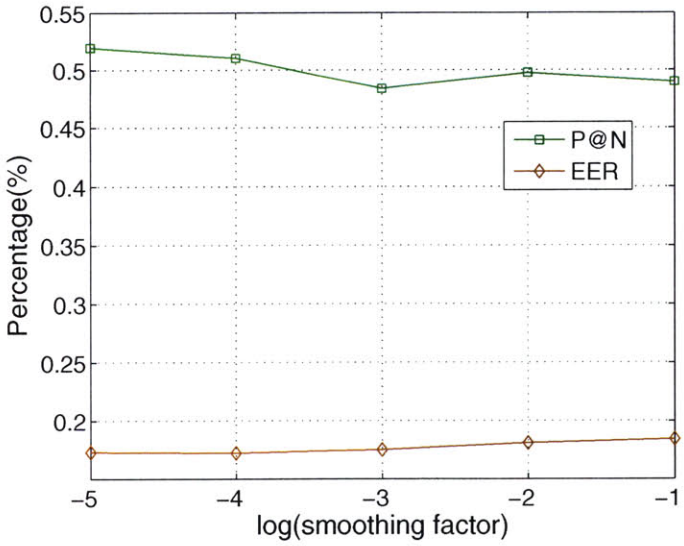

Figure 3-2: This figure illustrates the effect of setting different smoothing factors. Since in terms of EER (in red), the point -4 holds the best value and P@N is the second best, we choose $\lambda = 0.0001$ (the log base is 10) as the best setting for the smoothing factor.

for P@10 because as we can see in Table 3.1, not all keywords occur more than ten times in the test set. In the figure, $\lambda = 0.0001$ was the best setting for the smoothing factor mainly in terms of EER, so this value was used for all subsequent experiments.

As shown in Figure 3-3, we next investigated the effect of setting different adjustment window sizes for the SDTW when fixing the smoothing factor to 0.0001 and the score weighting factor α to 0.5. The results shown in the figure confirmed our expectation that an overly small DTW window size could overly restrict the warp match between keyword references and test utterances, which could lower the performance. An overly generous DTW window size could allow warping paths with an excessive time difference, which could also affect the performance. Based on these experiments, we found that a window size equal to 6 was the best considering both P@N and EER.

We also ran the keyword spotting experiments with different settings of the score weighting factor α while fixing the smoothing factor to 0.0001 and SDTW window size to 6. As shown in Figure 3-4, P@N basically prefers the arithmetic mean metric, while the EER metric is relatively steady. By considering both metrics, we chose 0.5

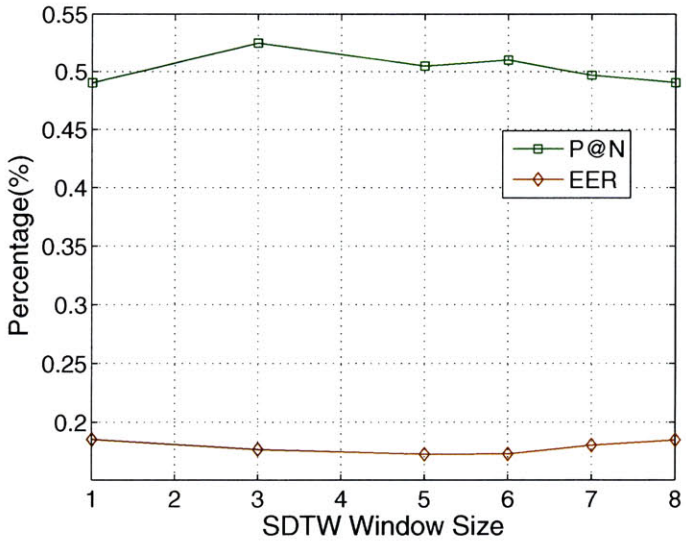

Figure 3-3: This figure illustrates the effect of setting different numbers of Gaussian components. At windows size 6, the best EER and second best P@N can be obtained.

as the best setting for α.

The number of Gaussian components in the GMM is another key parameter in our system. With fixed smoothing factor (0.0001), SDTW window size (6) and score weighting factor (0.5), we ran several experiments with GMMs with different numbers of Gaussian components, as illustrated in Figure 3-5. Due to the random initialization of the K-means algorithm for GMM training, we ran each setting five times and reported the best number. The result indicated that the number of Gaussian components in the GMM training has a key impact on the performance. When the number of components is small, the GMM training may suffer from an underfitting problem, which causes a low detection rate in P@N. In addition, the detection performance is not monotonic with the number of Gaussian components. We think the reason is that the number of Gaussian components should approximate the number of underlying broad phone classes in the language. As a result, using too many Gaussian components will cause the model to be very sensitive to variations in the training data, which could result in generalization errors on the test data. Based on these results, we chose 50 as the best number of GMM components.

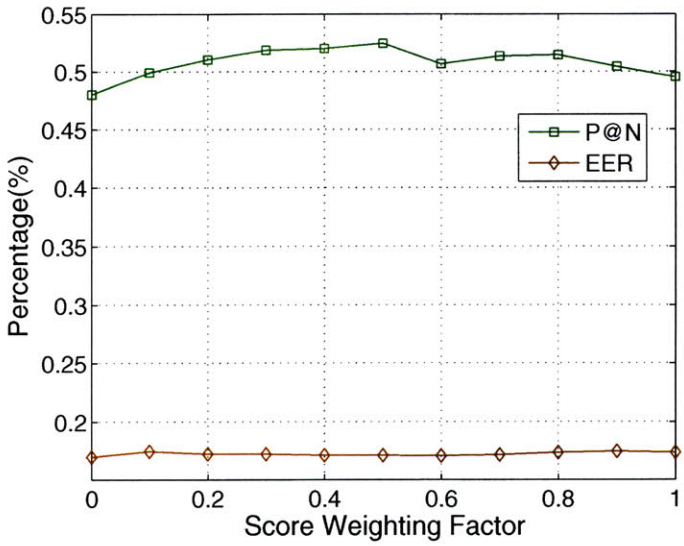

Figure 3-4: This figure illustrates the effect of setting different score weighting factors. At $\alpha = 0.5$, the best EER and P@N can be obtained.

3.3.2 MIT Lecture Experiments

The MIT Lecture corpus consists of more than 300 hours of speech data recorded from eight different courses and over 80 general seminars [13]. In most cases, the data is recorded in a classroom environment using a lapel microphone. For these experiments we used a standard training set containing 57,351 utterances and a test set with 7,375 utterances. The vocabulary size of both the training and the test set is 27,431 words.

Since the data was recorded in a classroom environment, there are many non-speech artifacts that occur such as background noise, filled pauses, laughter, etc. As we mentioned in the GMM training section, this non-speech data could cause serious problems in the unsupervised learning stage of our system. Therefore, prior to GMM training, we first ran a speech detection module [12] to filter out non-speech segments. GMM learning was performed on frames within speech segments. Note that the speech detection module was trained independently from the Lecture data and did not require any transcription of the Lecture data. 30 keywords were randomly selected; all of them occur more than 10 times in both the training and test sets. In

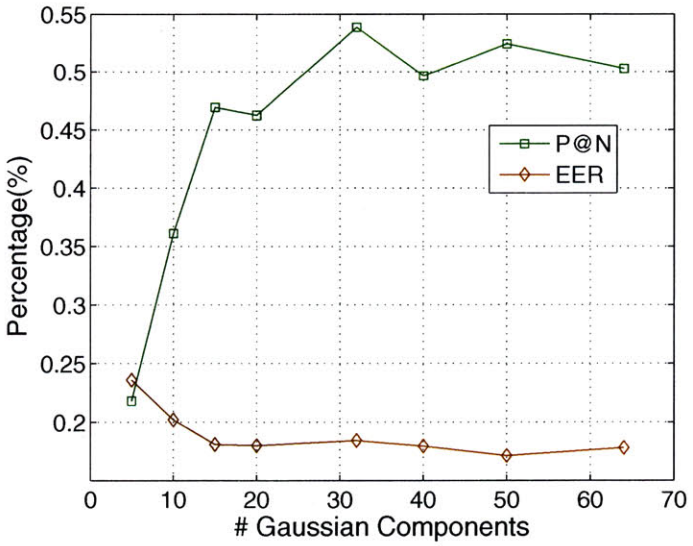

Figure 3-5: This figure illustrates the effect of setting different number of Gaussian components. When the number of Gaussian component equals to 50, the best EER and second P@N can be obtained.

addition, all keywords occur less than 80 times in the test set to avoid using keywords that are too common in the data. Table 3.2 shows all the keywords and the number of occurrences in the training and test sets.

Table 3.3 shows keyword detection performance when different numbers of keyword examples are given. As a result of the TIMIT experiments, we fixed the smoothing factor to 0.0001, the SDTW window size to 6 and the score weighting factor to 0.5. All of the three evaluation metrics improve dramatically from the case in which only one keyword example is given to the case in which five examples are given. Beyond five examples of a keyword, the trend of the performance improvement slows. We believe the reason for this behavior is that the improvement from one example to five examples is mainly caused by our voting-based, score-merging strategy. When going from five examples to ten examples, we gain additional performance improvement, but there are always some difficult keyword occurrences in the test data. Table 3.4 gives the list of 30 keywords ranked by EER in the 10-example experiment. We observe that the words with more syllables tended to have better performance than ones with only two or three syllables.

Table 3.2: "word" (a, b) represents the keyword "word" occurs a times in the training set and b times in the test set.

zero (247:77)	space (663:32)	solutions (33:29)
examples (137:29)	performance (72:34)	matter (353:34)
molecule (28:35)	pretty (403:34)	results (121:35)
minus (103:78)	computer (397:43)	value (217:76))
situation (151:10)	therefore (149:46)	important (832:47)
parameters (21:50)	negative (50:50)	equation (98:61)
distance (58:56)	algorithm (35:36)	direction (214:37)
maximum (20:32)	responsible (92:10)	always (500:37)
likelihood (13:31)	mathematical (37:15)	never (495:21)
membrane (19:27)	problems (270:23)	course (847:76)

Table 3.3: Effect of Different Numbers of Keyword Examples

# Examples	P@10	P@N	EER
1	27.00%	17.33%	27.02%
5	61.33%	32.99%	16.82%
10	68.33%	39.31%	15.76%

Since the data used in [14] is not yet publicly available, we are unable to perform direct comparisons with their experiments. Nevertheless, we can make superficial comparisons. For example, in the case of 5 keyword examples, the P@10 performance (61.33%) of our system is competitive with their result (63.30%), while for the P@N and EER metrics, we are lower than theirs (P@N : 32.99% vs. 52.8%, EER : 16.8% vs. 10.4%). We suspect that one cause for the superior performance of their work is their use of a well-trained phonetic recognizer. However, this will require additional investigation before we can quantify this judgement.

3.4 Conclusion

In this chapter we have presented an unsupervised framework for spoken keyword detection. Without any annotated corpus, a completely unsupervised GMM learning framework is introduced to generate Gaussian posteriorgrams for keyword examples and test utterances. A modified segmental DTW is used to compare the Gaussian

Table 3.4: 30 Keywords ranked by EER. The EER value is within parentheses.

responsible (0.23%)	direction (10.25%)	matter (22.80%)
situation (0.46%)	parameters (10.48%)	always (23.00%)
molecule (4.93%)	algorithm (11.26%)	therefore (23.92%)
mathematical (6.66%)	course (11.40%)	membrane (24.02%)
maximum (7.50%)	space (13.75%)	equation (24.85%)
solutions (8.13%)	problems (17.78%)	computer (25.25%)
important (8.50%)	negative (18.00%)	minus (25.65%)
performance (8.82%)	value (19.35%)	examples (27.02%)
distance (8.96%)	likelihood (19.36%)	pretty (29.09%)
results (9.27%)	zero (22.65%)	never (29.53%)

posteriorgrams between keyword examples and test utterances. After collecting the warping paths from the comparison of every pair of the keyword example and the test utterance, we use a voting-based, score-merging strategy to give a relevant score to every test utterance for each keyword. The detection result is determined by ranking all the test utterances with respect to their relevant scores. In the evaluation, due to various system parameters, we first designed several experiments on the smaller TIMIT dataset to have a basic understanding of appropriate parameter settings as well as to verify the viability of our entire framework. We then conducted experiments on the MIT Lecture corpus, which is a much larger vocabulary dataset, to further examine the effectiveness of our system. The results were encouraging and were somewhat comparable to other methods that require more supervised training [14].

Chapter 4

Unsupervised Keyword Spotting Based on Multigram Modeling

4.1 Introduction

In the last chapter, we presented an unsupervised keyword spotting system based on matching Gaussian posteriorgrams. When performing a keyword search, one or a few audio samples of the keyword must be provided. In some scenarios, however, audio samples of a keyword are not very easy to obtain. It is desirable to develop an alternative method which only requires text input instead of audio based samples. In this chapter, we will present an unsupervised keyword spotting system which only requires time aligned text samples of a keyword. The basic idea is derived from the system proposed in Chapter 3, but differs in the way it represents speech frames and the use of a multigram model to enable text-based keyword search.

4.2 System Design

4.2.1 Overview

We first give an overview of the system. The inputs are untranscribed speech data and a keyword. The output is a list of test utterances ranked by the probability

of containing the keyword. In Chapter 3, we have demonstrated how to use the Gaussian posteriorgram to represent speech frames, while in this system, instead of calculating the frame's posterior probability on every Gaussian component, we simply label each speech frame by the Gaussian component with the highest posterior probability. In other words, each frame is only represented by an index label. By providing some time aligned example utterances, a Joint-Multigram model is then trained to build a mapping between letters and Gaussian component indices. Then, the multigram model is used to decode any given keyword into Gaussian component indices. Keyword detection is done by comparing the decoded Gaussian component indices with all the test speech frames also represented by the most probable Gaussian component labels.

The core components of the system as well as their relationship are illustrated in Figure 4-1. Detailed descriptions of each component is given in the following sections.

4.2.2 Unsupervised GMM Learning and Labeling

The unsupervised GMM training is exactly the same as what we did in Chapter 3. After a GMM is trained, the posterior probability of each speech frame is calculated on every Gaussian component. The label of the Gaussian component with the highest posterior probability is then selected to represent that speech frame. After this step, all speech frames from either training or test utterances are labeled frame-by-frame by Gaussian component indices.

4.2.3 Clustering

Due to speaker variance, we only use the most probable Gaussian component index to represent speech frames. Therefore, even for a single phone, it is common to have different Gaussian component index labeling results. For example, frames in a phone spoken by a female speaker could be labeled by Gaussian component 15, while frames in the same phone spoken by a male speaker could be labeled by Gaussian component 28. Since the keyword is directly represented by a series of Gaussian

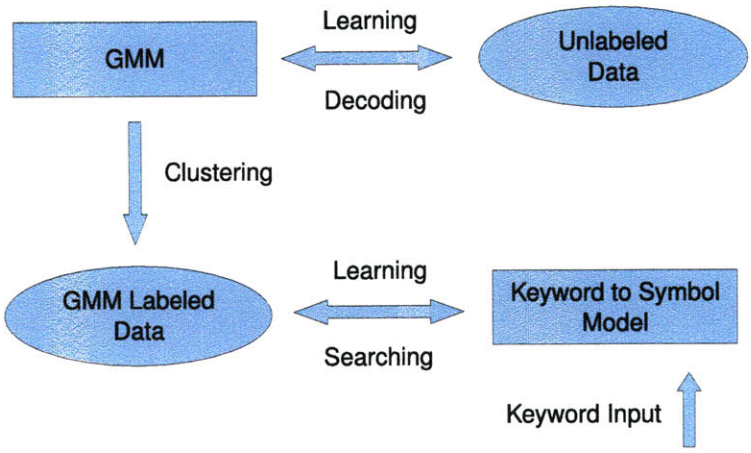

Figure 4-1: This figure illustrates the core components in the proposed unsupervised keyword spotting system. The input is untranscribed speech data and a keyword. GMM is used to learn and decode speech frames. A clustering algorithm is applied to the learned GMM to produce Gaussian component clusters. Each speech frame is then labeled by the Gaussian component cluster where it most likely belongs. A keyword to symbol model is employed to translate the keyword input into several possible Gaussian cluster index sequences. Keyword spotting is done by searching the keyword generated Gaussian cluster index sequences in the test data.

component indices, the quality of representation plays a critical role in our system design. To address this problem, a clustering algorithm is used to reduce the number of Gaussian components based on their statistical similarity. After clustering, we replace the Gaussian component label with the cluster label it belongs to. In the previous example, if Gaussian component 15 and 28 are clustered together, a cluster label is given to all the frames belonging to Gaussian component 15 and 28.

To cluster Gaussian components, we first define a similarity metric between two components. There are several ways to measure the similarity between probability

distributions. The well-known Jensen-Shannon Divergence [30] is employed as the measurement. The Jensen-Shannon Divergence (JSD) of two Gaussian components is defined by

$$JSD(G_1||G_2) = \frac{1}{2}D(G_1||G_m) + \frac{1}{2}D(G_2||G_m) \tag{4.1}$$

where

$$G_m = \frac{1}{2}(G_1 + G_2) \tag{4.2}$$

and D is the standard Kullback-Leibler (KL) divergence

$$D(G||G_m) = \sum_i G(i) \log \frac{G(i)}{G_m(i)} \tag{4.3}$$

By using JSD, a pair-wised similarity matrix can be produced for all the learned Gaussian components. Then, an unsupervised clustering algorithm can be employed on this similarity matrix. Since it is difficult to determine the best number of clusters, we used the Affinity Propagation [10] algorithm, which does not require a pre-defined number of clusters, and also works well for poor initialization of cluster centers. This algorithm has been shown to be able to find clusters with a lower error rate than other popular methods [10]. Once the clustering is complete, we should have a reduced number of Gaussian components. Each Gaussian component is then relabeled by its corresponding representative cluster as well as the decoding results of the first stage.

4.2.4 Keyword to Symbol Sequence Modeling

In order to map Gaussian component indices to letters, some transcribed time aligned keyword examples are needed. Two strategies are explored for building this mapping. One is called flat modeling, and the other is called multigram modeling.

