Audio pattern discovery and retrieval

Wang, Lei

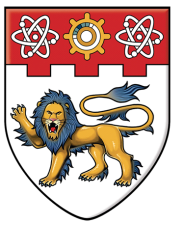
2012

Wang, L. (2012). Audio pattern discovery and retrieval. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/51781

https://doi.org/10.32657/10356/51781

Audio Pattern Discovery and Retrieval



NANYANG
TECHNOLOGICAL
UNIVERSITY

WANG LEI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2012

Abstract

This thesis explores unsupervised algorithms for pattern discovery and retrieval in audio and speech data. In this work, audio pattern is defined as repeating audio content such as repeating music segments or words/short phrases in speech recordings. The meanings of "pattern" will be defined separately for different types of data, for example, repeating pattern discovery in music will extract segments with similar melody in music piece; In human speech, the same words/short phrases spoken by single or multiple speakers are also defined as speech patterns; In broadcast audio, repeated commercials/logo music are also considered as patterns.

Previous work on audio pattern discovery focuses on either symbolizing the audio signal into token sequences followed by text-based search or using Brute-Force search techniques such as self-similarity matrix and Dynamic Time Warping. Symbolization process that relies on Vector Quantization or other modeling techniques may suffer from misclassification errors, and the exhaustive search requires high computation cost and can also be affected by channel distortion and speaker variation in audio data. Such limitations motivate me to explore more efficient and robust approaches to automatically detect repeating information in audio data.

In this thesis, different unsupervised techniques are examined to analyze music and speech separately. For music, an efficient approach which extends Ukkonon's suffix tree construction algorithm is proposed to detect repeating segments. For speech data, an iterative merging approach which is based on Acoustic Segment Model (ASM) is proposed to discover recurrent phrases/words in speech.

This thesis also explores the techniques of searching audio pattern in broadcast audio which consists of diverse content such as speech, music/songs, commercials, sound effects and background noise. Existing audio pattern retrieval techniques focus only on specific

audio types so that their applications are limited and cannot be applied generally. In this work, a robust query-by-example framework is proposed for retrieving mixed speech and music pattern, where the ASM is examined to model music data.

To verify the research, the proposed techniques are applied on both public domain audio database such as TIDIGITS corpus as well as TRECVID database and a self-collection of 30 English pop songs. The experimental results show that the proposed work achieves robust and better performance to existing techniques.

Acknowledgments

During the past years, it was my great honor to work with the groups of mentors and fellows in Nanyang Technological University (NTU) as well as Institute for Infocomm Research (I^2R), Singapore. Without their advice and encouragement this thesis would not be possible.

Foremost, I would like to express my gratitude to my supervisor, Dr. Chng Eng Siong (NTU), and co-supervisor, Dr. Li Haizhou (NTU, I^2R) for their excellent guidance and great support throughout my candidature. I would like to thank them for all their rewarding discussions, corporations, and kind suggestions to my study and life. Their extensive knowledge and sharp insights immensely influenced my research and their lasting encouragements help me to become a capable and confident researcher. Moreover, I would like to give special thanks to Prof. Lee Chin-Hui from Georgia Institute of Technology, USA for his valuable comments and suggestions on my research work.

I would like to thank the fellows in our team in NTU for their generous help and cooperation. I want to especially express my appreciation to Dr. Xiao Xiong, my senior, for his patience in teaching me the fundamentals of speech recognition as well as his encouragement in adapting myself to the research work. Moreover, I want also thank my teammates: Mr. Nguyen Trung Hieu, Mr. Omid Dehzangi, Dr. Zhao Shengkui, Dr. Lyu Dau-Cheng and Mr. Wu Zhizheng for their support and friendship.

I also would like to thank the colleagues in Speech and Dialogue Processing Lab of I^2R for their generous help. I want show my appreciation to Dr. Ma Bin for his discussion on my research work. I also want to thank other current and former colleagues, Ms. Tong Rong, Dr. Kinnuen Tomi Henrik, Dr. Sim Khe Chai for their help and corporation.

In addition, I thank the technical staffs in the Emerging Research Lab at NTU for providing me all the facilities, and thank the technical staffs in Parallel & Distributed Computing Centre (PDCC) for their kind help.

Last but not least, I want to thank my wife Jingying and my parents in China, for their constant love and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviation

ACORNS	Acquisition of Communication and Recognition Skills
AMG	Adaptive Motif Generation
ANN	Artificial Neural Network
ASM	Acoustic Segment Model
ASR	Automatic Speech Recognition
BER	Band Energy Ratio
BW	Bandwidth
DP	Dynamic Programming
DTW	Dynamic Time Wrapping
EM	Expectation Maximization
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HM-Net	Hidden Markov Network
HTK	Hidden Markov Model Toolkit
KWS	Keywords Spotting
LVCSR	Large-Vocabulary Continuous Speech Recognizer
MAP	Mean Average Precision
MFCC	Mel-Frequency Cepstral Coefficient
MI	Mutual Information
MIREX	Music Information Retrieval Evaluation eXchange
MVN	Mean and Variance Normalization
NIST	National Institute of Standards and Technology
OOV	Out-of-Vocabulary
PCA	Principal Component Analysis
PDS	Power Density Spectrum
RT	Rich Transcription
SAX	Symbolic Aggregate approXimation
SDR	Spoken Documents Retrieval
SFM	Spectral Flatness Measure
SNR	Signal-to-Noise Ratio
SRILM	SRI Language Modeling Toolkit
SSS	Successive State Splitting
STE	Short-Term Energy

TP	Transitional Probability
TREC	Text REtrieval Conference
TRECVID	TREC Video Retrieval Evaluation
VAD	Voice Activity Detection
VSM	Vector Space Model
VQ	Vector Quantization
ZCR	Zero Crossing Rate

Chapter 1

Introduction

The amount of audio data available on the Internet has increased exponentially in the past decades [1–4] - the availability of database such as broadcast news archives [5], radio recordings [3], music and songs collections [4], lecture and presentation recordings [6], meeting room recordings [7], podcasts [8], personal archives [9] have grown, and are now readily assessable to the public. However, most of these data have limited label information, or worse, have no label information due to the high cost of manual transcription. Hence, it is not easy for users to browse/preview the data or retrieve desired clips as compared to text documents. To overcome this limitation, much work has been carried out to explore methods that can automatically discover and structure audio content for browsing and retrieval purposes [10–13]. This has given rise to several established platforms for researchers to exchange and compare their ideas in this area. For example, the Music Information Retrieval Evaluation eXchange (MIREX) [14], Star Challenge 2008 multimedia search evaluation campaign [15] and NIST Rich Transcription (RT) evaluation series [7] are well known competitions for content-based audio analysis. These research activities inspire this thesis to investigate audio pattern discovery and retrieval technologies.

1.1 Research Objectives and Motivations

In this work, the topic "audio pattern discovery and retrieval" refers to the following research problems: 1) to automatically discover repeating patterns in audio data; 2)

to retrieve similar audio segments given a query example. Specifically, exploration of unsupervised algorithms for the research problems is a key focus.

The first objective of this thesis is to examine unsupervised techniques to automatically discover repeating patterns in music and speech corpus. A music piece consists of different segments such as chorus, verses, bridge, vocal runs, starting as well as ending [4], and listeners usually tune to each music piece through the repetitions of chorus and verses. Hence, the identification of repeating music segments can provide a convenient way for listeners to browse music collections to locate favorite pieces. To detect repeating segments, the music signal is usually analyzed at the feature level, i.e. a feature extraction module is first used to extract a sequence of time-stamped multivariate feature vectors [16] from the raw music data. As repeating segments of music have roughly the same length, existing symbolic repeating subsequences detection techniques such as suffix tree construction algorithm can be extended and applied. The research problem focused in this thesis is the efficient repeating subsequences discovery for multivariate time series data. Here, the efficiency factors mean the computational cost and memory requirements in running the algorithm.

The techniques used for pattern discovery in music however are not directly suitable for speech data due to different nature of these signals' characteristics. For example, Wolfe [17] mentioned that: 1) The formants of phonemes in speech may vary from one speaker to another while the formants due to most musical instruments do not vary significantly; 2) The spectrum of speech is dominated by a relatively small number of narrow bands while music spectrum usually has large number of bands; 3) The pitch of speech generally varies continuously, whereas the pitch of an individual musical note is usually relatively stable, i.e. music has better short-term stability of the frequency components. These differences in the characteristics of the two signals suggest that speech pattern discovery requires a more robust modeling approach to address the larger variations of speech patterns.