Note that this transcribed data can be a very small portion of the training data. For example, the number of example utterances can range from a few seconds to roughly 10 minutes in duration. Such a relatively small amount of labeling work is

Table 4.1: 4 examples of the word "artists." "60:2" represents the first two speech frames are labeled by Gaussian component 60.

	artists
1	60:2 42:13 14:5 48:1 14:1 41:11 26:1 13:13 26:1 13:2 26:1 41:1
2	42:10 41:4 48:1 28:2 19:1 26:1 13:5 26:1 41:8 13:9 26:3 13:6
3	42:20 41:9 13:9 41:1
4	14:1 42:1 14:15 48:1 28:1 48:1 41:1 48:2 14:1 48:3 41:2 26:1 13:15 26:1

affordable and can be done in a reasonable time.

Flat Modeling

The flat modeling approach collects all occurrences of an input keyword in the example utterances. Each keyword instance provides an alternative instance of a keyword. Thus, after word-level alignment, each example can provide a Gaussian component sequence that corresponds to the input keyword. The flat modeling collects these sequences and builds a one-to-many mapping from the input keyword to these possible Gaussian component sequences.

Table 4.1 shows an example of the keyword "artists" and its corresponding 4 example utterances in TIMIT. Note that in this table, "42:10" means there are ten occurrences of the Gaussian component "42." It is clear that these 4 Gaussian component sequences are different from each other but share some similar segments such as the beginning of the second and the third example.

Joint Multigram Modeling

The flat model is straightforward, but it does not use the shared patterns among the multiple example Gaussian component sequences. These shared patterns could be useful because they indicate the reliable and consistent regions of the output of the unsupervised GMM learning. By using these shared patterns, we can represent the keyword by combining several patterns, which is expected to be a better generalization

of the training data example.

The basic idea is that instead of directly mapping a keyword to Gaussian component sequences, we build a letter to Gaussian component sequence mapping that can produce Gaussian component sequences at the letter level. Specifically, given a word "word," flat modeling builds a mapping from "word" to Gaussian component sequence "112233" and "11222333," while the new model builds three mappings which are "w" to "11," "or" to "22" and "d" to "33." In the latter case, if we collect enough mappings that cover all the letters (and their possible combinations) in English. Then, when facing a new keyword, we can still produce a possible Gaussian component sequence by combining mappings of the corresponding letters. We call this model the Joint Multigram (JM) model [9]. The following paragraphs will give a formal definition of this model.

The JM model tries to model two symbol sequences with hidden many-to-many mapping structures. The model takes two symbol sequences as input, and outputs the best joint segmentation of these two sequences using the maximum likelihood criterion. Formally, if $A = a_1 \cdots a_n$ and $B = b_1 \cdots b_m$ are two streams of symbols, the JM model defines a likelihood function of joint cosegmentation L of the input sequences A and B

$$\mathcal{L}(A, B) = \sum_{L \in \{L\}} \mathcal{L}(A, B, L) \tag{4.4}$$

where L

$$L = \begin{pmatrix} a_{(s_1)} & a_{(s_2)} & \cdots & a_{(s_w)} \\ b_{(t_1)} & b_{(t_2)} & \cdots & b_{(t_v)} \end{pmatrix}. \tag{4.5}$$

is the set of all possible cosegmentations and $\mathcal{L}$ is the likelihood function. $a_{(s_i)}$ and $b_{(t_j)}$ defines one symbol or a combination of several symbols in sequence A and B.

Note that segmentations in each cosegmentation pair can have arbitrary length. For example, the length of $a_{(s_2)}$ and $b_{(t_2)}$ can be unequal. But to make this model tractable, we usually define two maximum lengths (L_a, L_b) to constrain the maximum

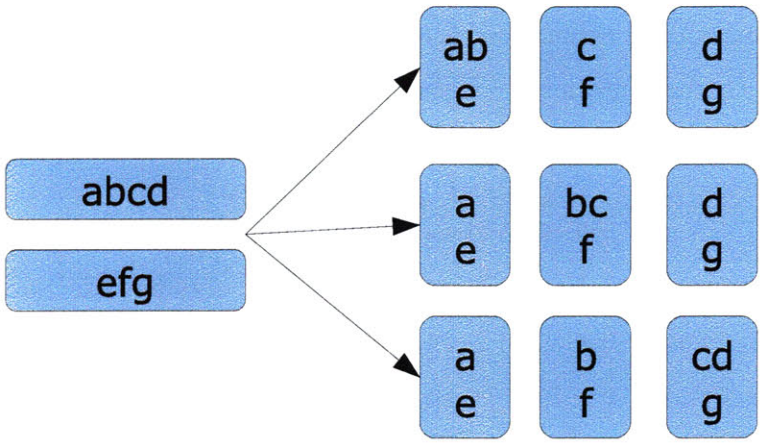

Figure 4-2: This figure illustrates a cosegmentation of two string streams "abcd" and "efg". A cosegmentation consists of two segments from the input two strings respectively, e.g., $\begin{pmatrix} ab \\ e \end{pmatrix}$. and $\begin{pmatrix} c \\ f \end{pmatrix}$.

possible length of the segmentation of A and B. For instance, if we define a $(2, 3)$ joint multigram model, the maximum length of $a_{(s_i)}$ is 2 and the maximum length of $b_{(s_i)}$ is 3. Figure 4-2 gives an example of all possible cosegmentations of two streams of symbols "abcd" and "efg." The left part of the figure shows the possible cosegmentations of these two strings.

Then, we define the probability of a cosegmentation as

$$P(a_{(s_i)}, b_{(s_j)}) \tag{4.6}$$

which means the probability of cosegmenting $a_{(s_i)}$ and $b_{(s_j)}$ from the given symbol sequences. Clearly, we have

$$\sum_{i,j} P(a_{(s_i)}, b_{(s_j)}) = 1 \tag{4.7}$$

Therefore, a JM model JM can be represented by

$$JM = (L_a, L_b, P_c) \tag{4.8}$$

where P_c is the set of probabilities of all possible cosegmentations.

If we have a full estimation of P_c, by iterating all possible cosegmentations, the most likely cosegmentation L' is given by

$$L' = \arg \max_{L \in \{L\}} \mathcal{L}(A, B, L) \tag{4.9}$$

The estimation of P_c can be obtained by the standard EM algorithm and has an efficient forward-backward implementation. We give a brief description of the forward-backward algorithm and the update function.

We define a forward variable $\alpha(i, j)$ which represents the likelihood of all possible cosegmentations up to position i in symbol sequence A and j in symbol sequence B, and a backward variable $\beta(i, j)$ which accounts for the likelihood of all possible cosegmentations starting from $i + 1$ in A and $j + 1$ in B to the end. Formally, we have

$$\begin{aligned} \alpha(i, j) &= \mathcal{L}(A_1^i, B_1^j) \\ \beta(i, j) &= \mathcal{L}(A_{i+1}^n, B_{j+1}^m) \end{aligned}$$

Then, we can calculate α recursively by

$$\alpha(i, j) = \sum_{k=1}^{L_a} \sum_{l=1}^{L_b} \alpha(i - k, j - l) P(A_{i-k+1}^i, B_{j-l+1}^j)$$

where $0 < i \leq n$ and $0 < j \leq m$. Similarly, we can calculate β as

$$\beta(i, j) = \sum_{k=1}^{L_a} \sum_{l=1}^{L_b} \beta(i + k, j + l) P(A_{i+1}^{i+k}, B_{j+1}^{j+l})$$

where $0 \leq i < n$ and $0 \leq j < m$. Note that the initial conditions are $\alpha(0, 0) = 1$ and $\beta(n, m) = 1$.

Then, if we denote α^r, β^r and P^r as the statistics in the r-th iteration, we can derive the update function of the $P^{r+1}(a_{(s_p)}, b_{(s_q)})$ as

$$P^{r+1}(a_{(s_p)}, b_{(s_q)}) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha^r(i - k, j - l) P^r(a_{(s_p)}, b_{(s_q)}) \beta^r(i, j)}{\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha^r(i, j) \beta^r(i, j)}$$

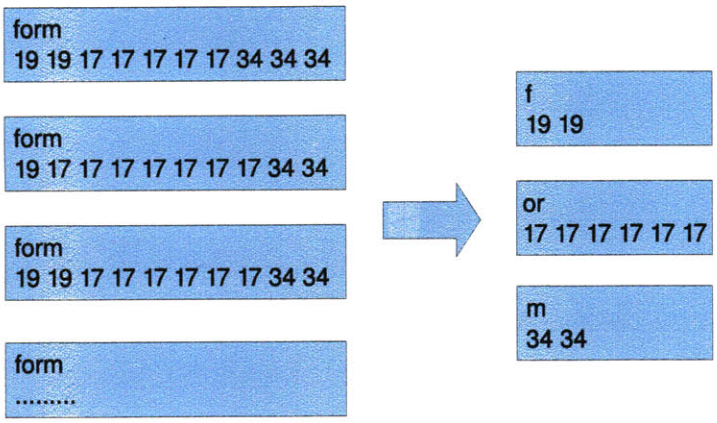

Figure 4-3: This figure illustrates an example of the JM Model learning. On the left, there are several Gaussian component sequence samples of the word "form". After the JM model learning, the maximum likelihood cosegmentations are shown on the right. Each cosegmentation is assigned to a probability that is used in the decoding phase.

where the segmentation $a_{(s_p)}$ is with length k and the segmentation $b_{(s_q)}$ is with length l. With multiple training instances, this update function can be re-written by

$$P^{r+1}(a_{(s_p)}, b_{(s_q)}) = \frac{\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_t^r(i-k, j-l)P^r(a_{(s_p)}, b_{(s_q)})\beta_t^r(i,j)}{\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_t^r(i,j)\beta_t^r(i,j)}$$

where T is the set of training instances.

Figure 4-3 gives an example of the JM model. In this example, a JM model for letter sequence "f o r m" to Gaussian component indices sequence "19 17 34" is built. The left side is a set of training instances and the right side is the modeling result.

4.2.5 N-best Decoding

The JM decoder takes a keyword as input, and outputs the best possible segments of the keyword as well as the corresponding mapped sequences of Gaussian components. Formally, given a keyword A, the most likely Gaussian component index sequence $B^{'}$ can be calculated by

$$B^{'} = \arg\max_{B} \mathcal{L}(A, B, L^{'}) \tag{4.10}$$

where $L^{'}$ denotes the best cosegmentation found for A and B. Note that this is for maximum likelihood decoding. We can also apply a maximum a posterior (MAP) decoder as

$$B^{'} = \arg\max_{B} \mathcal{L}(A, L^{'}|B)\mathcal{L}(B, L^{'}) \tag{4.11}$$

where $\mathcal{L}(A, L^{'}|B)$ accounts for the likelihood of the match between A and B, which can be calculated by using the conditional probability $P(a_{(s_i)}, b_{(s_j)})$. The other term is $\mathcal{L}(B, L^{'})$ that defines a symbol language model for the output index sequence according to the best cosegmentation $L^{'}$. In the implementation, it can be a bi-gram or tri-gram symbol language model.

Since our decoding quality directly affects the performance of the following symbol sequence matching, we want to provide as much information as possible in the decoding result. Thus, in addition to the one-best decoder, we developed a beam search based n-best decoder. The beam width is adjustable and, in the current implementation, the setting is 200 at each decoding step.

Table 4.2 shows the 5-best decoding results for the word "artists." Since the total likelihood of the cosegmentations is a multiplicative probability, the decoder prefers a smaller number of cosegmentations. In Table 4.1, we see that the beginning "42 41" is the shared pattern of two examples. Therefore, it accumulates more counts in the maximum likelihood training. The decoding result (all top 5 begin with "42 41") confirms this shared pattern.

Table 4.2: 5-best decoding result of the word "artists." "42:20" represents that the Gaussian component 42 should be given to the first 20 frames of the keyword "artists".

	artists
1	42:20 41:9 13:9 41:1
2	42:20 41:9 41:1 13:2 26:1 41:1
3	42:20 41:9 41:1 48:1 28:1 48:5
4	42:20 41:9 13:9 13:9 26:3 13:6
5	42:20 41:9 13:9 26:1 13:14 26:1

4.2.6 Sub-symbol-sequence Matching

After the n-best sequence of Gaussian components for an input keyword is obtained, a symbol sequence matching algorithm is used to locate the possible regions in the test utterances where the keyword appears. This goal requires two properties of the matching algorithm. First, it should employ sub-sequence matching. Each n-best result represents one alternative of the keyword. The matching algorithm should operate with a sliding-window manner in order to locate a keyword in a test utterance of continuous words. Second, the algorithm must be efficient. Since the test utterance set may contain thousands of speech utterances, it is impractical for a keyword spotting system to be computationally intensive.

Smith-Waterman Algorithm

Based on these two requirements, we chose to use the Smith-Waterman (SW) [32] sub-string matching algorithm. Given two strings A and B with length m and n, this algorithm is able to find the optimal local alignment between A and B using a dynamic programming strategy. By back-tracing the scoring matrix, this algorithm can output the matched local alignment.

Formally, given two strings A, B with length m, n, we define a scoring matrix H as

$$\begin{aligned} H(i,0) &= 0, \quad 0 \leq i \leq m \\ H(0,j) &= 0, \quad 0 \leq j \leq n \end{aligned}$$

Table 4.3: The scoring matrix of "aaabbb" and "aabbbb." In calculation of the $H(i,j)$ function, a matching is given 2 points, while an insertion, deletion and mismatch is given -1 points.

		a	a	a	b	b	b
	0	0	0	0	0	0	0
a	0	2	2	2	1	0	0
a	0	2	4	4	3	2	1
b	0	1	3	3	6	5	4
b	0	0	2	2	5	8	7
b	0	0	1	1	4	7	10
b	0	0	0	0	3	6	9

where $H(i,j)$ is the maximum similarity score between the sub-string of A with length i and the sub-string of B with length j. Then, the scoring matrix H can be recursively calculated by

$$H(i,j) = \max \begin{cases} 0 & (1) \\ H(i-1,j-1) + w(A_i, B_j) & (2) \\ H(i-1,j) + w(A_i, -) & (3) \\ H(i,j-1) + w(-, B_j) & (4) \end{cases}$$

where $1 \leq i \leq m$ and $1 \leq j \leq n$. If string A is the reference string and string B is the string for matching, four cases can be developed in this recursive calculation. Case (1) ensures that all the values in this scoring matrix are positive. Case (2) represents a match or mismatch by comparing the symbols A_i and B_i. The similarity function w controls the amount of the reward (for a match) or the penalty (for a mismatch). Case (3) denotes the deletion error which indicates the target string misses one symbol by comparing with the reference string. Function w also controls how much penalty is given in this case. Case (4) is the reverse case of Case (3). It is called insertion error because this time the target string has other symbols that do not appear at the same position in the reference string. Table 4.3 shows an example of the scoring matrix of two strings "aaabbb" and "aabbbb."

After the scoring matrix H is calculated, the optimum local alignment is obtained

by starting from the highest value in the matrix. For instance, suppose we find the highest value at $H(i,j)$. Starting from here and at each step, we compare the values in $H(i-1,j)$, $H(i-1,j-1)$, and $H(i,j-1)$ to find the highest value as our next jump. If there is a tie, the diagonal jump is preferred. We continue with this process until we reach a matrix position with value zero or with coordinates $(0,0)$. By recording the back-tracing steps, an interval where both string A and string B have an optimum local alignment can be restored.

In terms of keyword spotting, the reference string becomes the decoding result (Gaussian component sequences) and the target string becomes all the test data. By running the SW algorithm throughout the test data, we can rank all the optimal local alignments by their corresponding similarity score. The higher the score is, the more likely that utterance has the input keyword.

Region Voting Scheme

In order to provide as much useful information as possible, we explored the voting of n-best results for sequence matching.

N-best results are incorporated by letting each decoding result contribute a portion of the similarity score to an utterance. Then, all similarity scores are collected for the final ranking. Some local alignments may only correspond to a part of the keyword. For instance, for the input keyword "keyword,", some optimal local alignments may point to the second syllable (sub-word) "word," while other alignments may refer to the first syllable (sub-word) "key." Therefore, it is not enough to only collect the similarity score on an utterance-by-utterance basis. We need to consider every possible region in an utterance. The basic idea is that for each utterance, if we find several local alignments for the reference sequences corresponding to the same region of an utterance, the similarity score collected for this region is reliable. We directly sum up the similarity scores for this utterance. In contrast, if we find an utterance with several local alignments, but they are all from non-overlapped regions, none of these similarity scores are used in calculating the total score of that utterance.

The region voting idea is demonstrated in Figure 4-4. In this figure, E_i denotes

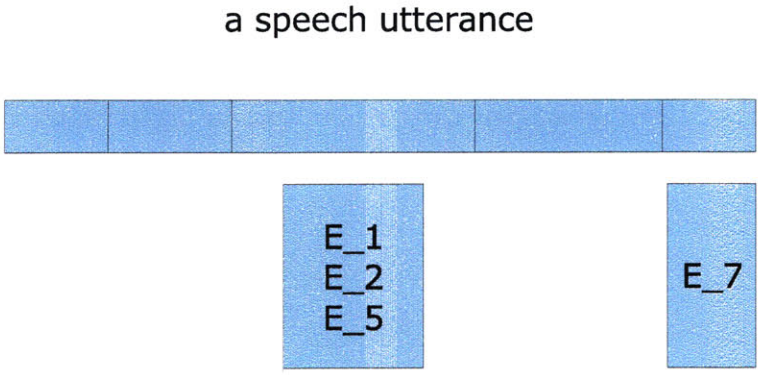

Figure 4-4: This figure illustrates a region voting example on a speech utterance. E_i denotes the similarity score contributed by the i-th keyword sample. In this example, the third segment in the utterance accumulates three scores, while the last segment only has one. Therefore, the scores on the third segment are more reliable than the score on the last segment.

the similarity score contributed by the i-th example of the keyword. We can see that the third region accumulates three similarity scores, while the last region only has one. In this case, we use the similarity score of the third region to represent the similarity score of the whole utterance. Note that in practice, since it is possible for voted regions to overlap, the final region is calculated by merging similar overlapping regions.

Specifically, the algorithm flow of the region voting scheme is as follows:

Step 1. Given a keyword, run a n-best decoder to get the set of decoding result D.

Step 2. Pick an utterance U_i in the test data.

Step 3. Pick a decoding result $D_j \in D$; run the SW algorithm $SW(U_i, D_j)$ to find an optimal local alignment a_{U_i}.

Step 4. Mark the region in that utterance U_i with the similarity score from the alignment a_{U_i}.

Step 5. Go to Step 3 until we iterate through all the decoding results.

Step 6. Check all the marked regions and merge the overlapped regions as well as their similarity scores.

Step 7. Find a region with the highest similarity score and use this score as the

final similarity score for the utterance U_i.

Step 8. Go to Step 2 until we cycle through all the test utterances.

Efficiency is considered in each step. In steps 4, 6, and 7, an efficient binary range tree is constructed for region merging and for searching for the highest similarity score.

After obtaining the similarity scores for all test utterances, we rank them by their similarity scores. The higher the rank is, the higher the probability the utterance contains the input keyword.

4.3 Experiments

Keyword spotting tasks using the proposed framework were performed on the TIMIT corpus. We give a brief overview of the experiment settings. For GMM training, a GMM with 64 Gaussian mixtures is used. After clustering, 10 broad phone classes are found. In the multigram training, a maximum of 20 iterations is used if the model does not converge. The maximum likelihood decoder produces 20-best results and all of the results are used for the following symbol matching. In the final matching result, no thresholds are used but instead all test utterances are ranked by the possibility of their containing the input keyword.

4.3.1 TIMIT Dataset

The TIMIT standard training set was used as untranscribed data, including 3,696 utterances. The test set contains 944 utterances. There are a total of 630 speakers. The vocabulary size of both the training and testing sets is 5,851.

Some word statistics from the TIMIT dataset are shown in Figure 4-5. It is clear that the word occurrence distribution follows the "long-tail" rule. The most frequent words are usually not keywords, such as articles and prepositions. These kinds of words can be viewed as stop words as in a text-based search system. According to [16], the duration of the keyword is a strong clue in the detection performance. The more syllables a keyword has, the higher the detection rate tends to be. In the TIMIT

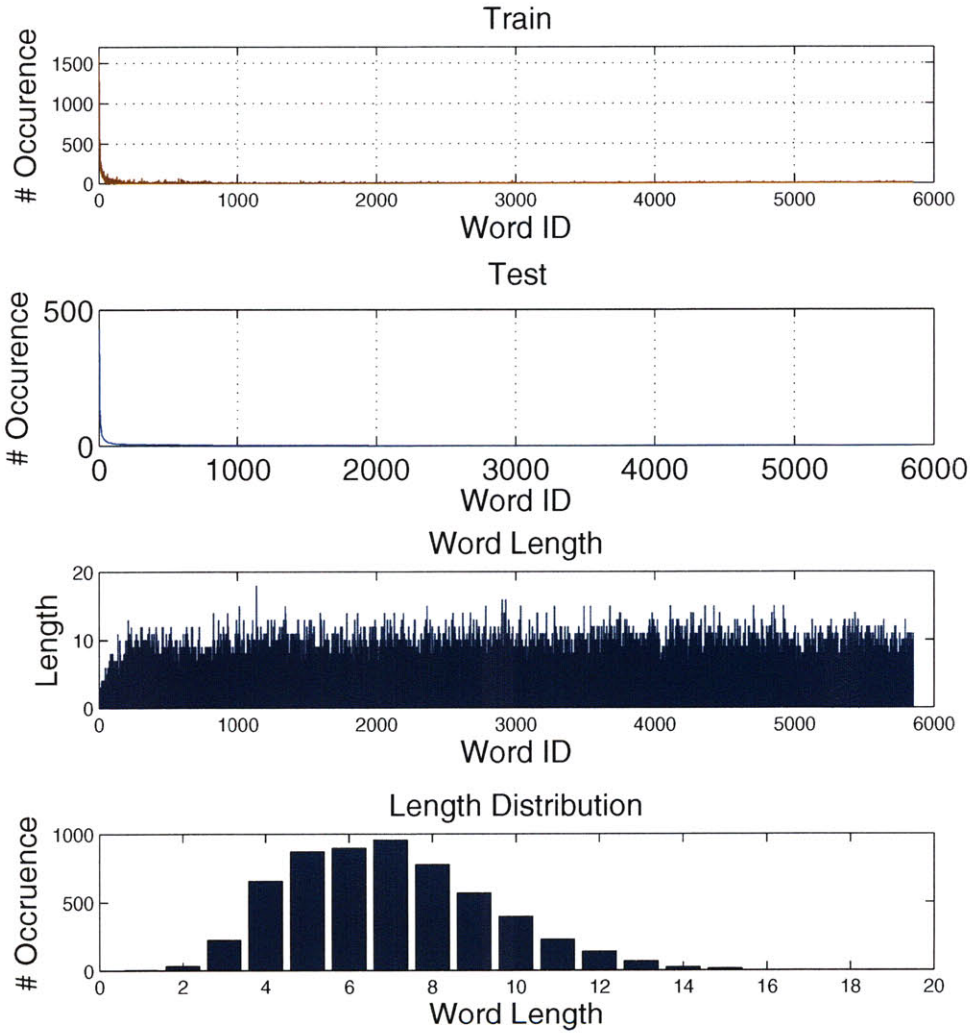

Figure 4-5: This figure illustrates word statistics on the TIMIT dataset. The first two sub-figures are the word occurrence distribution on the TIMIT training and test set respectively. The third sub-figure is the word length plot in terms of the number of syllables. The fourth sub-figure is the histogram of the third plot.

Table 4.4: 8 Keyword Set for TIMIT

Keyword	# Occ in Training	# Occ in Testing
problem	22	13
development	9	8
artists	7	6
informative	7	5
organizations	7	6
surface	3	8
otherwise	2	1
together	2	1

dataset, a large amount of words have 6-8 syllables, which is suitable for a typical keyword spotting task.

4.3.2 Keyword Spotting Experiments

Based on the number of occurrences, eight keywords are randomly selected as the set of testing keywords. The set consists of one popular word, four less popular words and three rarely seen words in terms of the number of occurrences in the TIMIT training set. A list of these keywords is given in Table 4.4. The spotting results are represented by the Receiver Operating Characteristic (ROC) curve, shown in Figure 4-6 and Figure 4-7.

Based on the keyword spotting results, the following observations can be made:

High Top 1 Hit Rate. In 5 out of 8 cases, the precision of the detection is 100% at the first choice of possible utterances containing the keyword. By looking at the detailed results, this high top 1 hit rate benefits from our voting scheme. The utterances containing the keyword are emphasized by each keyword example because the regions that may have the keyword gradually accumulate a high similarity score.

More examples are better. More examples can cover more variances of how people speak a keyword. Since the JM modeling uses the criterion of maximum likelihood in estimating the parameters, the more training instances it has, the more accurate the model can be. Having a better JM model, the decoding quality also improves. Since

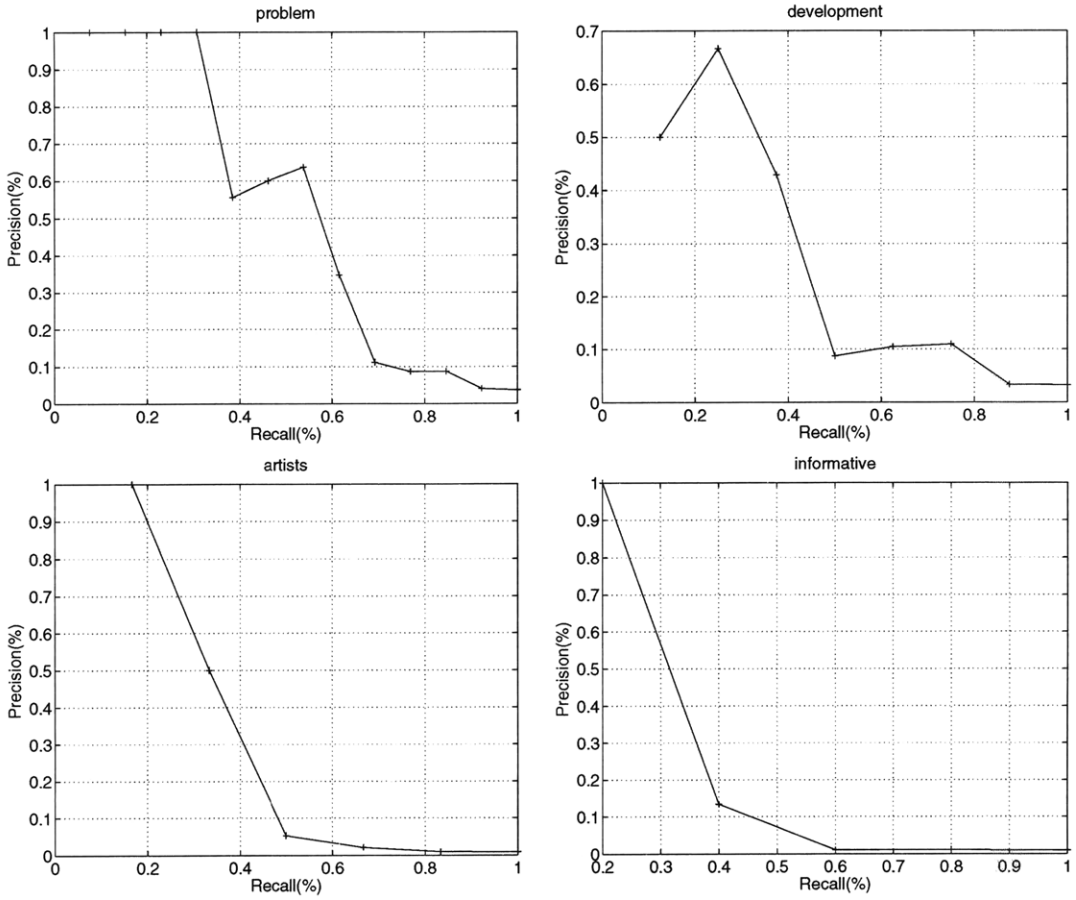

Figure 4-6: These four figures illustrate the ROC curves of four keyword spotting results on the TIMIT dataset. It is clear that a high top 1 hit rate can be obtained, while some keyword occurrences are difficult to detect.

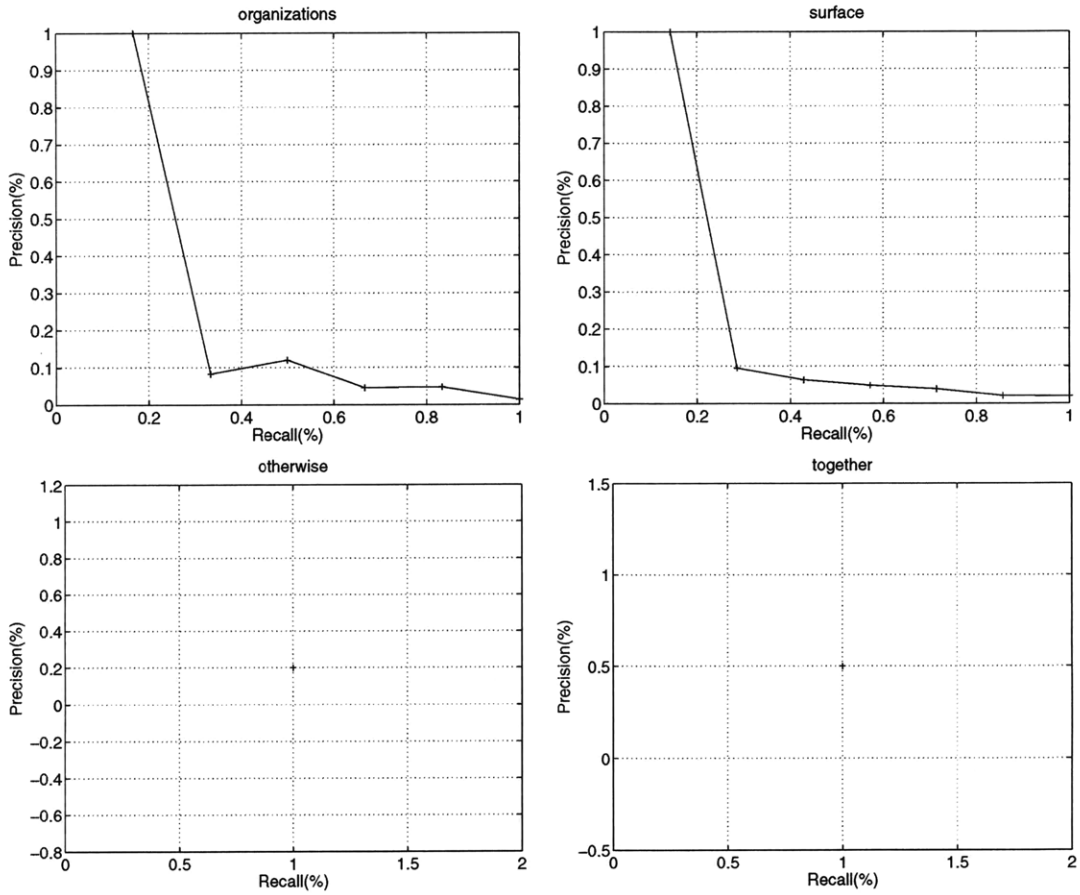

Figure 4-7: These four figures illustrate the ROC curves of another four keyword spotting results on the TIMIT dataset. Note that for the last two keywords "otherwise" and "together," since their number of occurrences in the test set is only 1, the ROC curve is a single point in the middle.

the symbol matching is a static process, the improved decoding result directly affects the matching results.

Even if the training set contains only one or two examples, the system can still have a hit. Although the last two keywords "otherwise" and "together" only appear twice in the training set, the system still provides a detection precision of 20% and 50%. Note that in these cases, since there are not enough examples that can be used to train the JM model, we directly use the training examples as the decoding result (flat model) for the symbol matching.

Some examples are difficult to detected. In all cases, it is difficult to achieve a high recall rate while maintaining a high precision, which indicates there are many examples in the test data that cannot be discovered by only using the decoding results. This is reasonable for the TIMIT dataset because 1) the speakers in the test set are completely different from the speakers in the training set; and 2) the utterances used in the test set are completely different from the training set. As mentioned before, the pronunciation of a keyword depends on the speakers as well as the context. As a result, these hard examples that cannot be easily detected may be caused by either or both of these two issues. By looking at the detailed voting results, we find that some hard examples are completely different from the training examples. For example, in detection of the keyword "informative," we found an example whose last several frames are all labeled with cluster 19, while all training examples end with cluster 48.

4.3.3 Performance Comparison

We also compared the detection performance for six keywords against the system we proposed in Chapter 3. We plot the Receiver Operating Characteristic (ROC) curve shown in Figure 4-8. The legend MG stands for the multigram based method, while GP stands for the Gaussian posteriorgram based method. It is clear that each keyword decreases in detection performance. Specifically, some keyword occurrences appear to be extremely difficult for detection using the multigram method, while they are easy to detect when providing audio samples, e.g., for the keyword "development," "organizations" and "surface." This result is reasonable because the

Table 4.5: Top 5 errors for the keyword "problem," "development," "artists" and "informative."

problem	development	artists	informative
possible	whole human	a tissue	untimely
downward	dull gleam	was stranded	employment
accomplished	predominantly	economic	at twilight
pond	grow more	meats test	introduced
bonfire	coincided with	glistened	important

Table 4.6: Top 5 errors for the keyword "organizations," "surface," "otherwise" and "together."

organizations	surface	otherwise	together
sterilization	substances	all the while	at gunpoint
musicians	sadness	others are	expense
radiosterilization	sometimes	economic	alligators
accusations	experts	other	vocabulary
rehabilitation	services	always use	time they're

multigram based method does not require any audio keyword samples and is much harder than the Gaussian posterior based method. However, for some easy keyword occurrence, multigram based method can still have a good detection rate.

4.3.4 Error Analysis

We performed an error analysis by looking at the ranking result of output of the voting scheme. The top 5 errors for each keyword are listed in Table 4.5 and Table 4.6. There are two types of errors that can be found in these two lists:

1) Words or phrases with similar pronunciations. For example, the word "possible[pɔsəbəl]" is mistakenly identified as "problem[prɔbləm]," while the phrase "a tissue[a tiʃu]" is identified as "artists[artists]." Furthermore, all of the top 5 errors for the keyword "organizations" fall into this error category mainly caused by the suffix "-ations." The voting scheme seems to be particularly sensitive to shared prefix or suffix errors, because every keyword example may contribute to a similarity score to the same region of the utterance, which accumulates the chance of that utterance

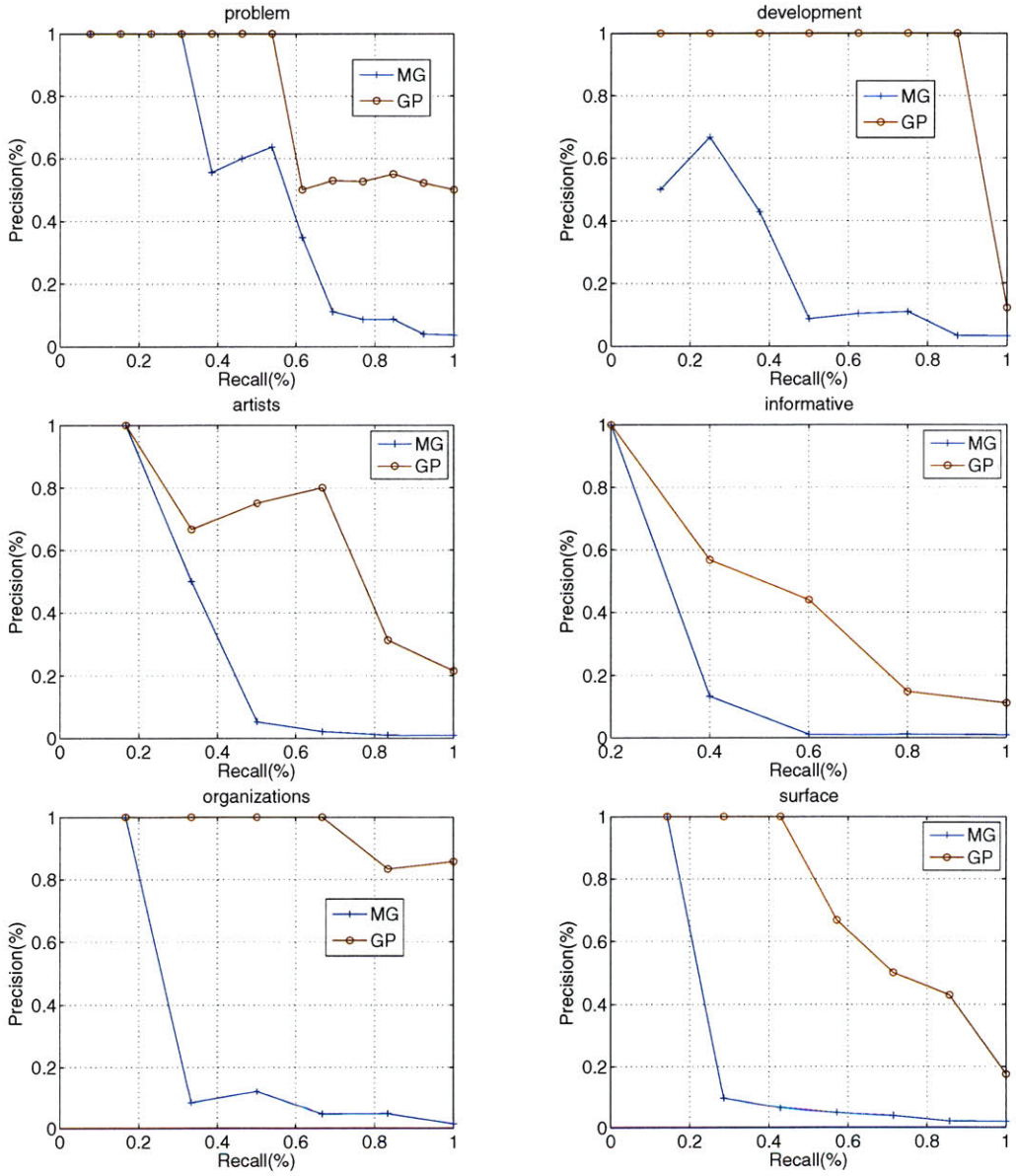

Figure 4-8: These six figures illustrate the performance comparison of two keyword spotting systems. In each figure, MG (in blue) stands for multigram based method, while GP (in red) stands for Gaussian posteriorgram based method. It is clear that in all cases Gaussian posteriorgram based method outperforms the multigram based method.

containing that keyword.