Automatic speech pattern discovery has attracted continued interest from both computer science [18] and psychology [19] respectively. According to the psychological study [19], the infant English learners, at the age of 10.5months, show their sensitivity to statistical regularities and other cues when they identify word boundaries from

fluent speakers. The computer science researchers inspired by these findings also explored techniques to discover speech patterns in an unsupervised manner. For example, previous research has studied the discovery of fundamental acoustic units [18] by automatically modeling the sub-word units from untranscribed speech corpus. In this thesis, sub-words units are further examined to discover meaningful speech patterns such as words and phrases.

The second objective of this thesis is to examine audio pattern retrieval techniques on broadcast audio data. Different from repeating pattern discovery, audio pattern retrieval is to search for audio information given a specific content. As discussed in [20], various techniques are proposed to retrieve desired archives from audio database, e.g. query-by-example [21–25], query-by-text [26–28] and query using semantic labels such as sound effects [10, 29, 30]. Among these audio retrieval methods, the task of query-by-example is to locate a given audio segment, i.e. a query example, in a given audio database. This thesis also focuses on exploring audio retrieval techniques for the query-by-example applications of broadcast audio data.

The retrieval of audio pattern from broadcast audio data remains a difficult problem [31] due to the diversity of audio content. Broadcast audio consists of various audio content types, e.g. speech, music/songs, commercials, speech with music background, sound effects and environmental noise. It is not easy to automatically convert broadcast audio into transcription for indexing as techniques of analyzing audio content such as music [4] and sound effects [10, 29, 30] are different from speech. Many techniques in the literature focus only on specific audio types so that their applications are limited to the selected audio types and cannot be applied to general broadcast audio. In this thesis, the research focus is to examine a retrieval technique which can process multiple audio content types such as news, music and commercials simultaneously.

1.2 Contribution of this Thesis

To achieve the above 2 objectives, this thesis examines unsupervised techniques to discover audio patterns in music and speech, and search for audio patterns in mixed speech and music audio database. The contributions of this thesis are as follows:

- To detect repeating segments in music piece, an efficient online technique based on [32] is proposed to construct a suffix tree to accept vectors as input. This technique achieves linear time operation requirement to construct a sparse self-similarity matrix [33] for pattern discovery. Experimental results on an English pop song collection show that the proposed technique has similar F1-measure performance with a significant computation reduction as compared to the conventional self-similarity approach[1].

- To discover recurrent speech patterns in untranscribed speech corpus, a self-organizing technique is proposed to iteratively merge sub-word units towards discovering long speech patterns. The process begins by first transcribing speech signal into sub-word sequence and then identifying two sub-word units with high co-occurrence. A new model is created by merging the two selected units. The iterative process continues till a predefined stop criterion is met. Experimental results on TIDIGITS corpus show that the proposed technique successfully discovers 10 out of 11 words with high F1 score[2].

- To locate queries in mixed speech and music audio broadcast database, a combined model is proposed to simultaneously model speech and music segments without the need to first discriminate between them. The combined model is built from two separate models: one is the conventional phoneme recognizer trained from speech data, and the other model is trained from music collection to represent music

[1]This work has been published in the following papers:

(i) **Lei Wang**, Eng Siong Chng and Haizhou Li, "A tree-construction search approach for multivariate time series motifs discovery," *Pattern Recognition Letters*, vol. 31, no. 9, pp. 869-875, 1 July 2010.

(ii) **Lei Wang**, Eng Siong Chng and Haizhou Li, "Efficient repeating segments discovery in music using adaptive motif generation algorithm," in *Proc. APSIPA ASC 2009*, Sapporo, Japan, October 4 - 7, 2009.

(iii) **Lei Wang**, Eng Siong Chng and Haizhou Li, "Efficient sparse self-similarity matrix construction for repeating sequence detection," in *Proc. ICME 2009*, New York City, USA, June 28 - July 3, 2009.

[2]This work has been published in the following paper:

(i) **Lei Wang**, Eng Siong Chng and Haizhou Li, "An iterative approach to model merging for speech pattern discovery," in *Proc. APSIPA ASC 2011*, Xi'an, China, October 18 - 21, 2011.

sounds. The broadcast audio archives are decoded using this combined model and indexed by n-gram statistics of the decoded output. Experimental results show that the proposed technique outperforms baseline system and is robust and effective to transcribe broadcast audio data containing both music and speech on a query-by-example task of searching for 100 audio queries in a 28 hour audio corpus extracted from TREC Video Retrieval Evaluation (TRECVID) [34] database[3].

1.3 Organization of this Thesis

This thesis is organized as follows:

Chapter 2 reviews current applications and techniques for audio pattern discovery and retrieval - published work of music structure analysis, speech pattern discovery, audio indexing and retrieval, query-by-example and clustering are reviewed. Chapter 2 also surveys existing techniques to perform unsupervised repeating pattern discovery.

Chapter 3 examines an efficient technique to detect repeating segments in music.

Chapter 4 proposes a self-organizing approach to iteratively merge sub-word units to discover long speech patterns.

Chapter 5 proposes a method to index mixed speech and music audio database for audio pattern retrieval.

Chapter 6 concludes the thesis and suggests possible future research goals.

[3]This work has been published in the following paper:

(i) **Lei Wang**, Haizhou Li and Eng Siong Chng, "A vector-based approach to broadcast audio database indexing and retrieval," in *Proc. ICME 2007*, Beijing, China, July 2-5, 2007, pp. 512-515.

Chapter 2

Literature Review

Audio pattern discovery and retrieval is a broad research topic that has generated much research literature and many applications in the past two decades. The past work targeted different data and application areas such as music segment discovery, speech pattern discovery and content-based audio retrieval. The literatures [35–37] provide broad review for each application research area.

This chapter first introduces different applications of audio pattern discovery and then reviews recent literature on techniques, implementation, and solutions for the audio pattern discovery.

This chapter is organized as follows: Section 2.1 describes applications developed towards audio pattern discovery; Section 2.2 examines commonly used audio features followed by existing solutions including unsupervised techniques to discover repeating patterns from symbolic sequence and multivariate time series.

2.1 Applications of Audio Pattern Discovery and Retrieval

Many large audio databases such as music recordings [4], podcasts [8], broadcast news [5], dialogues [38] and conversations [39], meetings/conferences [40], lectures [6, 41] and presentations [42], etc are now readily available to users on the Internet. However, most of these data have limited or no available tag information and thus interested users cannot easily search for desired audio segments. Hence, the ability to automatically analyze such data has received much interest.

This section reviews applications and techniques to automatically analyze music, speech and broadcast TV recordings by detecting repeating patterns. Specifically, the applications include music structure analysis and summarization, speech summarization, audio indexing and retrieval, and other multimedia applications such as sports program analysis.

2.1.1 Music structure analysis and summarization

Music/songs data are among the most popular audio content. Much of these data have been produced commercially with additional information online such as title, genre, performer, lyrics and history [35] to consumers. Although the additional information allows interested users to search for desired songs based on these criteria, its use is limited when user seeks to query songs based on the music itself, e.g. query by humming [43, 44], or query by similar melody characteristic [45]. For such queries, musically salient features of the desired music pieces should be directly used to query the music databases. Such requirement has motivated the research for music structure analysis.

The music structure analysis research is a field of audio analysis that seeks to locate structures such as chorus, verses, bridge, vocal runs, in a music piece [4]. Human listeners usually recognize music structure through repetitions, i.e. segments which occur frequently such as chorus and verses. Therefore, an important component of music structure analysis research is to first detect repetitions [46].

Figure 2.1: SmartMusicKIOSK system for music structure analysis [47].

A successful application which implements the above idea is the SmartMusicKIOSK [47] system. The SmartMusicKIOSK was developed to locate chorus and repeating verses of a piece of song. It enables music shop customers to search out their desired parts of a song efficiently.

2.1.2 Speech pattern discovery and summarization

Applications to automatically analyze and summarize broadcast TV and radio programs are highly sought as such data are of major interest to the public. A common approach is to first generate a basic transcription of the spoken audio using a large-vocabulary continuous speech recognizer (LVCSR) system and then uses the transcribed text for indexing, or to further apply text summarization techniques to generate summarized information [2, 5, 11, 38, 48, 49].

Speech summarization techniques have been applied to many different types of speech data sources such as spoken news [5, 11, 49, 50], spontaneous presentations [42], dialogues [38], conversations [39], lectures [41], meetings [51] and voicemail messages [52]. For example, one representative system of spoken news summarization is the Rough'n'Ready system developed by BBN Technologies [5]. The system incorporated a wide range of speech technologies to index speech data, translate languages, generate rich transcription information such as speaker identification, highlight of keywords and proper nouns, create a structural summarization and provide tools for browsing. Figure 2.2 shows the BBN Broadcast Monitoring System.