2) Words or phrases containing similar broad phone classes. Since the Gaussian component labeling is generally equivalent to assigning a broad phone class label to every frame in an utterance, it is possible for two completely different words to have similar sequences of broad phone class labels. For example, in the error list of the keyword "problem," the word "accomplished" is picked because the part "-ccompli-[28 19 26 14]" has the same broad phone class sequence with "proble-[28 19 26 14]." It is the same for the word "vocabulary" in the error list of the keyword "together" (the part "vocab[26 45 28]" and "toge[26 45 28]").

We can hypothesize that the main cause of these two types of errors is due to limited acoustic model resolution. Specifically, the number of both error types can be reduced by increasing the information contained in the broad phone class labels. If we can model the broad phone classes in a more detailed way, it may be possible to differentiate the words or phrases with similar pronunciations as well as broad phone class representation. However, the modeling resolution problem is a double-edged sword. Due to speaker variance, increasing model resolution could be used to model the differences caused by different speaking styles. This may even increase the noise in the symbol matching module because a keyword now can have many completely different Gaussian component sequences, such as one sequence for male speakers and one sequence for female speakers. Therefore, how to determine a suitable model resolution has trade-offs, which may require further experiment and investigation.

4.4 Discussion

This chapter has presented an unsupervised framework to do keyword spotting in speech data without keyword audio examples. The framework first trains a GMM to label speech frames in an unsupervised way. Then, a clustering algorithm groups similar GMM components to make a smaller set of representative GMM component clusters. As a result, all speech utterances can be labeled in a frame-by-frame way by cluster indices. Time aligned text based examples of keywords are given to build a

Joint-Multigram model that is used to find a mapping from letters in the keyword to the clustered Gaussian component labels. A voting based symbol matching module is then used to find the possible regions containing the input keyword in test utterances. By running the experiments on the TIMIT dataset, we demonstrate the possibility of using only text based samples to perform the keyword spotting task in a speaker independent environment. A good top 1 hit rate is achieved and the detection precision is reasonably well even though there are only very few training examples. In the error analysis, we find that the unsupervised modeling resolution might affect the keyword detection performance. How to set the suitable resolution is a trade-off between the detection recall rate and the amount of noise brought by the speaking variance of speakers. In addition, theoretically the multigram model can be trained on any word aligned data other than keyword examples. However, since it uses an EM training strategy which requires a large amount of data, in the current implementation, we only use the keyword examples to train a small multigram model. We will leave the large scale multigram model training for future work.

Chapter 5

Unsupervised Learning of Acoustic Units

5.1 Introduction

In the previous two chapters, unsupervised GMM learning played a critical role in both of the keyword spotting systems. We are naturally interested to investigate the underlying behavior of unsupervised GMM training. Since the success of the keyword spotting task has demonstrated that the Gaussian posteriorgrams as well as the Gaussian cluster representation can be used to label speech frames, it is helpful to understand what acoustic patterns/events can be found by only providing speech signals without any transcription.

In this chapter, we first develop a statistical method to investigate the learning ability of unsupervised GMM training. A phonetic histogram analysis is applied to extract acoustically meaningful information for each learned Gaussian component cluster. The basic idea is to apply unsupervised GMM training to the TIMIT dataset. After the training converges, a clustering algorithm is used to cluster the learned Gaussian components. Each speech frame is then labeled by the most probable Gaussian component cluster. Since the TIMIT dataset provides time aligned phonetic transcription, a phonetic histogram can be drawn for each Gaussian component cluster. Specifically, each bar in a histogram represents a pre-defined phone. The

height of a bar represents the normalized portion of how many times the Gaussian component cluster represents that phone. By looking at the histogram of a Gaussian component cluster, we can hypothesize the broad acoustic class each Gaussian component cluster may represent.

In the second part of this chapter, taking the phonetic histogram analysis one step further, we explore some unsupervised methods to help determine broad acoustic classes without any phonetic transcription. We develop three acoustic feature detectors for detecting voiced regions, sonorants as well as obstruents. The detection results are used to give an acoustic label to each speech frame. The acoustic labels are then combined with Gaussian component cluster labels to produce an acoustic feature histogram for each Gaussian component cluster. Based on the resultant histograms, a decision tree based method is used to infer the acoustic meanings of each acoustic feature histogram as well as to hypothesize the broad acoustic class each histogram represents. The evaluation was performed on the TIMIT dataset. The comparison of the ground truth shows that the proposed unsupervised broad acoustic class modeling framework can differentiate vowels, semi-vowels, fricatives, stops and closures.

5.2 Phonetic Histogram Analysis

5.2.1 GMM Training and Clustering

The same unsupervised GMM training and clustering strategy is used to give each speech frame a Gaussian component cluster label. Specifically, all speech frames in the TIMIT dataset are sent to unsupervised GMM training. The number of Gaussian components is set to 64. After the GMM is trained, the Affinity Propagation algorithm is used to cluster the Gaussian components. Then, each speech frame in both the training and the test set is labeled by a cluster index. Since the TIMIT corpus has time-aligned phonetic transcription, for each Gaussian component cluster and a specific phone, we can calculate how many times each cluster represents a phone. Based on the calculation, a histogram can be drawn for each Gaussian component

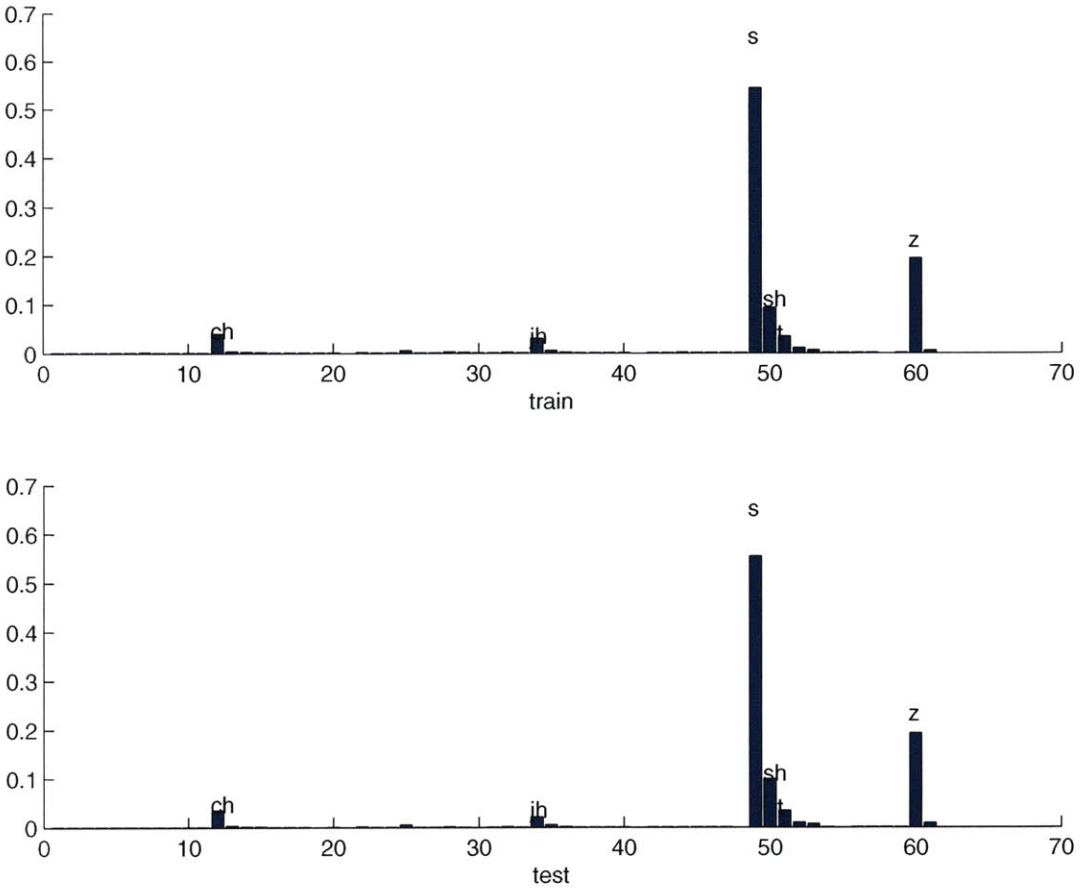

Figure 5-1: This figure illustrates the histogram of the Gaussian component cluster 13. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. It is clear that this cluster mainly represents fricatives and affricates.

cluster.

5.2.2 Phonetic Histograms

Figure 5-1 shows the phonetic histogram of the Gaussian component cluster 13. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. It is clear that this cluster mainly represents fricatives and affricates.

Figure 5-2 illustrates the phonetic histogram of another Gaussian component cluster 41. Since most bars are labeled as vowels or semi-vowels, this cluster mainly represents vowels and semi-vowels.

Figure 5-3 illustrates the phonetic histogram of the Gaussian component cluster 14. This cluster mainly represents retroflex.

Figure 5-4 illustrates the phonetic histogram of Gaussian component cluster 19. This cluster mainly represents stop closures.

Figure 5-5 illustrates the phonetic histogram of the Gaussian component cluster 26. This cluster mainly represents fricatives, some stops and their closures.

Figure 5-6 illustrates the phonetic histogram of the Gaussian component cluster 28. This cluster mainly represents stop closures and nasals.

Figure 5-7 illustrates the phonetic histogram of the Gaussian component cluster 42. This cluster mainly represents low vowels.

Figure 5-8 illustrates the phonetic histogram of Gaussian component cluster 45. This cluster mainly represents high vowels.

Figure 5-9 illustrates the phonetic histogram of the Gaussian component cluster 48. This cluster mainly represents nasals except for some vowels and retroflex.

Figure 5-10 illustrates the phonetic histogram of the Gaussian component cluster 60. This cluster mainly represents semi-vowels and low vowels.

After investigating all histograms, we find that each Gaussian component cluster takes responsibility for a small set of phones. Phones in a set share similar acoustic characteristics. Therefore, each phone set can be called a broad acoustic class. As a result, with the help of phonetic transcription, the phonetic histogram analysis can provide a way of determining the broad acoustic class that each Gaussian component cluster represents.

This analysis also provides a fundamental viability of why the unsupervised keyword spotting frameworks can work. Without any transcription, unsupervised GMM training and clustering are able to automatically extract some acoustically meaningful information from speech signals only. When the learned knowledge is used to label speech frames, it can provide enough information for tasks such as keyword spotting.

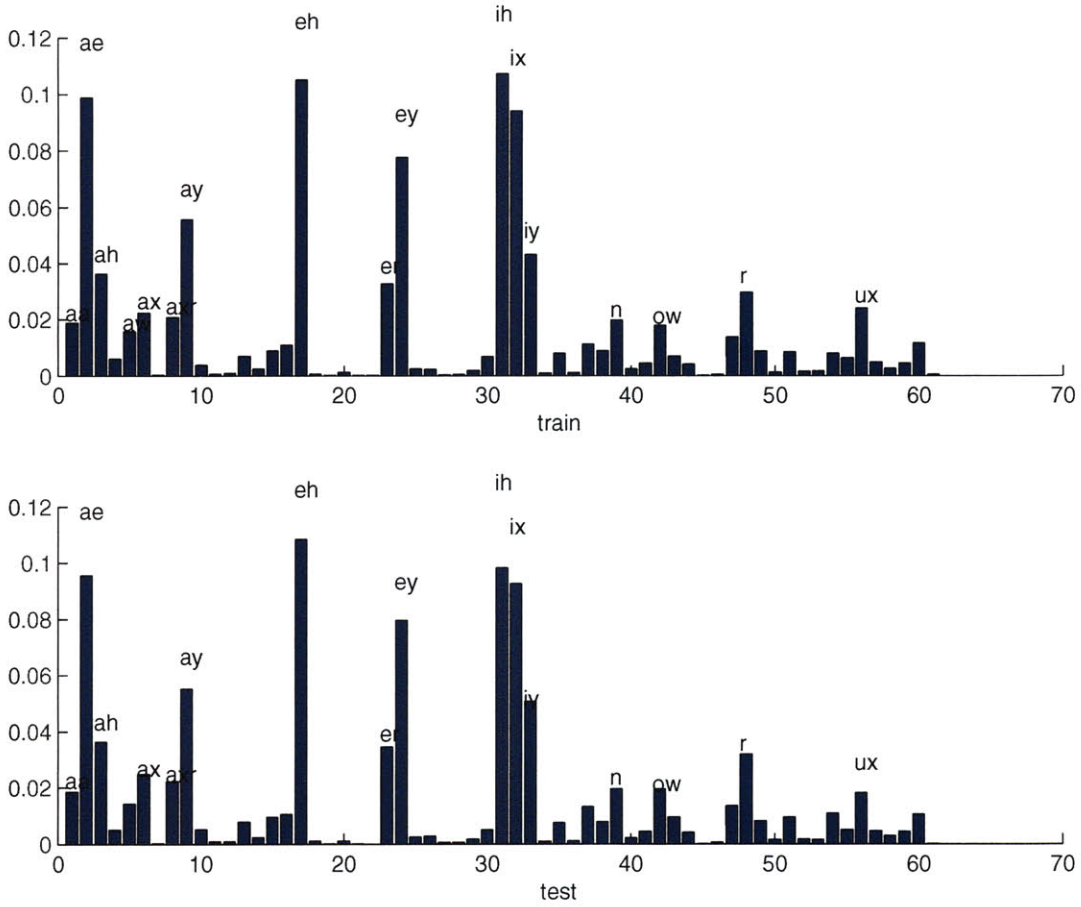

Figure 5-2: This figure illustrates the phonetic histogram of the Gaussian component cluster 41. Each bar in the figure represents the portion of times that this cluster represent a phone. The tob sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. Since most bars are labeled as vowels or semi-vowels, it is clear that this cluster mainly represents vowels and semi-vowels.

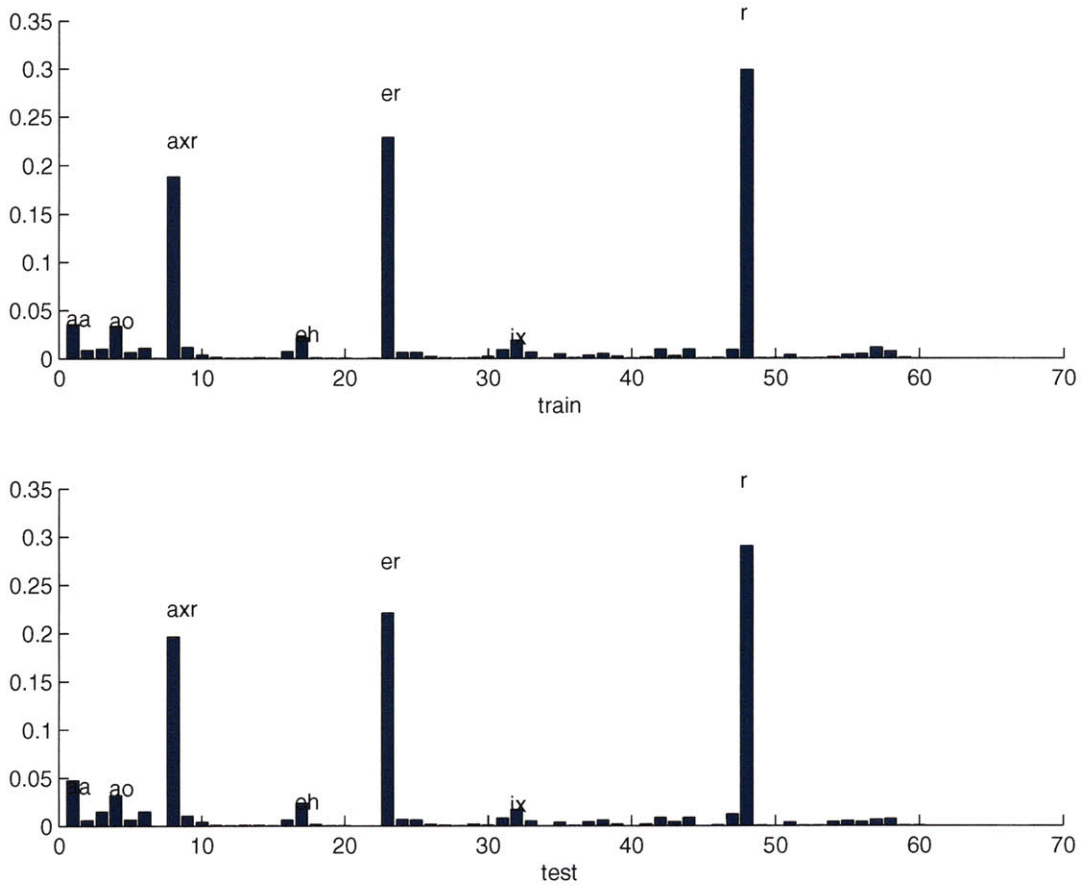

Figure 5-3: This figure illustrates the phonetic histogram of the Gaussian component cluster 14. Each bar in the figure represents the portion of times that this cluster represent a phone. The top sub-figure is the histogram on the training set, while the bottom sub-figure is the histogram on the test set. This cluster mainly represents retroflex.