Another example of speech summarization technology is the research undertaken by the Japanese national project "Spontaneous Speech: Corpus and Processing Technology" in 1999 [53]. They [11] proposed a probabilistic method to extract relatively important words from transcribed speech and use these words to generate summarized text sentences. In their recent work [2], a two-stage summarization framework consisting of important sentence extraction and word-based sentence compaction was used to produce text summaries. Their approach also directly produced audio summaries by identifying important sentences, words and between-filler units from original utterances. Zechner [38] also used utterance extraction techniques for an automatic summarization system called DIASUMM to summarize open-domain spoken dialogues. The system used

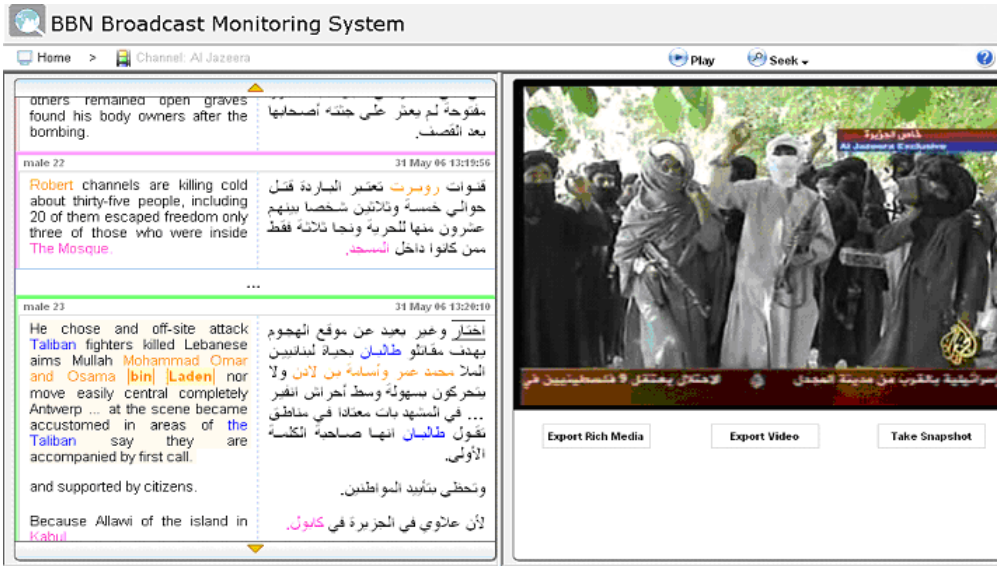


Figure 2.2: BBN broadcast monitoring system. The application can show transcription in both the original language and English with the synchronized video. (From BBN website)

dialogue transcription as input and identify important utterances by detecting and removing speech disfluencies, determining sentence boundaries and linking cross-speaker information units.

Besides sentence extraction techniques, other methods based on transcription also have been explored. Valenza *et al.* [48] combined acoustic confidence measures [54] and inverse frequency values to evaluate individual words in the news transcription so that higher ranked words could be included in the summaries. Jin and Hauptmann [49] introduced several supervised strategies to generate titles for broadcast news with the aid of pre-selected news titles in the training corpus.

In the above mentioned work, the LVCSR was applied to generate basic transcription. However, the LVCSR recognition performance on broadcast data is poor. This leads to much work to improve LVCSR performance by incorporating prosodic features into transcription [55–57]. The prosodic features such as intonational variation in pitch, energy, pause and speaking rates are found to signal the relative importance of speech segments [58]. An early application of prosody on speech summarization was for voicemail messages [52] where the researchers explored the use of prosodic and lexical features to discriminate the important segments/words of the transcription. In the spoken news

summarization applications, prosodic features such as pitch, power and pause were employed to locate stressed words [55]. For talk shows [57], prosodic features such as pitch, phoneme duration, sentence length and power were used together with linguistic information to evaluate the importance of each sentence in the transcription.

Unlike conventional speech summarization, recent speech pattern discovery research does not rely on speech recognizers or transcription. Instead, a discovering process to automatically generate lexicon-like information from unknown languages is examined. A typical example of such application is the Acquisition of Communication and Recognition Skills (ACORNS) project launched at Europe in 2006 [59]. The ACORNS project developed and implemented mathematical model which is capable of acquiring human verbal communication behavior. Similar applications include [13] and [60] which generate keywords from single-speaker lecture recordings by detecting the recurrent speech patterns for topic detection.

2.1.3 Audio indexing and retrieval

As the majority of multimedia data available on the Internet are unstructured, it is not easy to locate desired segments. This motivates researchers to propose various technologies to index large audio database, e.g. spoken documents retrieval (SDR) [23, 26–28], keywords spotting (KWS) [61], and query-by-example [20, 62] systems.

Most of the SDR systems aim to retrieve the most relevant spoken documents given a text query from a large speech corpus [27], e.g. the TREC Spoken Document Retrieval Track [26] evaluation task focuses on audio retrieval of broadcast news clips. To retrieve spoken documents, a common approach many systems [23, 27, 28] adapted is to first transcribe audio followed by the construction of an index using the transcription. This process is illustrated in Figure 2.3.

Such systems however face two major issues: i) out-of-vocabulary queries and ii) high word error rate. To address these problems, the vocabulary-independent approach to speech indexing has been proposed. In this way, speech utterances are first transcribed into subwords [63] or acoustic lattice [24, 25] and then index and search.

Another application on speech indexing and retrieval is KWS. KWS systems are used to detect selected words in speech utterances [61] and is similar to SDR as both systems aim to locate queries in speech corpus.

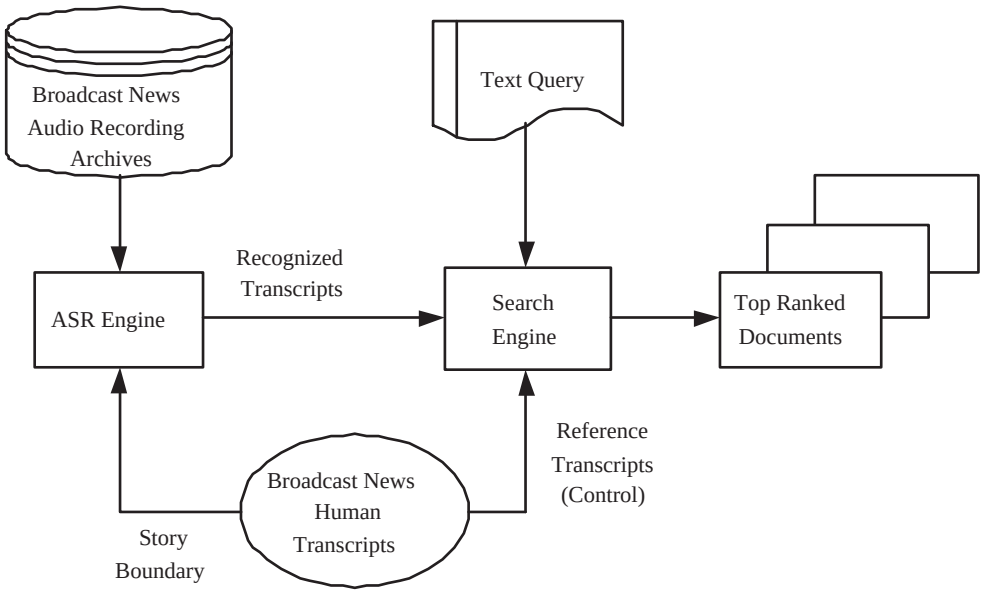


Figure 2.3: SDR system recommended by TREC Spoken Document Retrieval Track 1997.

To locate similar audio queries for generic audio segments, query-by-example applications have been developed. For example, in [22], the researchers proposed an algorithm to first extract signatures from the audio signal and searched for queries by matching the desired signature. However, the performance of their system degrades when there is distortion in the audio signals. In another approach, Velivelli *et al.* [20] modeled the predefined audio query using hidden Markov model (HMM) and use the models as queries. However, this approach is poor for short query segment as HMM estimation is not robust. In [64], classification methods are applied to retrieve similar segments belonging to the same semantic class.

2.1.4 Other multimedia applications

Broadcast sports program is also an important class of multimedia data. Much research has been done on sport event detection using audio and video information. This section discusses existing work which exploits the use of audio information for sports event detection.

Most sports video consumers are usually more interested in the highlights of a sports video than the entire recording of a game, e.g., for a soccer game, the highlights are the

goal scoring scenes, foul scenes, penalty shots, etc. Hence, many automatic highlights extraction techniques [65–69] have been examined and proposed. To detect these scenes, audio content are sometimes exploited. For example, in [65], the announcers' excited speech and ball-bat impact sound were used to detect possible highlight candidates. In [66], audio signals were classified into semantic classes such as applause, cheering, music, speech and speech with music and used as features for subsequent classifiers to identify highlight segments. In another approach, the ball hits have been detected to indicate the highlights in table tennis games [67]. Moreover, high energy segments were found to closely follow the occurrences of goals in basketball games [68]. To analyze soccer game, Wang [69] classified audio features such as Mel-Frequency Ceptral Coefficient (MFCC) and Linear Prediction Cepstral Coefficient (LPCC) into three groups: "whistle", "acclaim" and "noise". Such auditory labels are useful for high-level semantic analysis.

In contrast to sports program, scenes from movies and sitcoms cannot be easily classified as many portions of the program may be equally important and the event boundaries may not be obvious [70]. The research article [71] presents a review of the existing works on such data.

The audio classification approach has also been applied to segment and analyze movies and sitcoms [71–76]. In [71] and [73], audio features such as short-term energy and pitch were extracted to measure a movie's tempo so that the movie could be segmented according to the change in tempo. To assign segments into semantic classes, HMM was used to identify emotional events such as laughter and horror [72, 74]. In other research, different audio features including short-time energy [75],band energy ratio, zero-crossing rate, frequency centroid, bandwidth and MFCCs were combined to classify gunfires, explosions, car braking and other audio effects in action movies [76].

This section has reviewed a broad range of applications which involve speech pattern discovery, indexing and retrieval, and also various audio mining technologies for sports/movie videos. In the next section, the existing techniques will be examined in details.

2.2 Unsupervised Techniques for Repeating Pattern Discovery

Pattern discovery is a popular research topic in many fields such as genomic, image processing, and natural language processing. Effective unsupervised techniques have been proposed for different applications. As this thesis deals with the problem of discovering repeating patterns from audio signal, only the techniques for such data type will be concentrated.

The general process to discover repeating patterns from audio data is summarized in Figure 2.4. The process first extracts features characterizing the audio signals and then performs pattern discovery. To discover patterns, previous published work can be categorized into 2 major approaches: feature symbolization by statistical modeling followed by symbolic subsequence detection, and brute-force search. In this section, the pattern discovery techniques for symbolic sequence are reviewed in Section 2.2.2 and the statistical modeling and brute-force search techniques are reviewed together for multivariate vector sequence in Section 2.2.3.

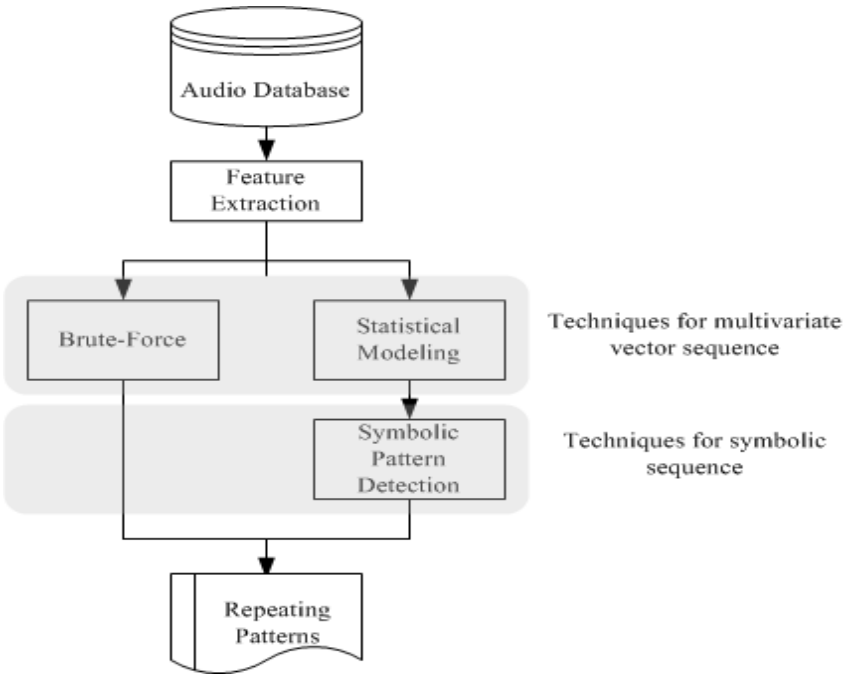


Figure 2.4: Current approaches for audio pattern discovery.

Before examining unsupervised techniques in details, several common audio characterization methods are introduced.

2.2.1 Audio characterization

Many different audio features have been proposed to capture the most relevant and discriminative characteristics of audio signal for modeling and recognition. Existing literatures [77] classify audio features into two main categories: (a) physical features extracted directly from the audio waves, and (b) perceptual features. Some of the commonly used features are described below [77, 78]:

(a) Physical Features

Short-term Energy (STE): STE is also referred to as volume or loudness, and it measures the total spectral power of an audio signal. Although it is widely used in Voice Activity Detection (VAD) applications, it is not robust for many audio recordings.

Band Energy Ratio (BER): Band energy models the distribution of spectral power in different sub-bands of frequency domain. BER measures the energy ratio of a high frequency band to a low one. A large BER indicates that the audio segment contains mostly high frequencies.

Zero Crossing Rate (ZCR): ZCR measures the total number of times the waveform crosses the zero axis over a unit length of time. It is useful to discriminate speech from non-speech components such as music.

Brightness: It is computed as the centroid of the short-time Fourier magnitude spectra measured in log frequency. It is sometimes referred to as frequency centroid.

Bandwidth (BW): BW represents the power-weighed standard deviation of the spectrum in a frame. Brightness and bandwidth are usually combined to describe statistical characteristics of the spectrum in a frame.

Pitch: It is defined as the sensation of frequencies of a sound. It also refers to the fundamental frequency (F0) of an audio segment. F0 is a physical term of audio

waveform, however, pitch is a perceptual term and it is the perceived F0. Normally voiced speech and music have well-defined pitch.

(b) Perceptual Features

Linear Prediction Cepstral Coefficient(LPCC): LPCC [79] is the cepstral coefficients derived from the linear prediction model. It is widely used for speech recognition task.

Mel-Frequency Ceptral Coefficient(MFCC): MFCC [80, 81] is the most popular features for LVCSR. The feature captures the short time segment statistics based on models of human auditory perception with respect to different frequency range.

Chroma Feature: Chroma feature [16] is a powerful representation of music audio. Chroma features captures the salient features of music by projecting the spectrum into 12 bins of semitones (or chroma) of the musical octave.

Enhanced Features: To obtain more robust features, cepstral features can be further processed to reduce noise, multi-speakers' effect and channel distortion. For instances, to reduce channel and background noise effects, segmental histogram equalization has been applied on MFCCs for robust speech recognition [82]. To discriminate different speakers, discrete cosine transform has been applied on MFCCs to generate longer term temporal features for speaker verification task [83].

The above list shows various audio features commonly used in current research. Audio pattern discovery techniques can then be formulated as a problem to discover repeating audio patterns in the sequence of audio feature vectors.

In the following sections, different repeating pattern discovery techniques of various fields ranging from genomic to signal processing, conventional clustering techniques as well as statistical modeling techniques are reviewed. To discover patterns, the audio feature vectors can be either converted to a symbolic sequence or processed directly. The following subsections discuss the approaches dealing with these two types of data in details.

2.2.2 Techniques for symbolic sequence

From the late 1990s, research on symbolic sequence has resulted in the development of numerous methods to discover repeating patterns or motifs. Repeating pattern discovery of symbolic sequence has been used in many different applications such as motif finding in genomic sequences [84, 85] and themes extraction in music data represented by MIDI symbolic sequence [86]. A broad review of methods to discover repeating patterns can be found in [85] and [87]. Among these methods, 4 common approaches are discussed: suffix tree [32, 88–90], dot-matrices [86, 91], incremental search [92, 93] and local alignment [94, 95].

One approach to find repeating pattern is to represent a token sequence as a suffix tree structure so that the occurrences of each subsequence can be retrieved from each non-leave nodes [88]. Linear time algorithms such as [32] have been proposed to construct suffix tree efficiently. To improve the efficiency to detect repeating substring patterns in suffix tree, He [89] proposed to first identify all possible candidates for repeats and then prune to reduce false alarms. However, the use of suffix tree only works effectively for exact match sequence. For symbolic sequence interspersed by unwanted characters, an inexact-suffix tree structure which allows motifs discovery has been studied in [90].

Another approach to discover repeating pattern in symbolic sequence is to construct dot-matrices [91]. The dot-matrices were initially proposed to represent pairwise alignments of genomic sequences. Each element in the dot matrix indicates if the corresponding two symbols are identical, as illustrated in Figure 2.5. The dot-matrices have been further extended in [86] to produce correlative matrix which can be used to find nontrivial repeating patterns from music MIDI sequence. However, such approach operates in a brute-force manner and is not efficient for long symbolic sequence.