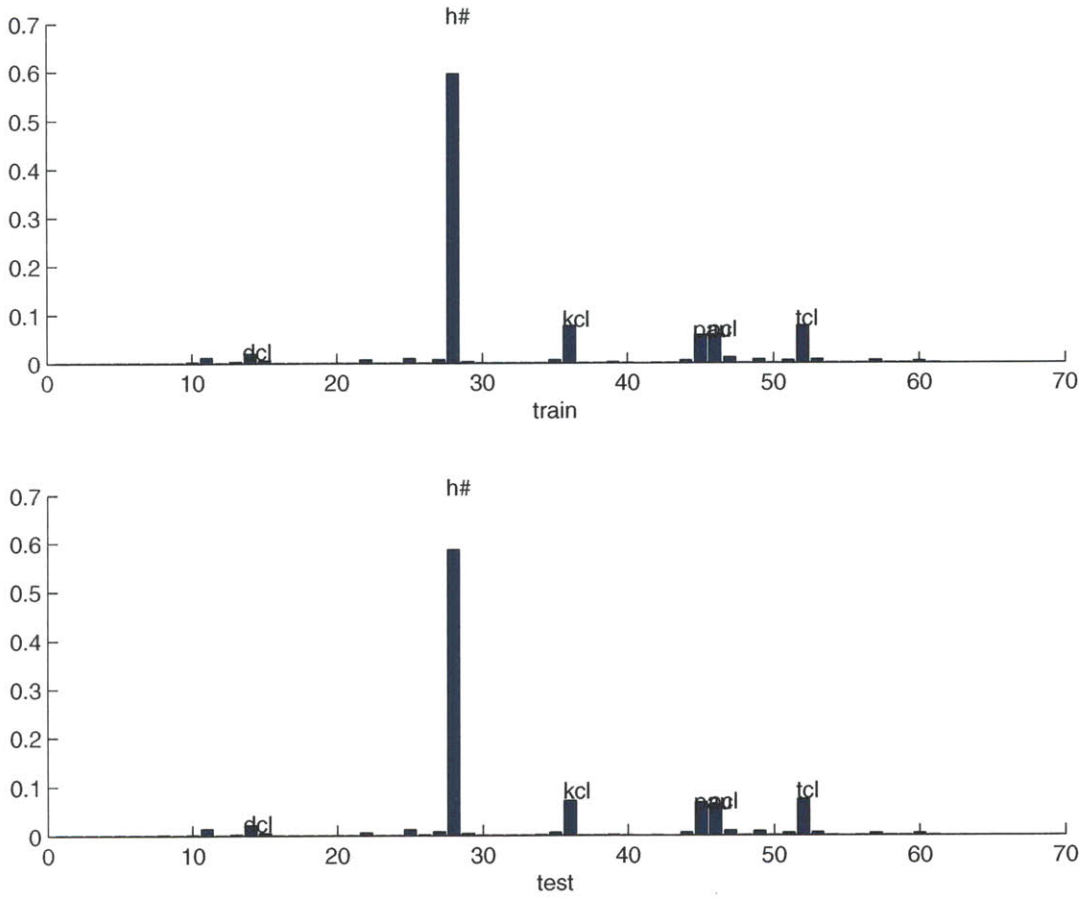

Figure 5-4: This figure illustrates the phonetic histogram of the Gaussian component cluster 19. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents stop closures.

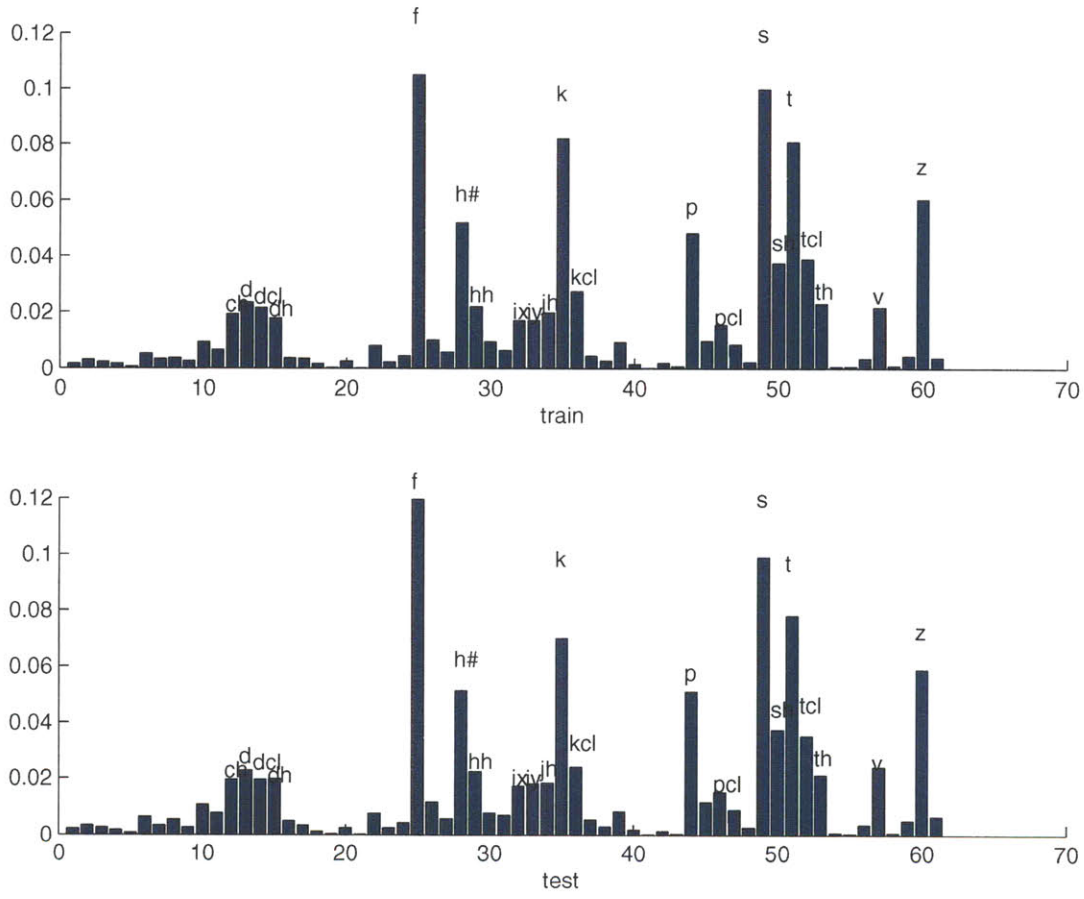

Figure 5-5: This figure illustrates the phonetic histogram of the Gaussian component cluster 26. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents fricatives, some stops and stop closures.

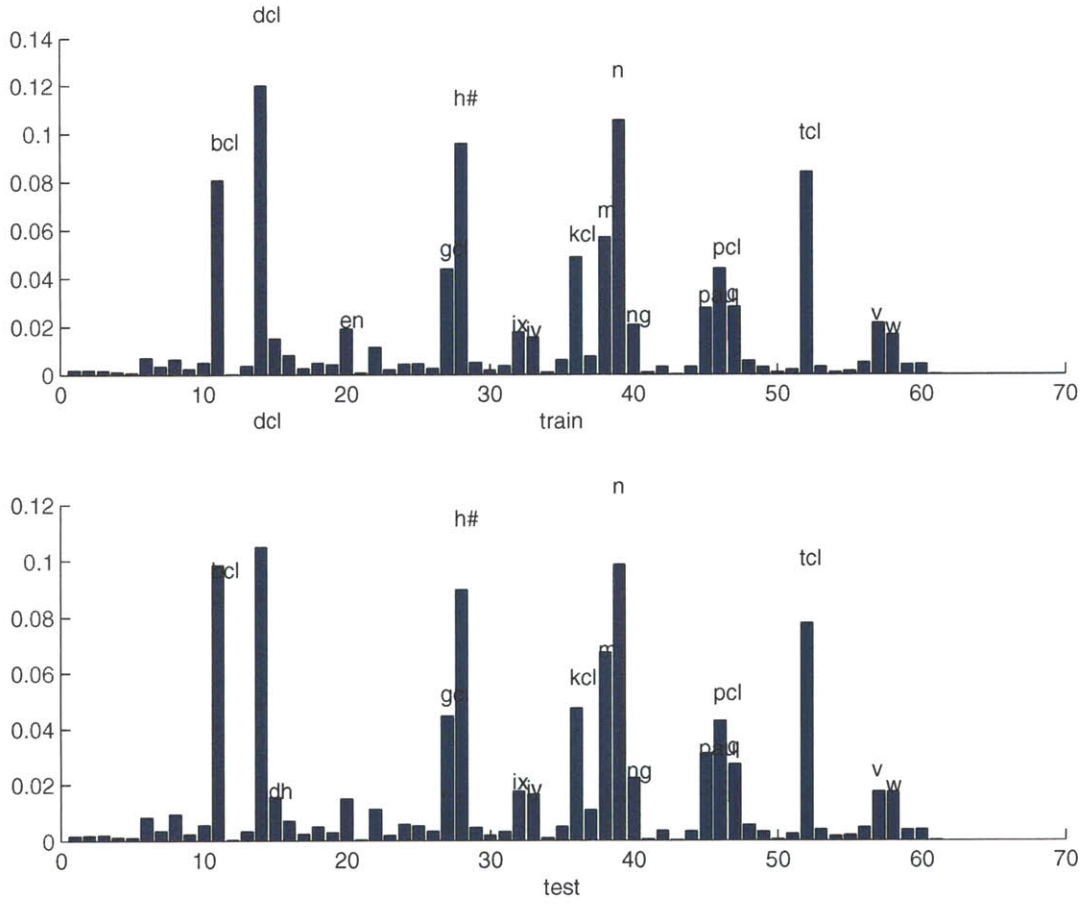

Figure 5-6: This figure illustrates the phonetic histogram of the Gaussian component cluster 28. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents stop closures and nasals.

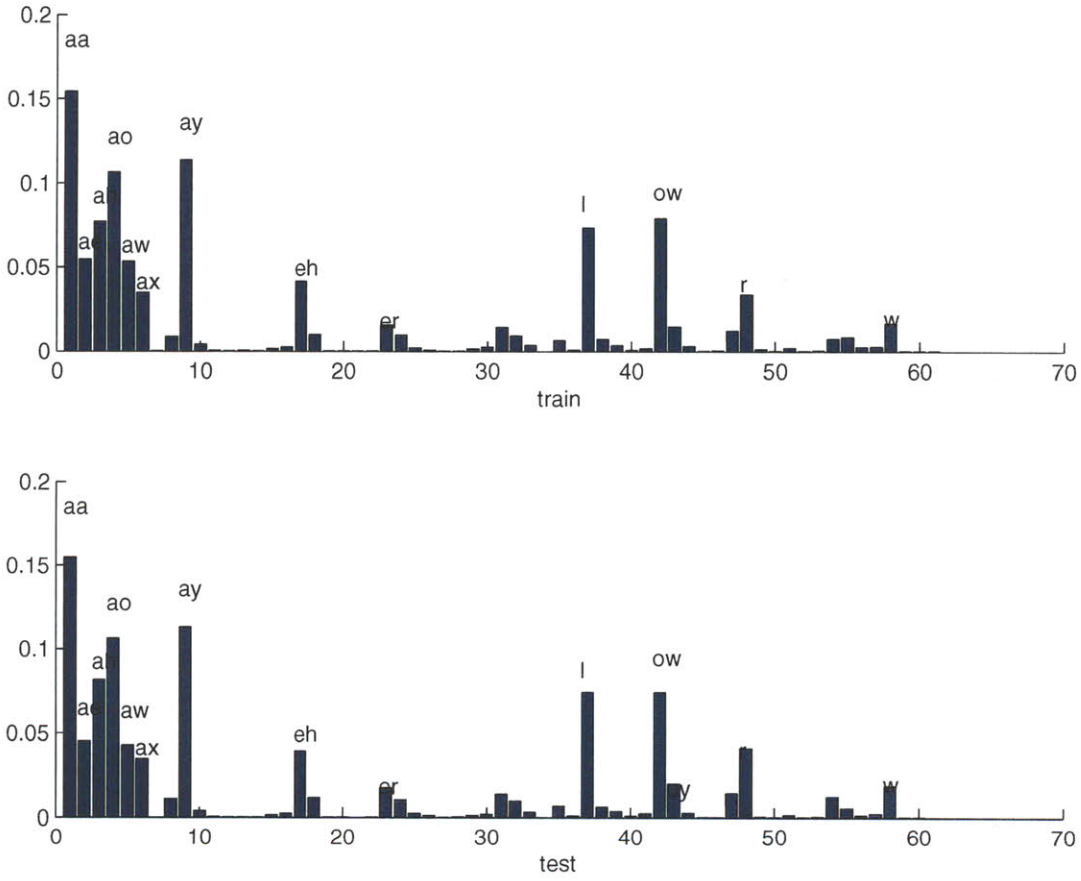

Figure 5-7: This figure illustrates the phonetic histogram of the Gaussian component cluster 42. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents low vowels.

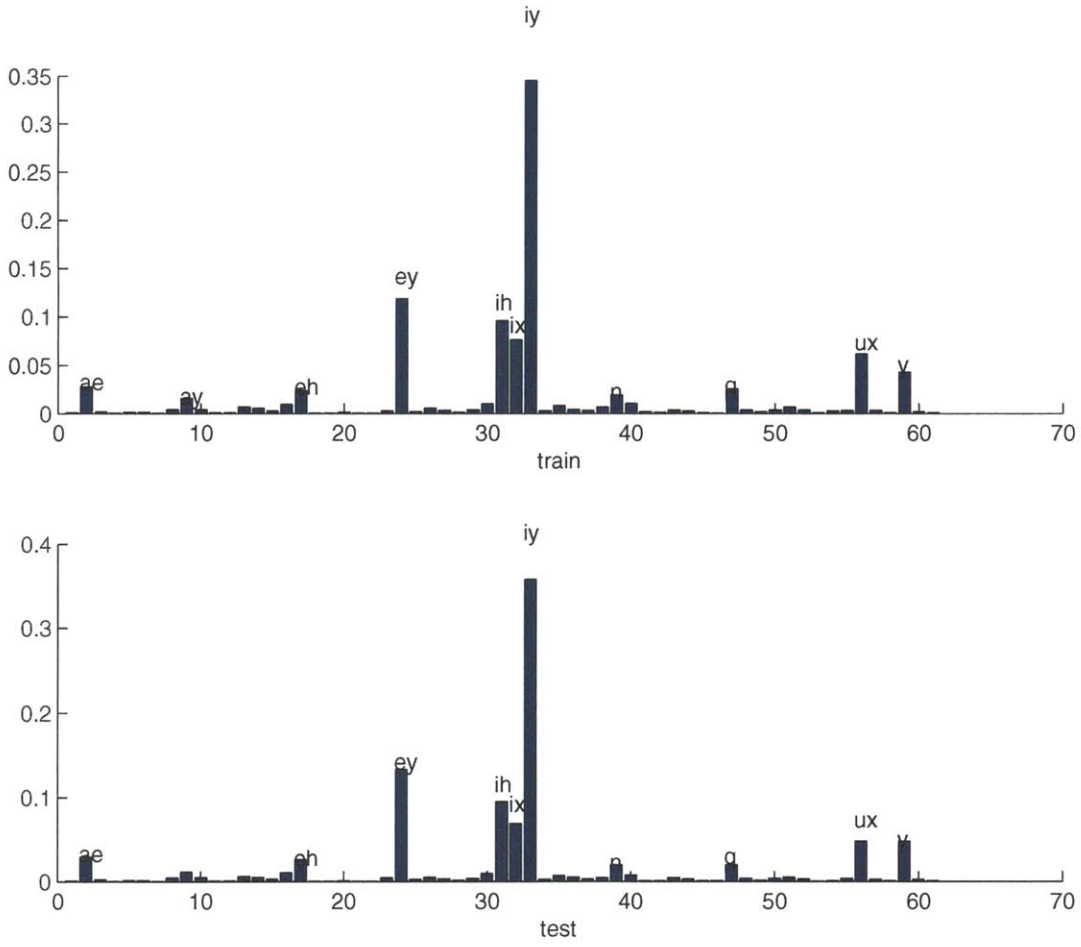

Figure 5-8: This figure illustrates the phonetic histogram of the Gaussian component cluster 45. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents high vowels.

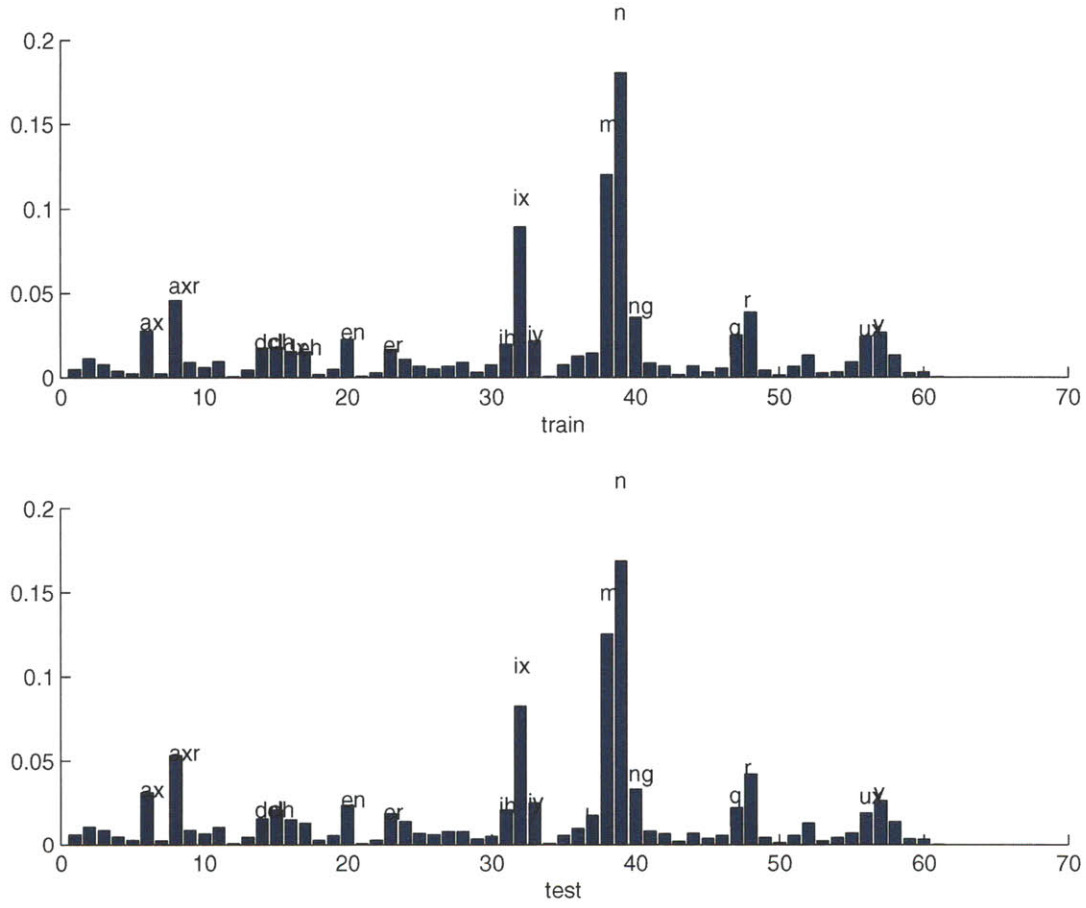

Figure 5-9: This figure illustrates the phonetic histogram of the Gaussian component cluster 48. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents nasals except for some vowels and retroflex.

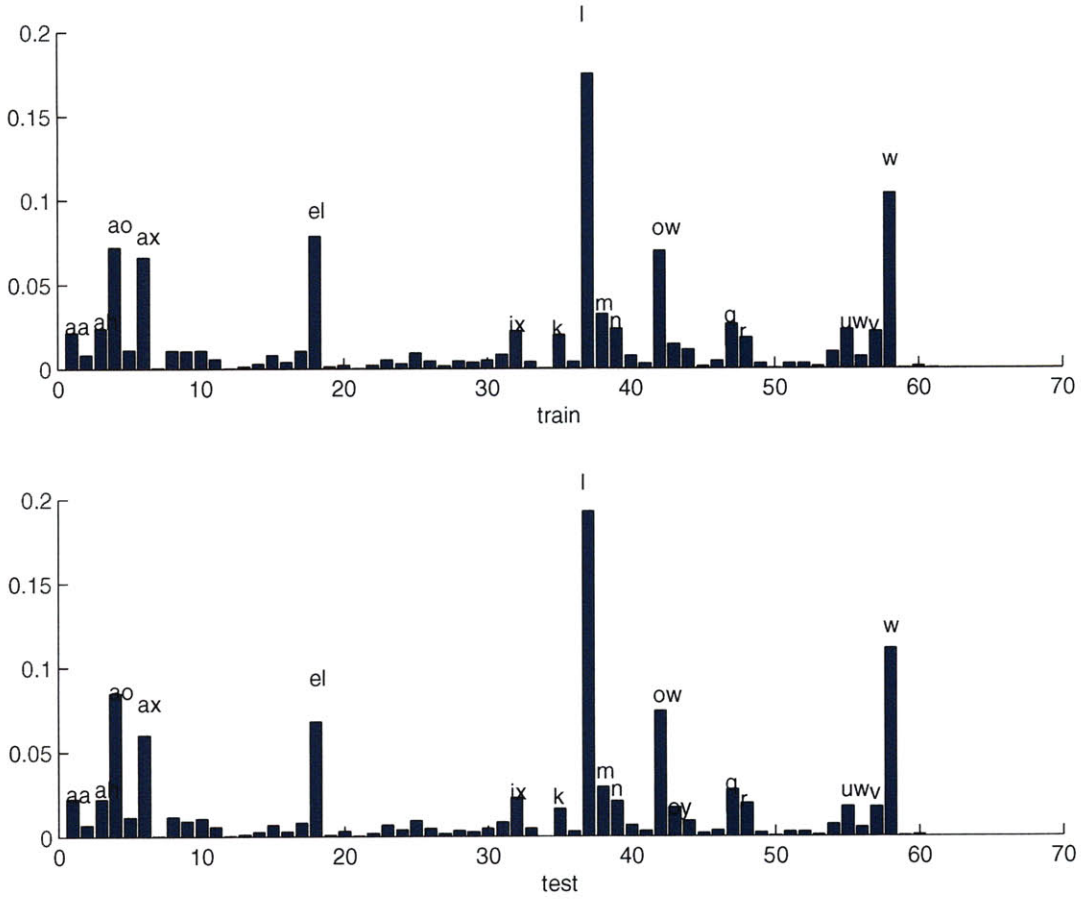

Figure 5-10: This figure illustrates the phonetic histogram of the Gaussian component cluster 60. Each bar in the figure represents the portion of times that this cluster represents a phone. The top sub-figure is the histogram of the training set, while the bottom sub-figure is the histogram of the test set. This cluster mainly represents semi-vowels and low vowels.

5.3 Unsupervised Broad Acoustic Class Modeling

As mentioned in the last section, unsupervised GMM training and clustering can provide a broad acoustic class finding framework. However, understanding what acoustic class each Gaussian component cluster represents still needs a time-aligned phonetic transcription. It is natural to explore whether there is an unsupervised way to hypothesize the broad acoustic class that each Gaussian component cluster represents. In this section, we describe some preliminary work to address this problem.

5.3.1 Framework Overview

The unsupervised broad acoustic class modeling framework can be divided into three steps. The first step labels each input speech frame with Gaussian cluster index. As a result, a speech utterance is converted to a series of Gaussian cluster indices. In the second step, three different acoustic feature detectors are applied to every speech utterance. In the third step, we aggregate the acoustic feature labels with the Gaussian component cluster labels and use a decision tree based method to hypothesize the broad acoustic class that each Gaussian cluster represents. The overview of the proposed system is shown in Figure 5-11.

5.3.2 Unsupervised Acoustic Feature Analysis

Since no transcription is provided, it is necessary to extract some acoustically meaningful features directly from the speech signal. We designed three different acoustic feature detection modules. These three acoustic detectors are employed to locate three acoustic features on speech frames, including vowel centers, obstruents and voicing regions. Each detector takes a speech utterance as input and outputs the time points (for vowel center and obstruent detector) or time intervals (for voicing detector). The reason why these features are selected is that they basically cover the most reliable acoustic clues for determining a broad acoustic class. In the following sections, we give a detailed description of the three detectors.

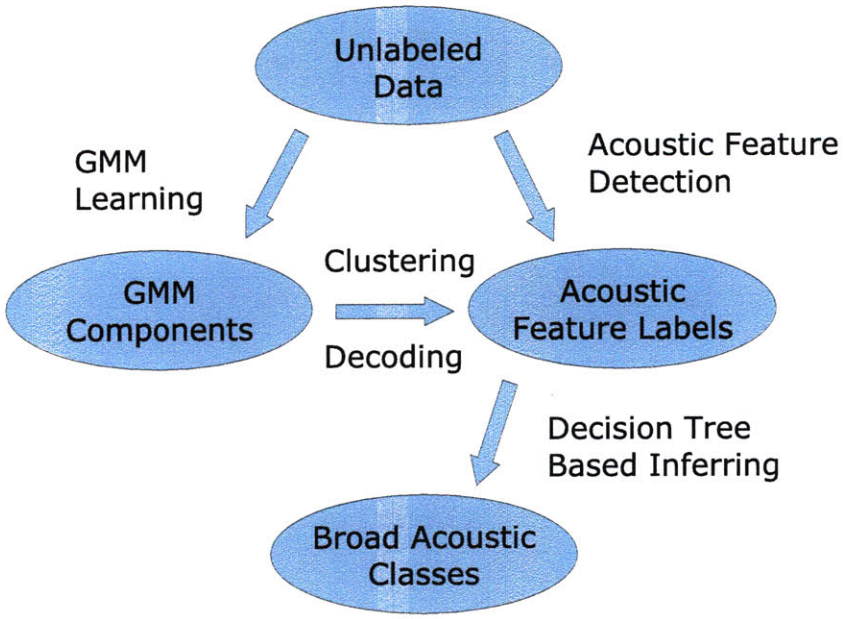

Figure 5-11: This figure illustrates the overview of the broad acoustic class modeling framework. The unsupervised broad acoustic class modeling framework can be divided into three steps. The first step labels each input speech frame with Gaussian cluster index. As a result, a speech utterance is converted to a series of Gaussian cluster indices. In the second step, three different acoustic feature detectors are applied to the input utterance. In the third step, we aggregate the acoustic feature labels with the Gaussian component cluster labels and use a decision tree based method to hypothesize the broad acoustic class each Gaussian cluster represents.

5.3.3 Vowel Center Detection

In vowel center detection, a speech rhythm guided syllable nuclei detection method is employed [46]. The basic idea is to use rhythm information to locate syllable nuclei in a non-parametric way. Since the rhythm based method requires no parameters, it is a completely unsupervised process. No training data is needed. Since syllable nuclei appear mainly around the center of vowels and semi-vowels, we can directly use this syllable nuclei detection module to locate vowel centers.

Detection Algorithm Overview

The speech-rhythm guided syllable nuclei detection algorithm can be divided into two main stages. In the first stage, we apply an envelope analysis to the input speech

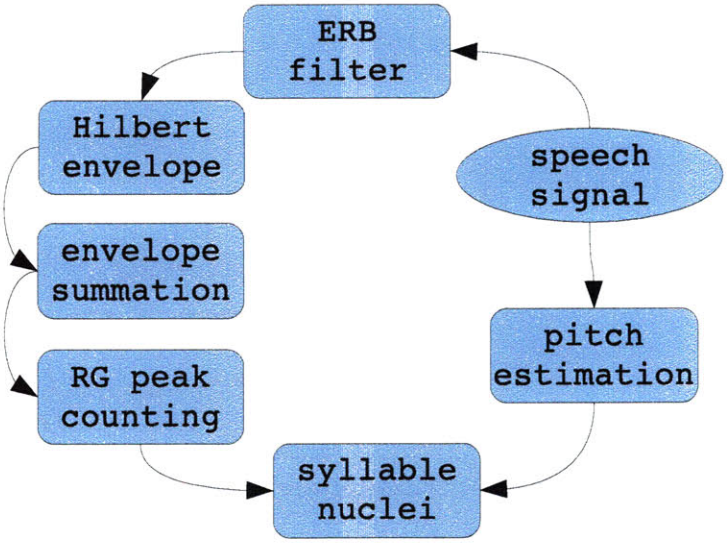

Figure 5-12: This figure illustrates the algorithm flowchart for the rhythm based syllable nuclei detection system. RG stands for our rhythm guided method. The speech-rhythm guided syllable nuclei detection algorithm can be divided into two main stages. In the first stage, we apply an envelope analysis to the input speech signal. In the second stage, we estimate the speech rhythm on the signal envelope to help with the syllable nuclei detection.

signal. In the second stage, we estimate the speech rhythm on the signal envelope to help with the syllable nuclei detection. A flowchart of the proposed algorithm in shown in Figure 5-12. Each of the stages is described in more detail in the following three subsections.

Envelope Analysis

To filter out noise at different frequency levels, the peripheral auditory based band-pass filter has been widely used in analyzing speech signals. In our approach, we use an equivalent rectangular bandwidth filter (ERB) to first split the wave signal into 20 channels [22]. The ERB filter can be viewed as a series of gammatone filters operating on the ERB rate scale. The output signal of the i-th ERB filter is

$$x_i(t) = t^{m-1} \exp(-2\pi b_i t) \cos(2\pi C F_i^{nor}) H(t) \tag{5.1}$$

where m is the filter order (in our implementation, we set $m = 4$), b_i is the bandwidth,

CF_i^{nor} is the center frequency converted from the ERB scale, and $H(t)$ is the unit step signal.

After band-pass filtering, we extract the envelope $E_i(t)$ of each channel. Let $x_i(t)$ be the signal in the i-th channel. To compute the envelope, we apply the Hilbert transform on $x_i(t)$ to calculate the magnitude $X_i(t)$ as

$$X_i(t) = x_i(t) + i \cdot \mathcal{H}(x_i(t)) \tag{5.2}$$

where $\mathcal{H}(\cdot)$ denotes the Hilbert transform. The envelope of the i-th channel can be calculated by

$$E_i(t) = |X_i(t)| \tag{5.3}$$

In order to reinforce the energy agreement of each channel, we first normalize each channel envelope and then sum them to have the total envelope representation $E_i(t)$ of the input speech signal

$$E(t) = \sum_{i=1}^{N} E_i(t) \tag{5.4}$$

where N is the number of channels.

Unlike some previous methods, we do not choose to do sub-band selection, and we use direct summation instead of a sub-band temporal correlation of energy envelopes. There are two main reasons for this. First, since we are using a two-stage method, the more sub-bands we use to contribute to the total envelope, the more information we can use for speech-rhythm estimation. Second, we noticed that with temporal correlation, energy peaks may occur a little bit later than their actual location. The amount of delay is dependent on the length of the correlation window. Although this delay does not affect the total number of peaks, it does affect the accuracy of our subsequent speech-rhythm estimate. Therefore, we chose not to perform a correlation so as not to interfere with the second stage processing.

Rhythm Guided Peak Counting

A perfect peak counting algorithm would find all true positive peaks and make no false predictions. In most peak picking methods, there are two important parameters that determine peak selection [17]. The first is the ratio of the height of a peak over the highest peak in the current working set. The other is the ratio of the contrast of a peak with its immediate neighboring valleys. The first parameter uses global information while the second parameter may vary on a case by case basis, even within an utterance. In addition, due to the various sources of noise, it can be difficult to find parameter thresholds that avoid all spurious peaks while detecting all correct peaks. Thus, we seek to use the speech rhythm information to avoid requiring these two parameters in processing the spectral envelope. The basic procedure is to use a conventional method to find the first two syllabic nuclei via envelope peaks; then estimate the instantaneous speech rhythm based on these two nuclei; and then subsequently predict intervals where the next syllable nucleus may appear; finally, we use a simple slope based peak detection algorithm in each interval that avoids the use of any parameters or thresholds. This simple peak detection and speech rhythm estimation are performed repeatedly until the end of the utterance is reached.

To clearly and efficiently represent speech rhythm, we turn the speech rhythm estimation into a sinusoid function fitting problem. Given a peak location set P, we fit a sinusoid function F_{k_1,k_2}, with frequency, k_1, and phase offset, k_2, of which peak locations are matched to the peak locations in P. The target sinusoid function is

$$F_{k_1,k_2}(x) = sin(k_1 \cdot x + 2\pi \cdot k_2) \tag{5.5}$$

By using a least mean squares fitting scheme, the objective function can be written as

$$\{k_1, k_2\} = \arg\min_{k_1,k_2} \frac{1}{|P|} \sum_{i=1}^{|P|} (1 - F_{k_1,k_2}(p_i))^2 \tag{5.6}$$

where $p_i \in P$ denotes the location of peak p_i and $k_2 \in [0, 1)$. Using these notations, a stepwise description of the syllable nuclei detection method is as follows:

Step 1 After the i-th peak p_i is found, add p_i into P

Step 2 Based on the observed peaks in P, perform least mean squares fitting on P to calculate the current best k_1^i and k_2^i

Step 3 Align all the p_i in P to the nearest x_j and find x_{p_t} representing the sinusoid peak location to which the newly found p_t is assigned

Step 4 Calculate the smallest x_s where $F_{k_1^i,k_2^i}(x_s) = 1$ and $x_s > x_{p_t}$

Step 5 Run slope based peak counting within the range $[x_{p_t}, x_s + \frac{3\pi}{k_1^i}]$. If we have multiple peak candidates, pick the maximum one. Thus, we only allow one peak in this range.

Step 6 If a peak is found, go to Step 1. If not, set $x_{p_t} = x_s$ and go to Step 5. Repeat until reaching the end of the utterance.

Note that we need at least two peaks to estimate the first set of k_1 and k_2. We initialize k_1 to be $\frac{2\pi}{k_1} = 100ms$ to avoid the uninteresting solution of large k_1. We tried both the standard contrast based and simple slope based peak counting algorithms and found that the selection of these two algorithms has little effect on the results, especially when we consider the overall precision and recall. We illustrate the first three steps in our algorithm on a TIMIT utterance in Figure 5-13. The blue curve is the extracted Hilbert envelope; the red circles consist of the current peak location set P. The sinusoid function, shown in magenta, is the fitted estimate of the current speech rhythm. The black dotted lines correspond to vowel boundaries in the phonetic transcription. From top to bottom, the figure shows the first three iterations of our algorithm. In the top plot, the first two peaks corresponding to the vowels /eh/ and /ax/ have been found without rhythm. From their locations, a rhythm sinusoid is estimated and extended into the future. In the middle plot, the next peak has been identified in the /ao/ vowel, which corresponds to the maximum peak in the interval under location (corresponding to 1.5 cycles of the rhythm sinusoid from the last peak). The rhythm sinusoid is re-estimated and extended into the future. In the bottom plot, the next peak has been located in the /ih/ vowel. This particular peak could have been easily missed due to its small size.