To detect repeating pattern for long symbolic sequence, the incremental search approach has been proposed [92, 93]. Such algorithms first generate short candidate strings and use them as candidates. The candidates that rarely occur are removed, and the remaining candidates are appended with new symbols to form new candidates strings for the next iteration [92]. To avoid costly repetitions during search, an efficient algorithm to examine fewer intermediate repeating patterns was proposed in [93]. This algorithm scans the long sequence once to generates short potential repeating substrings followed

G	0	0	0	0	0	0	0
A	**1**	0	0	0	0	0	0
E	0	**1**	0	0	0	0	0
F	0	0	**1**	0	**1**	0	0
P	0	0	0	0	0	**1**	0
M	0	0	0	0	0	0	**1**
I	0	0	0	0	0	0	0
	A	E	F	K	F	P	M

Figure 2.5: A dot matrix for pairwise symbolic sequences "GAEFPML" and "AEFKFPM". The element "1" represents the corresponding symbols from the two sequences are the same, and "0" represents they are different.

by connecting them into a directed graph. Maximum-length repeating pattern search is carried out by examining the edges of the graph.

Dynamic programming (DP) has been popularly applied to discover motifs and search queries in genomic database [96]. One successful application is the BLAST genome search tool [97] that uses one of the DP algorithms - local alignment. This technique is able to identify similar sub-sequences between two symbolic sequences by constructing a two dimensional distance matrix in which each element represents the pairwise distance between the symbols of the two sequences. DP, together with a pre-defined gap penalty, is then used to detect stripes along diagonals. Due to its success in genomic area, local alignment techniques have also been used for speech applications [94, 95]. Nowell and Moore [94] modified local alignment to discover similar portions from multiple phoneme sequences in topic spotting application. Instead of applying it on symbolic sequences, Aimetti [95] applied local alignment on 'soft' distance matrix calculated from floating number input such as cosine distance between two front-end feature vectors in speech data to detect repeating words in two sentences.

2.2.3 Techniques for multivariate vector sequence

Researchers have also applied techniques to discover patterns directly on multivariate vector sequence without the symbolization process. Examples of multivariate sequence include time series such as sequence of feature vectors extracted from speech segment [98],

auditory and visual feature sequence of video data [99], fetal ECG data [100, 101], and on-body sensor data [101, 102]. In addition, the numerical time series [103] is also included here as it can be considered as a one dimensional vector sequence. As the distance measurements among vectors are real values, it is not feasible to detect exact matching repeating patterns. To overcome the above shortcomings, researchers have proposed many novel approaches for pattern discovery research, as discussed in the following sections.

The previous work on pattern discovery in multivariate vector sequence can be broadly categorized into two approaches: i) first converting the multivariate vector sequence into symbolic sequence using clustering or statistical modeling techniques followed by applying symbolic pattern discovery techniques; ii) Direct search techniques using self-similarity matrix, Dynamic Time Wrapping (DTW) and HMM.

2.2.3.1 Clustering techniques

One of the common approach to convert multivariate vector sequence into symbolic sequence is data clustering or vector quantization (VQ) [79, 81]. Specially, unsupervised clustering is performed and each target vector is represented by its class identity. Existing clustering methods have been reviewed in many literatures [104, 105] and different taxonometric presentations of the methods are available. In this thesis, the taxonomy of clustering methods follows [105] and is shown in Figure 2.6.

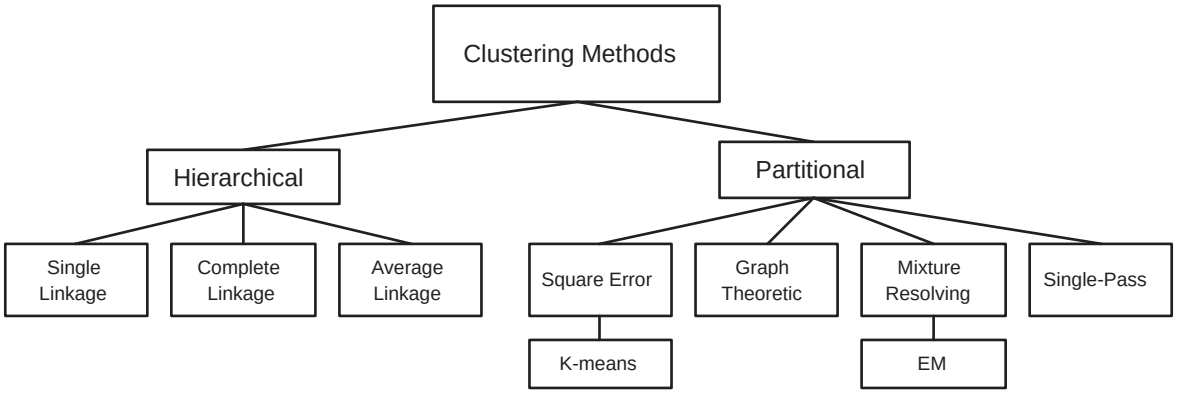


Figure 2.6: A taxonomy of clustering methods [105].

The clustering methods can be divided into two broad categories at the top level: hierarchical and partitional methods. The hierarchical agglomerative clustering algorithm

initially treats each pattern vector as an individual cluster and then merge similar pair of clusters agglomeratively. Eventually, a hierarchical structure is obtained to show the relationship among all the pattern vectors. To measure the distance between two clusters, linkage is used to select single points that can represent the clusters. Three linkage types are commonly used: 1) Single linkage defines the distance between two clusters as the minimum distance of all possible pairs of pattern vectors in the two clusters. 2) Complete linkage is opposite to the previous one and it uses the maximum distance. 3) Average linkage calculates the mean distance.

In another category, partitional algorithms aim to divide the data into various groups instead of constructing a hierarchical structure. There are several strategies to partition data [105]. First, squared error criterion is the most frequently used because of its simplicity in implementation. One typical example using square error is k-means clustering which iteratively reassigns the data to clusters based on the distance between data and clusters until convergence. Due to its simplicity, k-means has been applied in many research areas. In audio pattern discovery, research work such as [106–109] also utilizes k-means to convert frame sequence into symbols. For instance, in [106], speech data is first divided into short segments followed by applying segment clustering on their feature vectors to label the short segments. DP is performed on the label sequences to capture word patterns. Secondly, graph theoretic clustering presents the relationship among all the data so that the partitions will be clearly archived [110]. The well-known graph-theoretic divisive clustering algorithm obtains the partitions by removing the long edges from the minimal spanning tree constructed for data. Thirdly, mixture resolving methods assume the data are drawn from different distributions so that their goal is to determine the parameters of the models, e.g. the Expectation Maximization (EM) algorithm [104] has been applied to parameter estimation. Lastly, single-pass methods [111] have been introduced to reduce computation effort for large size of data and can partition data in linear time. The single-pass algorithm is based on a first-come-first-serve principle and assigns data into the currently nearest cluster. Such method can only produce an approximative partition.

Besides the above mentioned clustering methods, other approaches have also been proposed to symbolize the vector-based sequence. In [112], Keogh *et al.* proposed Symbolic Aggregate approXimation (SAX) [103, 113] to transform the univariate time series

data into low-complexity symbolic representation so that symbolic sequence search techniques can be applied. The advantage of SAX is its robustness to measurement of noise. To apply SAX for multivariate time series, the multivariate data is first projected into one dimension using principle component analysis (PCA). Motifs are then found using minimum description length [101]. As this approach only uses the first component of PCA, it is limited to analyze data that can be correctly represented by its first principal component [102].

Symbolization introduces quantization errors hence is not robust when actual data is different from training data such as speech utterances. To overcome this shortcoming, modeling techniques such as HMM can be used to increase the robustness of the frameworks.

2.2.3.2 HMM related techniques

HMM-based techniques that operate on the multivariate vector sequence without the symbolization process have been proposed in [79, 99, 114–117]. In their work, the HMM [79, 114] is used to characterize the spectral properties of a given audio signal. An HMM consists of several elements: i) A number of states; ii) The output observations with a probability distribution from each state; iii) Transition probabilities at each state; and iv) Initial state distribution. In speech recognition applications, researchers build HMM in a restricted topology, i.e. the state transitions can only repeat or transit from left to right. State sequence associated with such model has the property that as time increases the state index increases or stays on the same state [114]. Each state is typically modeled by a Gaussian Mixture Model (GMM). Figure 2.7 shows an example of HMM topology: a 3-state left-to-right HMM is built for each phoneme for a language. In some applications, more states may be used to create HMMs for modeling words.

The parameters of HMMs are usually estimated in a supervised manner using the Baum-Welch algorithm described in [114]. The detailed implementation can be also found in [118]. Besides the supervised methods, unsupervised variations of HMM have also been investigated [18, 99, 115–117, 119–124]. This section will also examine unsupervised methods to train HMM.

From late 1980s to early 1990s, researchers realized that conventional HMM approach faced problems in dealing with spontaneous speech due to acoustic variability. One of

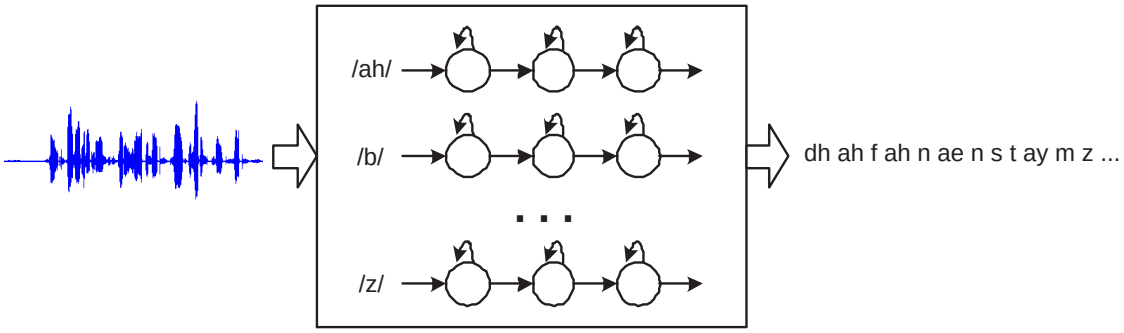


Figure 2.7: Structure of an HMM network in a typical phone-level English speech recognition system. Each English phone is modeled by a left-to-right HMM.

their arguments was whether the phonemes should be considered as the smallest units in speech. Many researchers put effort to explore other units such as sub-words, syllables and data-driven units from speech data. Such studies can be considered as pioneering work of speech pattern discovery, and Ostendorf summarized these previous work as "beads-on-a-string" model [125]. The following paragraphs review some of the studies in details.

The researchers in [119] examined method to build Markov models for isolated words by concatenating predefined fenone models. A fenone is known as a sub-phone unit in speech and it represents a single frame in their work. In their approach, a small amount of transcribed speech data is used to train 200 prototypes of fenones. To build word models, each frame of input frame is assigned a pre-defined fenonic label based on nearest neighbor criteria. The fenonic sequence is then used as a reference to construct the word model by concatenating the fenone Markov models. The recognition is carried out using DP. Obviously, such method can only be applied to single speaker corpus due to the limitation on predefined prototypes.

Unlike modeling fenones in isolated words speech, Takami and Sagayama [120] proposed Successive State Splitting (SSS) to study an optimal representation for a set of acoustic units in continuous speech automatically. By applying SSS, a hidden Markov network (HM-Net), which is a single HMM with a number of trained states with optimal parameters in an optimal topology, can be generated. In each iteration of SSS, the state with the largest divergence in HM-Net will be split into two states in either temporal or contextual domain, depending on the reduction in likelihood of all samples. Each state

sequence of HM-Net can be interpreted as a phone or sub-word unit. As an extension of SSS, Singer and Ostendorf [121] considered maximum likelihood criterion for selection of the state and the way to be split so that the HM-Net can grow towards to global optimal. Recently, Varadarajan *et al.* [122] improve the efficiency of SSS by simultaneously splitting all the existing states into 4 followed by merging back states with less occupancy. They also further proposed to use finite state machine as a transducer to automatically assign meaningful label to each state sequence hence SSS could be applied to continuous speech recognition. However, the HM-Net generated by SSS is complicated and the interpretation of state sequence may vary to different speakers.

In another attempt, a data-driven approach, namely Acoustic Segment Model (ASM), was proposed to characterize fundamental speech sounds for speech recognition in [18]. The ASM models have the similar topology as conventional HMMs of speech recognizer, however, its training process does not require transcription of speech data. Instead, ASM approach iteratively decodes untranscribed training data and re-estimate the ASM models using the latest decoding output by maximum likelihood criterion. Hence, the iterative training process can update ASM models towards the best representation of the data and each resulting ASM model represents a sub-word unit. These sub-word units have been further explored to create lexicon in [123]. The idea of ASM was also subsequently extended by Li *et al.* [124] to train universal phoneme models for the language identification task.

Unsupervised HMM techniques have also been examined to discover patterns in music data. For example, Ergodic HMM [115] illustrated in Figure 2.8 has been used to group and label similar fixed-length segments using MFCC features for the music structure analysis task. The most frequently occurred label is identified as "frequent label", and then heuristics are applied to select the fixed length (10sec) segments with the most number of "frequent label" as key phrases. Aucouturier and Sandler [116] further explored ergodic HMM using spectral envelope features. In their work, each HMM state is assumed to represent a musical part so that a music piece can be segmented. To improve long term modeling, Abdallah *et al.* [117] used longer duration cepstral feature vectors to train the ergodic HMM. In their implementation, the decoded Markov state sequence is modeled using histograms and the histograms are clustered to find repeating patterns.

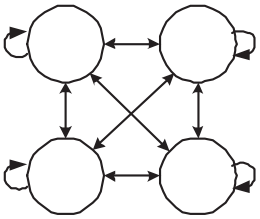


Figure 2.8: Structure of a 4-state ergodic HMM used in [115]. Each state is modeled by a Gaussian distribution.

In yet another variation of HMM, the hierarchical HMM was explored to discover recurrent patterns in video achieves [99], and the researchers proposed unsupervised adaptation algorithm to automatically model newly occurred events using both auditory and visual features.

HMM-based techniques create statistical models to describe acoustic phonemes or sub-word units in human language. It is however hard to build HMMs for longer audio content such as phrases in speech and music segments. Thus, direct search approach has been examined to solve the problem. In the following sections, direct search techniques such as self-similarity matrix, dynamic programming and other brute-force techniques are discussed.

2.2.3.3 Self-similarity matrix

Besides the above mentioned clustering and statistical modeling approaches to discover patterns in multivariate vector sequence, the self-similarity matrix [12, 33, 47, 126, 127] has also been widely studied for repetition detection in audio segments, especially for music structure analysis. Although such brute-force technique is computationally expensive, it is feasible to analyze music piece as such data lasts only for several minutes.

A self-similarity matrix, illustrated in Figure 2.9, is constructed based on a distance measure such as correlation between the feature vectors of an audio segment. This idea was first examined by Foote [12] in which he constructed a frame-to-frame similarity matrix to visualize similarities between segments of music. The repeating musical patterns are then found using image processing techniques by locating diagonal lines with high similarity values in the similarity matrix. Foote [128] also described a method to detect the music segment boundaries by analyzing local similarity of adjacent musical

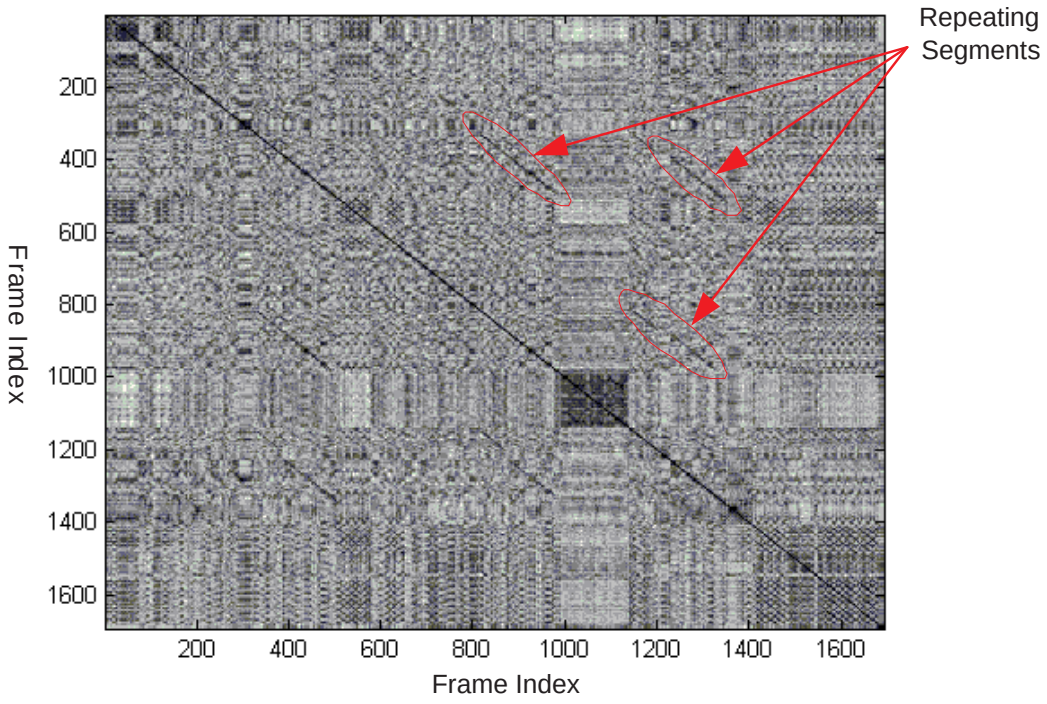


Figure 2.9: A self-similarity matrix generated for the pop song "When You Say Nothing at All" by Ronan Keating using the temporal chroma features. The stripes in the image represent the occurrences of repeating segments.

signals. Foote's work was based on MFCC features which although is robust for speech recognition applications, is not suitable to analyze music data [129]. Other researcher working on similarity matrix to detect repeating patterns have proposed more robust music features. Bartsch and Wakefield [16, 126] introduced the chroma features which can represent the cyclic attribute of pitch perception for music notes. Their experiments have shown that chroma features were able to capture better recurrent structures such as chorus and verse within a given song. Their work was extended by Goto [47] to detect and differentiate the chorus section from other repeating patterns in music audio signals. Other extensions of chroma feature include Zhang and Samadani's work [127] which used similarity measure based on blocks of chroma features to detect longer repeated melody. In another approach, Lu *et al.* [33] applied constant Q transform on music piece to extract features and used image processing techniques prior to repeating pattern detection process to enhance the stripes in the features self-similarity matrix.