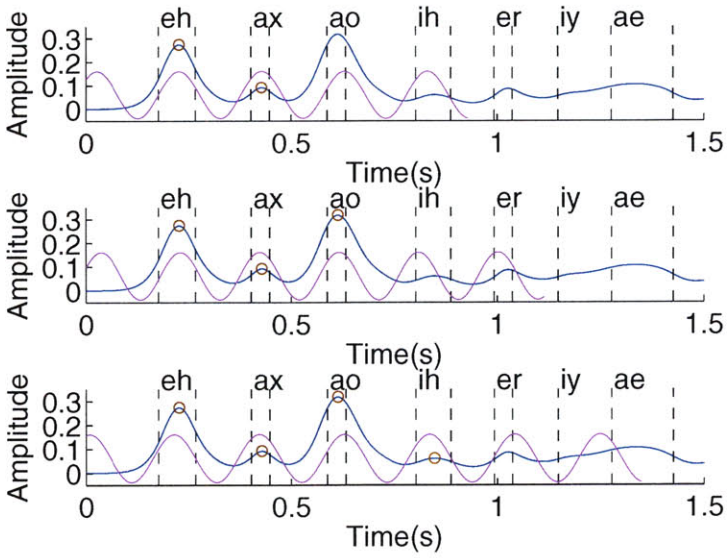

Figure 5-13: This figure illustrates an example of speech rhythm based detection of syllable nuclei. The blue curve is the extracted Hilbert envelope; the red circles consist of the current peak location set P. The sinusoid function, shown in magenta, is the fitted estimate of the current speech rhythm. The black dotted lines correspond to vowel boundaries in the phonetic transcription. From top to bottom, the figure shows the first three iterations of our algorithm. In the top plot, the first two peaks corresponding to the vowels /eh/ and /ax/ have been found without rhythm. From their locations, a rhythm sinusoid is estimated and extended into the future. In the middle plot, the next peak has been identified in the /ao/ vowel, which corresponds to the maximum peak in the interval under location (corresponding to 1.5 cycles of the rhythm sinusoid from the last peak). The rhythm sinusoid is re-estimated and extended into the future. In the bottom plot, the next peak has been located in the /ih/ vowel. This particular peak could have been easily missed due to its small size.

Pitch Verification

We observe that it is possible for our rhythm guided approach to find spurious peaks at the beginning or end of an utterance, or it can place some peaks in utterance internal pauses. Thus, pitch verification is used to remove spurious peaks in unvoiced regions. For both methods after all peaks are detected, a removal operation is performed if we find a peak being located in a highly likely unvoiced region.

5.3.4 Obstruent Detection

To have an accurate detection of obstruents, we also develop a multi-step algorithm. There are five steps in processing a speech utterance. First, a speech signal is decomposed into a band-pass filter with two channels. The lower channel ranges from 100Hz to 1500Hz, while the other, higher channel ranges from 4000Hz to 8000Hz. Then, the Hilbert transform is applied to the signal in each channel to calculate the magnitude $X_i(t)$ as

$$X_i = x_i(t) + i \cdot \mathcal{H}(x_i(t)) \tag{5.7}$$

where i ranges from 1 to 2, x_i is the signal in wave form, and $\mathcal{H}(\cdot)$ denotes the Hilbert transform. Therefore, the envelope E of each channel can be directly calculated by

$$E_i(t) = |X_i(t)| \tag{5.8}$$

The third step is to calculat the envelope ratio of the two channels. We denote channel one as the channel with low frequency range and channel two as the channel with high frequency range. The envelope ratio curve $R(t)$ can be calculated as

$$R(t) = \frac{E_2(t)}{E_1(t)} \tag{5.9}$$

It should be pointed out that, due to the different energy scales in each channel, each $E_i(t)$ should be normalized before calculating the envelope ratio.

The fourth step is to run the peak finding algorithm on the envelope ratio curve $R(t)$. As stated in the previous section, conventional peak finding algorithms tend to be very sensitive to the parameter settings, and there is no universal guideline for how to set parameters when facing a new dataset. In order to make all the components unsupervised, a fixed and aggressive setting of the peak finding algorithm is used but we add a fifth step to remove the noisy peaks.

In the fifth step, two additional sources of information are used to reinforce the agreement that the peaks found in the fourth step are all located in regions where

obstruents may appear. The first source of information is from our vowel center detector. After locating all the peaks in a rhythm guided way, we further extract the immediately adjacent valleys of each selected peak because it is possible for obstruents to appear in these regions. We collect the locations of vowel centers and their corresponding valleys. The other source of information is from the voicing detection (in the next section). Both the intervals that are voiced and intervals that are not voiced are collected. Then, for each peak in the fourth step, these two additional sources of information are used to score it. Specifically, each peak starts with score zero. If a peak is located around a valley(peak) on the vowel center curve, one credit adds to or subtracts from it. If a peak is located within an interval without(with) voicing, one credit adds to or subtracts from it. Finally, all peaks with a score above zero are selected as possible obstruents.

5.3.5 Voicing Detection

A pitch detection module is used to identify intervals that are voiced or not. This module provides necessary information for the obstruent detection. In addition, the rhythm guided vowel center detection approach also needs pitch verification to remove spurious peaks in unvoiced regions, such as a short pause and the beginning or the end of the utterance. In the current implementation, we use the ESPS pitch tracker [38].

5.3.6 Decision Tree Based Inferring

Given phonetic transcription, broad acoustic classes can be obtained by investigating the phonetic histograms generated from each Gaussian component cluster. Without any transcription, we may still infer the broad acoustic classes by looking at the acoustic feature histogram generated by each Gaussian component cluster. Specifically, by employing three acoustic feature detectors, each speech frame can be given a label indicating whether the frame has an acoustic feature or not. These acoustic feature labels can be viewed as phonetic transcriptions and can be used to produce acous-

tic feature histograms. The shape of a histogram provides important information to determine the broad acoustic classes.

The inferring algorithm is given as follows. For each Gaussian component cluster, we scan the entire input set of speech utterances and calculate the percentage of occurrences that have a specific acoustic feature. For example, if the Gaussian component cluster 1 appears 95 out of 100 times in a voiced region, we can infer that the phones that this component represents are likely voiced. As a result, this class will be given a feature label [+voicing]. If a speech frame appears within a 50 ms window of a vowel center or an obstruent, the frame will be given a feature label [+sonorant] or [+obstruent]. The set of feature labels are defined as [+/?/-voicing], [+/?/-sonorant] and [+/?/-obstruent]. A plus mark (+) means one frame has a particular feature, while a minus mark (-) means one frame not having that feature. A question mark (?) represents uncertainty, which means some occurrences have this feature while others do not. Each class will be assigned to these feature labels, and then the acoustic characteristics of each class can be inferred from the decision tree shown in Figure 5-14.

5.4 Experiments

In order to evaluate the broad acoustic class modeling framework, we designed two sets of experiments to 1) examine the performance of the three acoustic detectors; and 2) verify the broad acoustic class modeling framework. We use TIMIT as the experiment dataset, which includes a total of 6300 phonetically-transcribed read-speech utterances. In the first set of experiments, we mainly focused on examining the performance of the vowel center detection. In the second set of experiments, without any transcription, the broad acoustic class modeling framework was performed.

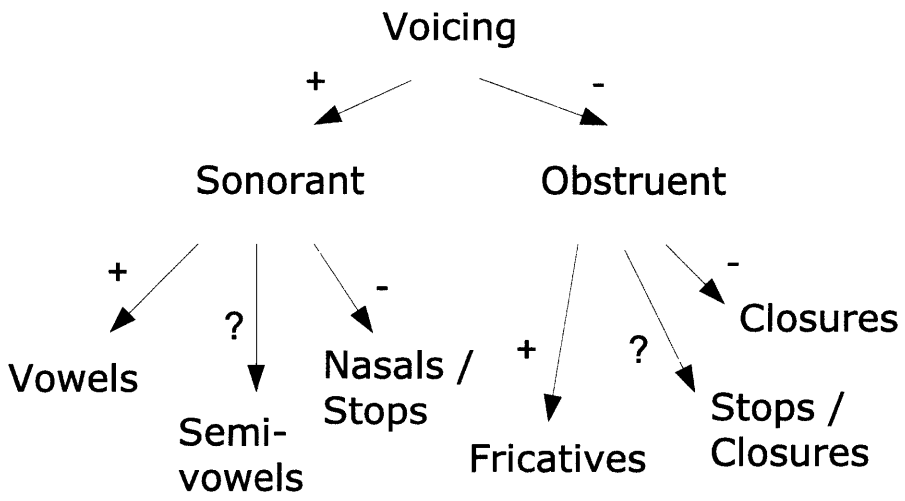

Figure 5-14: This figure illustrates the decision tree used in the broad acoustic class modeling framework. Plus mark means one speech frame having a particular feature, while minus mark means one speech frame not having that feature. Question mark represents uncertainty, which means some occurrences have this feature while others do not. The leaves of this tree are the broad acoustic classes output.

5.4.1 Vowel Center Detection Performance

Syllable-Nuclei Intervals

In order to establish the merit of the speech rhythm idea we examined the durations between nearby vowels in the TIMIT corpus, where vowels represented syllable nuclei. We gathered statistics on all syllable pair and triple sequences, measuring the interval between the first and the last vowel center of each sequence. As shown in the left plot of Figure 5-15, we are thus effectively measuring the syllable nuclei intervals (SNIs) of adjacent syllables (shown in blue), and of those separated by exactly one syllable (shown in red). Note that from any given vowel center, the expected interval to the next vowel center is approximately 200 ms, with an additional 200 ms to the following vowel center. The plot clearly shows tremendous overlap in the two distributions however, so that there is a range where either one or two syllabic nuclei could occur. This observation explains why previous approaches to syllabic nuclei detection often resorted to elaborate peak picking selection methods to decide where

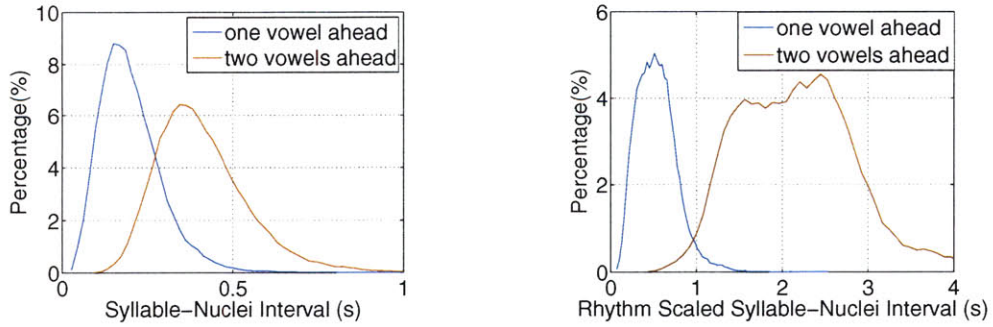

Figure 5-15: This figure illustrates the distribution of syllabel-nuclei intervals in the TIMIT and their corresponding rhythm-scaled versions. The blue curve is the syllable nuclei intervals (SNIs), while the blue curve is the syllable nuclei intervals separated by exactly one syllable. From any given vowel center, the expected interval to the next vowel center is approximately 200 ms, with an additional 200 ms to the following vowel center. The plot clearly shows tremendous overlap in the two distributions however, so that there is a range where either one or two syllabic nuclei could occur. This observation explains why previous approaches to syllabic nuclei detection often resorted to elaborate peak picking selection methods to decide where peaks could occur.

peaks could occur.

In an ideal case, if we knew the regular speech rhythm of syllabic nuclei, we would be able to better predict where the next syllable nucleus would occur. We can approximate this rhythm concept with our computed sinusoid frequency k_1 and use it to normalize the SNIs that were measured previously. Specifically, we scale each interval by an utterance-specific factor of $3\pi/k_1$, resulting in a dimensionless quantity that is plotted on the right side of Figure 5-15. This plot shows that the majority of SNIs for immediately adjacent syllables occur within an interval of $3\pi/k_1$ of an existing syllabic nucleus (blue). It also shows much less overlap with SNIs of following syllables (red). This result motivated our approach for syllabic nuclei detection, allowing us to avoid thresholds or parameters. The only parameter we selected was the value of $3\pi/k_i$.

The rhythm estimates used in Figure 5-15 were estimated over an entire utterance. A plot computed with rhythm estimates computed iteratively shows similar results. We have also found that the estimates for rhythm converge relatively quickly.

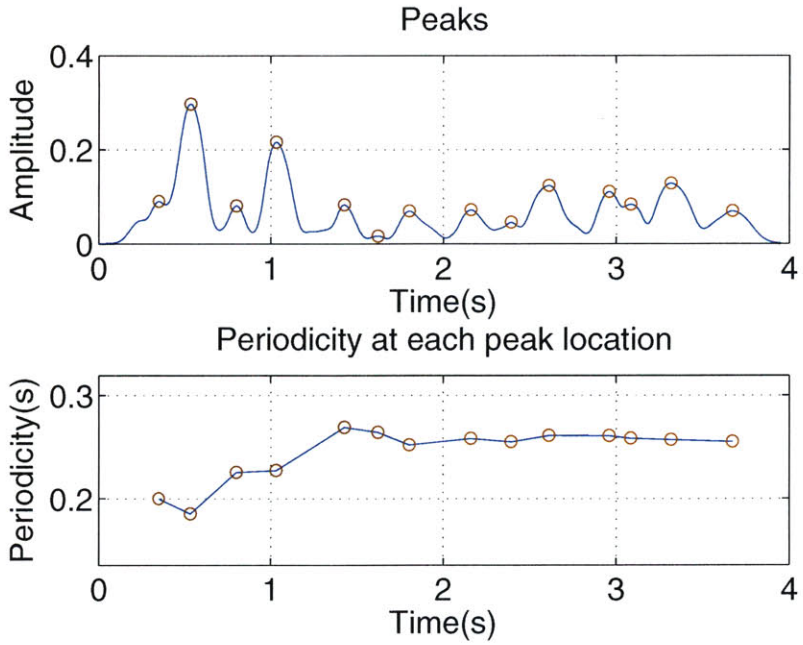

Figure 5-16: This figure illustrates how the rhythm sinusoid periodicity ($2\pi/k_i$) changes over time as it is computed in a left-to-right fashion over an TIMIT utterance. With each new peak detection (apart from the first two), shown in the upper plot of the figure, the sinusoid is re-estimated, and the period is plotted in the lower part of the figure. After several peaks have been located, the rhythm estimate becomes fairly stable, although the detection region still allows the detection of closely spaced syllable nuclei. Note that the default starting periodicity is 200 ms.

Figure 5-16 shows how the rhythm sinusoid periodicity ($2\pi/k_i$), changes over time as it is computed in a left-to-right fashion over an utterance. With each new peak detection (apart from the first two), shown in the upper plot of the figure, the sinusoid is re-estimated, and the period is plotted in the lower part of the figure. After several peaks have been located, the rhythm estimate becomes fairly stable, although the detection region still allows the detection of closely spaced syllable nuclei. Note that the default starting periodicity is 200 ms.

Performance Comparison

To demonstrate the performance of our rhythm-guided (RG) syllable nuclei detection approach, we compared it against the state-of-the-art syllable nuclei based speaking

rate estimation method TCSSC (temporal correlation and selected sub-band correlation) [41]. According to a recent comparative study paper of speaking rate [8], TCSSC has the best performance out of eight different systems. In addition, since the RG method and TCSSC have different signal processing modules, we built another system (nRG) that applies the conventional contrast based peak counting algorithm to the Hilbert envelope data without any rhythm guiding. Our intent was to quantify the degree to which rhythm information can help with syllable nuclei detection. We used the same ESPS pitch tracker [38] for both methods.

The results of our vowel detection experiments are reported in Table 5.1 in terms of the best overall recall, precision, and F-measure that could be achieved with each approach. A successful detection meant that there was exactly one syllabic peak detected within a 50 ms window of an actual vowel.

Since the TCSSC and nRG method used conventional peak picking, we tested a variety of different parameters to optimize performance. The only nRG parameter that was used was the $3\pi/k_1$ search interval, which was held fixed. All three methods had signal processing parameters that were varied to optimize performance.

The results indicate that in the best case scenario the three methods can locate between 80-87% of the vowels, and, in another best case scenario, can locate vowels with very high precision of 99%. The TCSSC and RG methods can achieve almost the same best recall performance, although the RG method requires significantly less parameter tuning.

Adding rhythm-based information clearly helps with recall and overall F-measure, although it seems to reduce best case precision over the nRG method. Overall, the best case F-measure showed the RG method outperformed both TCSSC and nRG methods. Given that TCSSC produced better recall results than nRG, it will be interesting to explore hybrid methods that combine elements of both the TCSSC and RG methods.

Table 5.1: The results of the TCSSC and nRG methods are obtained on the best parameter settings in terms of each criterion, while all results of the RG method is obtained by fixing the $3\pi/k_1$ search interval. The TCSSC and RG methods can achieve almost the same best recall performance, although the RG method requires significantly less parameter tuning.

	TCSSC	nRG	RG
Best Recall	0.8606	0.7997	**0.8659**
Best Precision	0.9969	**0.9984**	0.9886
Best F-measure	0.9021	0.8858	**0.9207**

5.4.2 Obstruent Detection Example

Figure 5-17 gives an example of the envelope ratio curve generated from a part of a TIMIT utterance. The top sub-figure contains three energy envelopes. The blue one is the envelope of the higher channel, while the red one is the envelope of the lower channel. The green one is the envelope from the rhythm guided vowel center detection. The red circles are the vowel centers found, and the plus marks denote the corresponding valleys of each vowel centers. The bottom sub-figure is the energy ratio envelope. Although three peaks are found on this envelope, by applying the removal process discussed in the obstruent detection section, we do not pick up the first peak because it is around a vowel center. Only the right two peaks (with square mark) are selected.

Since the obstruent detection is still in development, we leave the evaluation and comparison experiment for future work.

5.4.3 Broad Acoustic Class Modeling Result

We first give a brief overview of the parameters used in this experiment. The number of Gaussian components is set to 64. To cluster Gaussian components, the Affinity Propagation algorithm is performed 10 times to select the best clustering result using the minimum squared distance criterion. The ESPS [38] pitch tracker is used with all default parameter settings.