2.2.3.4 Variants of dynamic programming

Dynamic Time Warping (DTW) technique is another popular approach to detect patterns and it has also been widely applied for speech recognition applications [130] during the past decades. DTW is a realization of DP and can be used to detect unknown speech sequence to known template words. To detect templates, DTW finds an optimal alignment between two sequences, as illustrated in Figure 2.10. Earlier research work [80, 131] has shown the effectiveness of DTW for isolated words recognition.

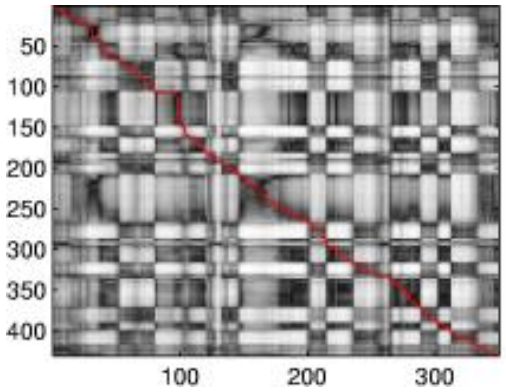


Figure 2.10: A distance matrix generated using classic DTW algorithm. The illustration compares the Short-Time Fourier Transform feature vectors of two words. The matrix contains the pairwise distances between vectors from each sequence. The curve line (dark line) along the diagonal shows the optimal alignment. [132]

Recently, Park and Glass [13] proposed the segmental DTW, a variant of conventional DTW, to detect repeating speech patterns between spoken utterances pairs. The method divides the entire search space into sub-space for patterns detection using DP hence the computational cost is reduced as the conventional DTW applies DP to the entire search space, as illustrated in Figure 2.11. All the detected patterns are then clustered into groups based on distance so that frequently occurred patterns can be captured. The segmental DTW has been successfully applied on single-speaker lecture recordings in which the same word spoken by the same speaker has only small variations. Researchers in [60] extended segmental DTW to obtain extractive summary by evaluating the sentences that contain required keywords. Other work such as [133] also use a variant of DTW to detect keywords in a call-center application. As DTW-based methods are applied directly on the feature vectors, they may not be robust in noisy and multi-speaker environment.

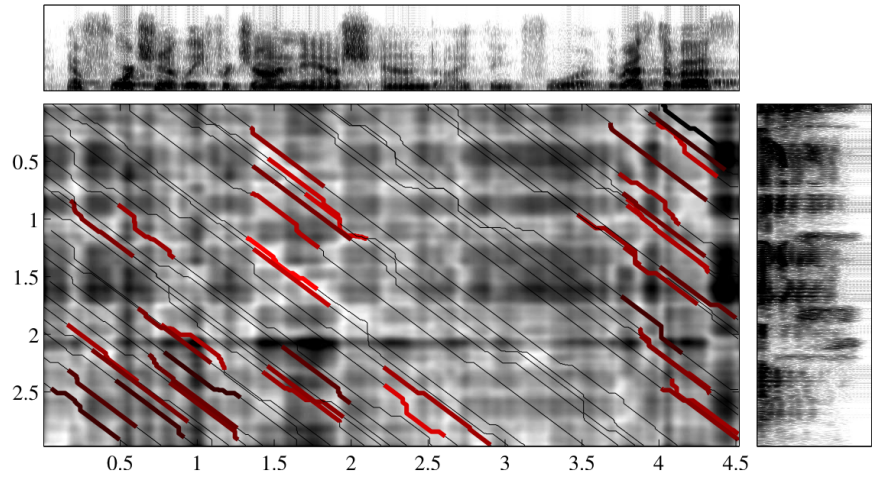


Figure 2.11: An illustration of segmental DTW algorithm. The distance matrix is cut into diagonal regions and DP is implemented in each region to find optimal alignment. Each alignment is trimmed to have the least average distance with a minimum length (the thick lines). [13]

In another approach, Oates [134] proposed PERUSE algorithm to detect frequently occurring patterns from a set of multivariate sensory data series. PERUSE adopts a sliding window to step over the entire database and uses DP to identify the best occurrence of the candidate motifs. PERUSE allows the repeating patterns found to be of variable lengths but it assumes that motifs occur frequently. The implementation for the dynamic programming process is exhaustive and is computationally expensive.

In yet another exhaustive search approach, researchers in [62] and [3] proposed to manually construct templates for different audio events and match the audio corpus to these templates in a brute-force manner. To avoid the manual identification of templates, an automatic method was implemented in [135] to identify keywords in audio signal by scanning the audio and locate repeating blocks using DTW.

This chapter summarized a broad range of work for audio pattern discovery and retrieval. Inspired by the previous work of repetition detection, a novel technique for music repeating segments detection is examined in Chapter 3, and then chapter 4 examines an unsupervised mechanism to discover speech patterns.

Chapter 3

Automatic Repeating Segments Discovery in Music

This chapter focuses on the research problem of the automatic discovery of repeating segments in music. The repeating segments, also known as "patterns" or "motifs", refer to the music segments with similar melody but possibly different lyrics. Such repeating segments are usually the salient portions of a music piece and can be used to recognize music structure. In addition, users can easily locate their favorite pieces from a collection of pop songs by listening to frequently occurring segments such as verses and chorus [46, 47].

It is difficult to detect repeating segments in music since segment length and number of times of repetitions are unknown. Existing techniques such as self-similarity approach [33] are able to solve the problem by detecting the stripes in pairwise distance matrix using image processing techniques. However, it requires high computational cost to process each element in the matrix. This motivates this work to explore more efficient method to discover repeating segments in music.

In this chapter, an efficient technique is proposed to discover repeating music segments in unsupervised manner. The technique extends the online suffix tree construction algorithm and accepts feature vector sequence as input hence to generate a list of candidate repeating segments. A refinement process is then applied to finalize the repeating segments. Experimental results on a collection of 30 English pop songs and two sensory data sets show that the computational cost is reduced greatly without sacrificing detection accuracy, comparing to the existing work [33]. The following sections are organized

as follows: a brief review of the previous techniques is provided in Section 3.1. The proposed technique is described in Section 3.2, and the refinement process is introduced in Section 3.3. Section 3.4 discusses the experimental results, and the conclusion is given in Section 3.5.

3.1 Introduction

Research on repeating segments discovery in music has attracted much attention in the last decade. For example, to analyze music represented by MIDI sequence, an RP-Tree [86] and a recursively scanning method [93] are studied to discover patterns. In another work, an N-gram inverted index structure is examined to mine music data for content based information retrieval in [136]. More symbolic sequence mining techniques such as suffix tree [32] are also developed to solve the general symbolic sequence mining problems, as discussed in the review paper [87].

Symbolic sequence techniques however cannot be directly applied to analyze composite music such as pop songs and symphony if their music scores are not available. This motivate researchers to study alternative representations of composite music. One such feature is the chroma feature which can capture the salient features of music by projecting the spectrum into 12 bins of semitones (or chroma) on the octaves [16]. Researchers explore various symbolization methods to convert feature vector sequence into symbols so that existing symbolic techniques can be applied. Unfortunately, symbolization techniques suffer from quantization errors and hence may not be robust for all applications [137].

Chapter 2 has discussed the use of self-similarity matrix [12, 33, 47, 126, 127] to detect repeating segments in music. The overall framework is shown in Figure 3.1. The music piece is first divided into short overlapping frames to generate feature vectors, and the pairwise similarity measure among these vectors are evaluated to construct an $N \times N$ vector-to-vector similarity matrix to visualize the similarities among the N features of a music segment. The repeating musical patterns are then found using image processing techniques by extracting the sub-diagonal lines with high scores in the similarity matrix. Figure 3.2 provides an illustration of the self-similarity matrix processed by each module.