After clustering, 9 clusters are found. Table 5.2 gives the percentage of occurrence

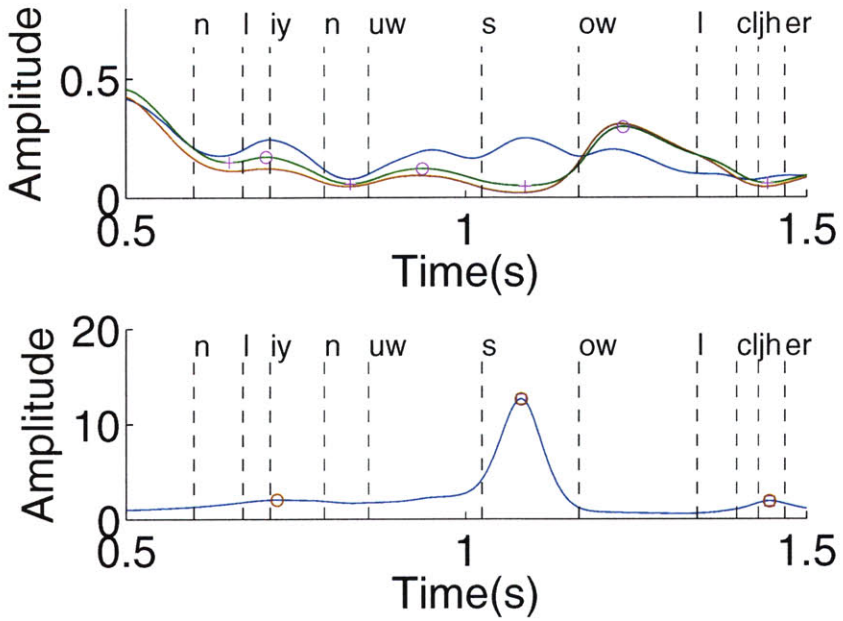

Figure 5-17: This figure illustrates an example of the envelope ratio curve generated from a part of a TIIMT utterance. The top sub-figure contains three energy envelopes. The blue one is the envelope of the higher channel, while the red one is the envelope of the lower channel. The green one is the envelope from the rhythm guided vowel center detection. The red circles are the vowel centers found, and the plus marks denote the corresponding valleys of each vowel centers. The bottom sub-figure is the energy ratio envelope. We can see that three peaks are found on this envelope. By applying our removal process discussed in fricative detection section, we do not pick up the first peak because it is around a vowel center. We select only the right two peaks (with square mark).

of each cluster having the voicing/sonorant/obstruent feature.

From the table, strong acoustic clues for each cluster can be observed. For example, com44 likely represents a broad acoustic class for vowels because most of its occurrences are voiced and sonorant. In order to calculate the feature label, "+" is set for a percentage larger than 70% and "-" is set for a percentage less than 10%. The mark "?" is given for all values in between. Therefore, by applying the decision rules, we can hypothesize the broad acoustic class that each cluster represents. Table 5.3 shows the result. Note that this part is currently the only supervised part in this framework. However, by setting appropriate thresholds for generating acoustic fea-

Table 5.2: After clustering, 9 clusters are found. This table gives the percentage of occurrence of each cluster having the voicing/sonorant/obstruent feature.

Cluster	Voicing	Sonorant	Obstruent
com44	98.57%	83.26%	0.53%
com17	97.61%	44.92%	0.10%
com30	94.24%	61.05%	4.71%
com16	90.94%	34.05%	0.25%
com12	90.73%	9.67%	0.60%
com18	60.93%	0.99%	1.73%
com14	6.05%	7.54%	33.62%
com19	3.53%	1.11%	42.89%
com40	0.51%	25.83%	75.97%

ture labels and applying the automatic rule based induction, this part can be easily converted into an unsupervised process.

To verify the correctness of our hypotheses, the corresponding phonetic transcriptions were used to generate a more detailed ground truth of the learned broad acoustic classes, shown in Table 5.4. The phonetic histograms are shown in Section 5.2.2 but from another random start. Note that all the phone labels are from the standard TIMIT 61 phone inventory. Compared to the data in Table 5.3, most hypotheses are correct except for minor sub-category differences. Although the com16 is incorrectly inferred because we do not include the retroflex class in our decision tree based inferring scheme, it does show a promising ability for capturing very short phones.

5.5 Summary

This chapter first presented a statistical method to investigate the learning ability of unsupervised GMM training. By examining the phonetic histograms generated from each Gaussian cluster, we observed that each Gaussian cluster is able to represent a set of phones (broad acoustic class) sharing similar acoustic characteristics. This observation gives the root cause of the viability of the proposed keyword spotting systems.

To model broad acoustic classes as well as hypothesize the acoustic meanings of

Table 5.3: From the statistics in Table 5.2, the feature labels "+", "-" and "?" are assigned to each cluster. "v" stands for voicing, "s" denotes sonorant and "o" represents obstruent. Based on the feature labels and the decision tree model, broad acoustic class for each cluster can be inferred.

Cluster	Feature Label	Acoustic Class
com1	[+v+s-o]	vowels
com2	[+v?s-o]	vowels/semi-vowels
com3	[+v?s-o]	vowels/semi-vowels
com4	[+v?s-o]	semi-vowels
com5	[+v-s-o]	nasals
com6	[?v-s-o]	closure/nasals
com7	[-v-s?o]	fricatives/closure
com8	[-v-s?o]	fricatives/closure
com9	[-v-s+o]	fricatives

each broad acoustic class, we also presented an unsupervised framework to automatically discover broad acoustic classes in unlabeled speech data. We developed a method to aggregate multiple sources of information to hypothesize the acoustic meanings of detected broad acoustic classes. We used the TIMIT dataset as the testbed and performed a completely unsupervised experiment to do broad acoustic class discovery. By comparing it with the underlying phonetic transcription, encouraging results are obtained.

Table 5.4: Ground truth of the broad acoustic classes each cluster represents. This result is based on the statistics on the TIMIT phonetic transcription.

Cluster	Broad Phone Class
com1	low vowels (aa ax au aw ow uw)
com2	semi-vowels(l w) / low vowels (ax au ow uw)
com3	high vowels (ae ay eh ih iy ix)
com4	retroflex (axr er r)
com5	nasals (n m ng)
com6	stop closures (bcl tcl kcl dcl) / nasals (n m ng)
com7	fricatives (f s z th) / stops closures (t)
com8	closures (bcl dcl gcl pcl tcl kcl epi pau h#)
com9	fricatives (s sh ch jh z)

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the main contributions of this thesis and discuss future work.

6.1 Summary and Contributions

The work discussed in this thesis is motivated by the problem of finding useful information in speech data in a completely unsupervised way. We first described the fact that in many speech processing problems, transcribed data is difficult to obtain. Then, we raised the problem of how much we can learn from the speech data without any transcription. In the main part of the thesis, we presented 1) two unsupervised spoken keyword spotting systems and 2) initial attempts of learning acoustically meaningful units.

In Chapter 3, a completely unsupervised keyword spotting system was proposed. Without any transcription information, a GMM model is trained to label speech frames with Gaussian posteriorgrams. Given one or more spoken examples of a keyword, a segmental dynamic time warping algorithm is used to compare the Gaussian posteriorgrams between keyword samples and the entire set of test utterances. The keyword detection result is then obtained by ranking the distortion scores of the test utterances. In the evaluation, we examined the TIMIT dataset as a development set to tune the parameters in our system, and the MIT Lecture corpus for a 30-

keyword set detection experiment. The results also demonstrated the viability and effectiveness of our approach.

In Chapter 4, we developed an unsupervised keyword spotting system when only text based samples are available. The system consists of five core components: the GMM learning, the clustering, the Joint-Multigram (JM) modeling, the JM n-best decoding and the symbol matching. The GMM learning part is the same as the system used in Chapter 3 but the most probable Gaussian component index labeling is used instead of Gaussian posteriorgrams. A clustering algorithm is then used to reduce the number of learned Gaussian components. The JM modeling is applied to build a mapping between a keyword and its corresponding Gaussian component indices. The decoding component converts a keyword to an n-best list of possible sequences of Gaussian component indices. The keyword spotting is done by symbol matching component through a region voting scheme. In the evaluation, we performed eight keyword spotting tasks on the TIMIT dataset. The results demonstrated the viability of our idea when a text-based keyword is desired.

In Chapter 5, to understand the underlying knowledge learned by unsupervised GMM, we investigated the phonetic histograms generated by each Gaussian cluster. By examining the results, we observed that unsupervised GMM training is able to group acoustically similar phones. Therefore, each Gaussian cluster can be used to represent a broad acoustic class. This observation gives the fundamental reason why the proposed keyword spotting systems can work in the unsupervised way. In addition, in order to develop a completely unsupervised broad acoustic class modeling framework, we explored three acoustic feature detectors to label speech frames. By combining the acoustic feature labels and GMM clustering labels, we developed a decision tree based broad acoustic class inferring framework. Without any transcription, not only can this framework output learned broad acoustic classes, but it can hypothesize the acoustic meanings of each class based on the acoustic feature histograms. The performance of the three acoustic feature detectors and the broad acoustic modeling framework were evaluated on the TIMIT dataset. The results are encouraging and motivates us to consider more unsupervised GMM learning applications in speech.

6.2 Future Work

Considering the problems addressed in this thesis, there is still much room for improvement for each of the three systems. We give detailed discussion in the following sections.

6.2.1 Unsupervised Keyword Spotting

Gaussian Posteriorgram Based Keyword Spotting

While this system represents our first attempt to use unsupervised methods to solve speech related problems such as spoken keyword spotting, there are still many issues waiting for more investigation. Specifically, in the unsupervised GMM learning, the current training method needs to manually set the number of Gaussian components. Based on the experiment we presented on TIMIT, it is clear that a good choice for the number of components can significantly improve performance. In the future we hope this number can be found in an unsupervised way such as using ideas from successive state splitting [31]. Since Gaussian posteriorgrams are probability vectors, it is also possible to use principle component analysis to reduce the vector dimensions. In the segmental DTW (SDTW) search, the current system requires a pre-set parameter R for adjustment window condition. By observation, changing this parameter highly affects the detection performance. More statistical analysis is needed to clearly understand the effect of this condition. Furthermore, since this framework is completely language independent and generic, we would like to examine its performance on languages other than English as well as other signal processing pattern matching tasks.

Multigram Based Keyword Spotting

Although the performance of the system only using text based keyword samples is not as good as the system using Gaussian posteriorgrams, it gives us many new thoughts on how to improve the post-processing techniques to compensate the information loss when using unsupervised learning. Specifically, the JM modeling is a multiplicative

process, which means the decoding result tends to prefer less segmentations. We find that sometimes it does not actually reflect the true segmentation of a word. Some smoothing techniques could potentially be used to solve this problem. In addition, the current symbol matching algorithm calculates the exact similarity based on label matching. A potential improvement is to use the distribution similarity between two broad phone labels instead of direct matching. As a result, a more accurate similarity score could be obtained.

6.2.2 Unsupervised Learning of Acoustic Units

In the broad acoustic class modeling framework, we have presented three acoustic feature detectors. The experiments on the TIMIT dataset shows the lack of knowledge in determining stops and nasals. Although we can infer these two kinds of phones by considering the combined information, it is better to have an indicator to further enhance our decision tree based inferring rules. Therefore, detectors for stops and nasals could be a part of our future work for improving the modeling resolution of broad acoustic classes. The obstruent detectors presented represents only our initial investigation into the problem of detecting obstruents. The signal processing part still needs further discussion and improvement.

Furthermore, more acoustic detectors can be used to model more detailed broad acoustic classes, such as determining the strong or weak fricatives, retroflex and approximant. It is also interesting to investigate the possibility of using unsupervised learning to automatically model and detect acoustic features. In our current frame based labeling method, we only consider the label itself without any context information. Some acoustic features could be represented by the transition between Gaussian component labels. For example, if the Gaussian component 5 represents voiced phones, component 3 represents silences and component 9 represents stops, a sequence of "5533399" can be used to determine a voiced stop by learning the possible prevocalic transition "55333." Therefore, if a sequence mining algorithm is applied to all speech utterances, some acoustically meaningful sequences can be learned, which could also help us determine the broad acoustic classes more accurately.

Bibliography

[1] http://www.nuance.com/recognizer/languages.asp.

[2] http://www.ldc.upenn.edu/.

[3] http://www.vistawide.com/languages/language_statistics.htm.

[4] A. Acero, X. D. Huang, and H. W. Hou. *Spoken language processing: a guided to theory, algorithm and system development.* Prentice Hall, 2001.

[5] G. Aradilla, H. Bourlard, and M. Magimai-Doss. Posterior features applied to speech recognition tasks with user-defined vocabulary. In *Proceedings of ICASSP'09*, pages 322–325, 2009.

[6] G. Aradilla, J. Vepa, and H. Bourlard. Using posterior-based features in template matching for speech recognition. In *Proceedings of INTERSPEECH'06*, pages 166–169, 2006.

[7] W. Chung-Hsien and C. Yeou-Jiunn. Multi-keyword spotting of telephone speech using a fuzzy search algorithm and keyword-driven two-level CBSM. *Speech Communication*, 33:197–212, 2001.

[8] T. Dekens, M. Demol, W. Verhelst, and P. Verhoeve. A comparative study of speech rate estimation techniques. In *Proceedings of INTERSPEECH'07*, pages 510–513, 2007.

[9] S. Deligne, F. Yvon, and F. Bimbot. Variable-length sequence matching for phonetic transcription using joint multigrams. In *Proceedings of EUROSPEECH'95*, pages 2243–2246, 1995.

[10] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[11] A. Garcia and H. Gish. Keyword spotting of arbitrary words using minimal speech resources. In *Proceedings of ICASSP'06*, volume 1, pages 337–340, 2006.

[12] J. R. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay. Recent progress in the mit spoken lecture processing project. In *Proceedings of Interspeech'07*, pages 2553–2556, 2007.

[13] J. R. Glass, T. Hazen, L. Hetherington, and C. Wang. Analysis and processing of lecture audio data: preliminary investigations. In *Proceedings of HLT-NAACL'04*, pages 9–12, 2004.

[14] T. Hazen, W. Shen, and C. White. Query-by-example spoken term detection using phonetic posteriorgram templates. submitted to ASRU'09, 2009.

[15] D. A. James and S. J. Young. A fast lattice-based approach to vocabulary independent wordspotting. In *Proceedings of ICASSP'94*, pages 377–380, 1994.

[16] A. Jansen and P. Niyogi. An experimental evaluation of keyword-filler Hidden Markov Models. Technical report, University of Chicago, 2009.

[17] O. Lartillot and P. Toiviainen. MIR in matlab: A toolbox for musical feature extraction from audio. In *Proceedings of ICMIR'07*, pages 22–27, 2007.

[18] C. Li and G. Biswas. Temporal pattern generation using Hidden Markov Model based unsupervised classification. In *Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis*, pages 245–256, 1999.

[19] P. Li, J. Liang, and B. Xu. A novel instance matching based unsupervised keyword spotting system. In *Proceedings of the Second International Conference on Innovative Computing, Informatio and Control*, pages 550–553, 2007.

[20] H. Lin, A. Stupakov, and J. Bilmes. Spoken keyword spotting via multi-lattice alignment. In *Proceedings of INTERSPEECH'08*, 2008.

[21] H. Lin, A. Stupakov, and J. Bilmes. Improving muliti-lattice alignment based spoken keyword spotting. In *Proceedings of ICASSP'09*, 2009.

[22] B. C. J. Moore. *An Introduction to the Psychology of Hearing.* Academic, 1997.

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

[24] A. Park and J. R. Glass. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 6(1):1558–1569, 2008.

[25] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of IEEE*, pages 267–296, 1989.

[26] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish. Continuous Hidden Markov Modeling for speaker-independent word spotting. In *Proceedings of ICASSP'89*, volume 1, pages 627–630, 1989.

[27] R. Rose and D. Paul. A Hidden Markov Model based keyword recognition system. In *Proceedings of ICASSP'90*, volume 2, pages 129–132, 1990.

[28] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[29] S. Siddiqi, G. Gordon, and A. Moore. Fast state discovery for HMM model selection and learning. In *Proceedings of the Eleventh International Conference on Artifici al Intelligence and Statistics (AI-STATS)*, 2007.

[30] S. D. Silvey and M. C. Ali. A general class of coefficients of divergence of one distribution from another. *Journal of Royal Statistical Society*, 28:131–142, 1966.

[31] H. Singer and M. Ostendorf. Maximum likelihood successive state splitting. In *Proceedings of ICASSP '96*, pages 601–604, 1996.

[32] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[33] I. Szoke, L. Burget P. Schwarz and, M. Fapso, M. Karafiat, J. Cernocky, and P. Matejka. Comparison of keyword spotting approaches for informal continuous speech. In *Proceedings of INTERSPEECH'05*, pages 633–636, 2005.

[34] I. Szoke, P. Schwarz, L. Burget, M. Karafiat, P. Matejka, and J. Cernocky. Phoneme based acoustics keyword spotting in informal continuous speech. *Lecture Notes in Computer Science*, 2005(3658):8–15, 2005.

[35] I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Fapso, M. Karafiat, and J. Cernocky. Comparison of keyword spotting approaches for informal continuous speech. In *2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, pages 12–15, 2005.

[36] J. Takami and S. Sagayama. A successive state splitting algorithm for efficient allophone modeling. In *Proceedings of ICASSP'92*, pages 573–576, 1992.

[37] Y. Takebayashi, H. Tsuboi, and H. Kanazawa. Keyword-spotting in noisy continuous speech using word pattern vector subabstraction and noise immunity learning. In *Proceedings of ICASSP'92*, volume 2, pages 85–88, 1992.

[38] D. Talkin. A robust algorithm for pitch tracking (RAPT). In *Proceedings of ICASSP'83*, pages 1352–1355, 1983.

[39] K. Thambiranam and S. Sridharan. Dynamic match phone-lattice searches for very fast and accurate unrestricted vocabulary keyword spotting. In *Proceedings of ICASSP'05*, pages 677–680, 1994.

[40] B. Varadarajan, S. Khudanpur, and E. Dupoux. Unsupervised learning of acoustic sub-word units. In *Proceedings of ACL'08*, pages 165–168, 2008.

[41] D. Wang and S.S. Narayanan. Robust speech rate estimation for spontaneous speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):2190–2201, 2007.

[42] M. Weintraub. Lvcsr log-likelihood ratio scoring for keyword spotting. In *Proceedings of ICASSP'95*, pages 129–132, 1995.

[43] L. D. Wilcox and M. A. Bush. Training and search algorithms for an interactive wordspoting system. In *Proceedings of ICASSP'92*, volume 2, pages 97–100, 1992.

[44] J. G. Wilpon, L. G. Miller, and P. Modi. Improvements and applications for key word recognition using Hidden Markov Modeling techniques. In *Proceedins of ICASSP'91*, volume 1, pages 309–312, 1991.

[45] J. G. Wilpon, L. R. Rabiner, C. H. Lee, and E. R. Goldman. Automatic recognition of keywords in unconstrained speech using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(11):1870–1878, 1990.

[46] Y. Zhang and J. R. Glass. Speech rhythm guided syllable nuclei detection. In *Proceedings of ICASSP'09*, pages 3797–3800, 2009.