Unfortunately, this approach requires a complexity of $O(N^2)$ for memory and computational requirement and hence limits the application of the approach to short sequences.

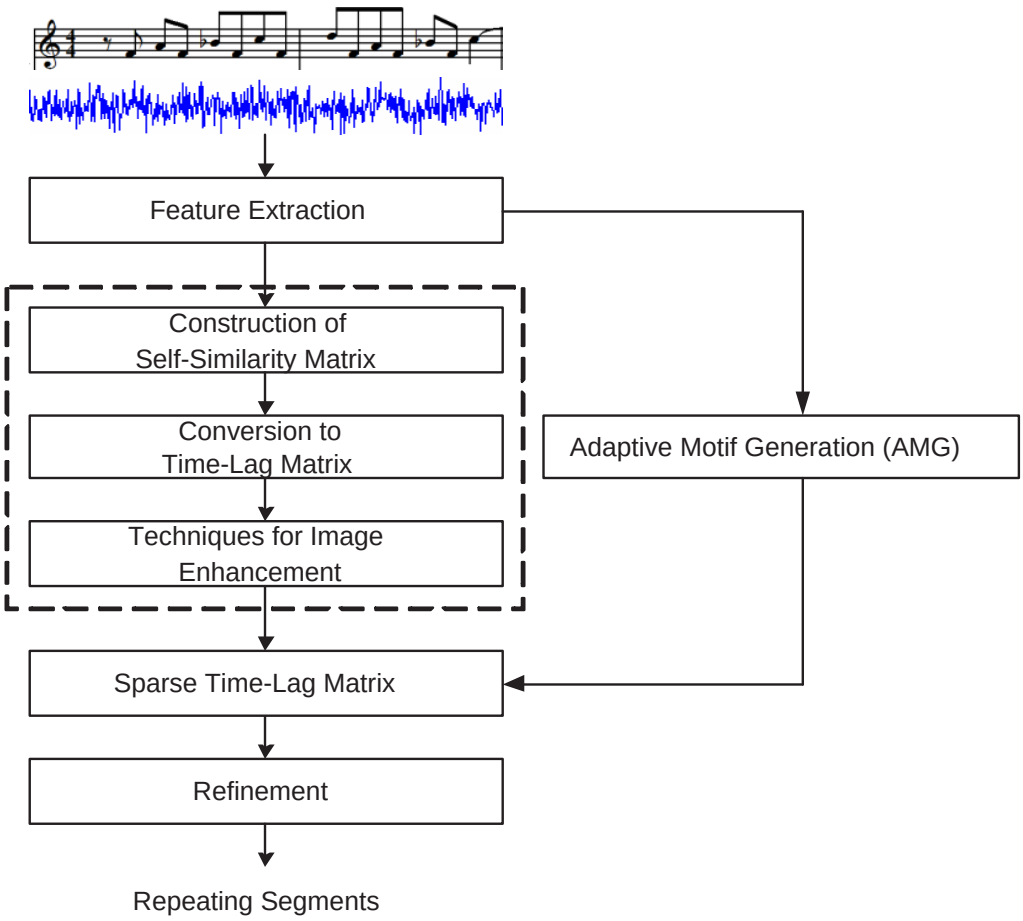


Figure 3.1: The framework of self-similarity matrix approach. The proposed Adaptive Motif Generation algorithm will replace the computational consuming modules.

Motivated by such limitation, this chapter is to examine an efficient and robust approach to detect motifs for multivariate vector sequence. The efficiency factors this work is concern with are the computational complexity and memory requirements. In this work, an efficient approach, namely Adaptive Motif Generation (AMG), is proposed to detect motifs, i.e. repeating segments, in multivariate sequences such as music feature vector sequences. The AMG first scans the input data to construct a list of candidate motifs in linear time. An adaptive threshold is used to make the construction process more

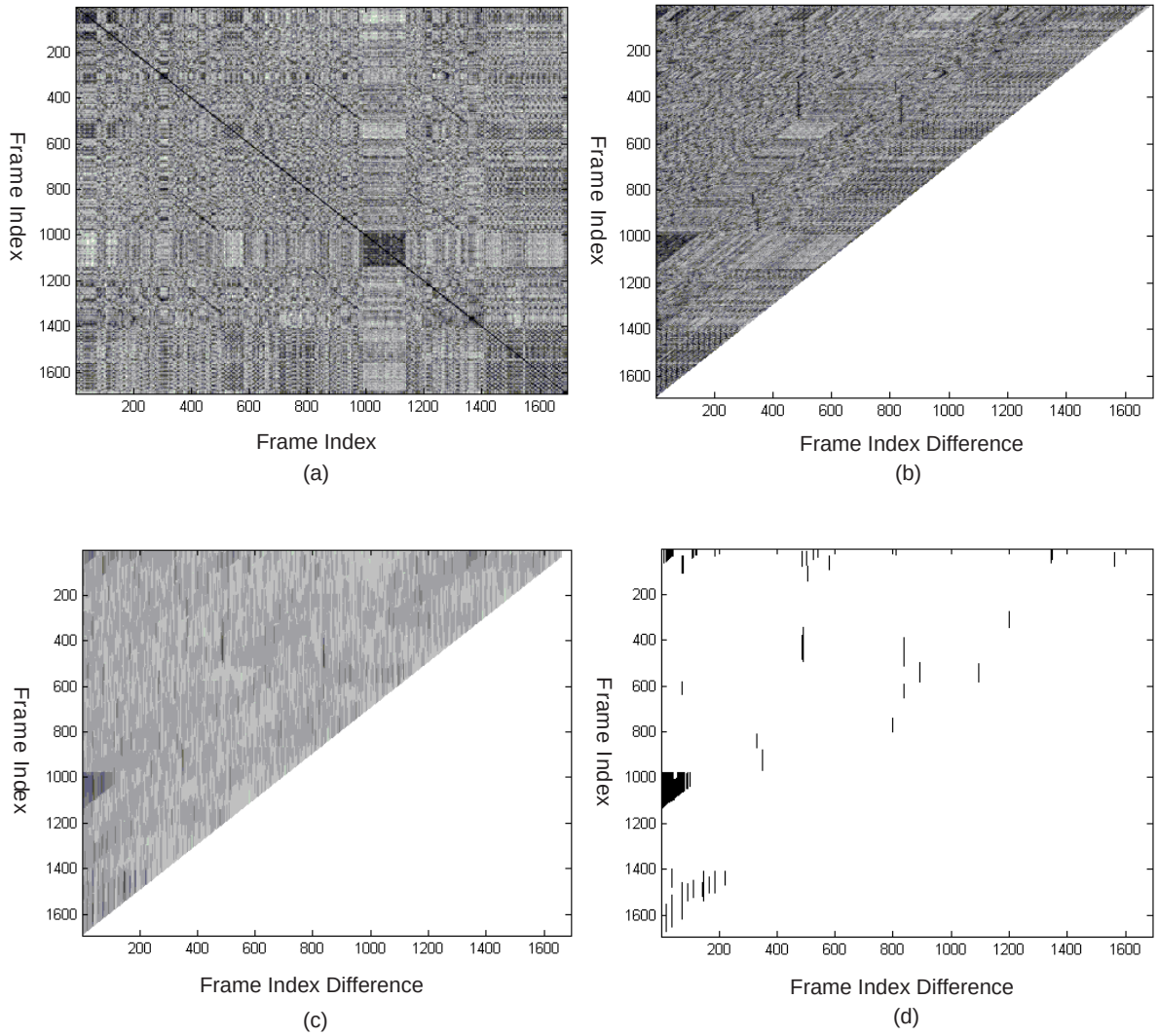


Figure 3.2: Illustrations of repeating music segments discovery using a similarity matrix. (a) The similarity matrix; (b) The corresponding time-lag matrix; (c) Time-lag matrix processed using image enhancement techniques; (d) Sparse time-lag matrix obtained by binarising the elements in (c). Refinement process will be applied to (d) to generate the final repeating segments.

robust to retain poor intermittent segments. The candidates are then post-processed to identify longer repeating patterns using techniques similar to the self-similarity matrix approach. Figure 3.1 shows the AMG block within the self-similarity framework. As the number of candidate motifs found in the initial process is very small, the candidate motif matrix constructed is very sparse ($< 1\%$ occupancy). Hence, the proposed approach can be applied to very large database.

The following section describes the proposed AMG algorithm in details.

3.2 Candidate Motif Generation

In this work, the process to discover repeating segments from multivariate vector sequence consists of 2 stages: stage 1 generates candidate motifs and stage 2 refines the candidates to detect the desired patterns. The candidate motif generation procedure is first described in this section, followed by the refinement process in the next section.

To find candidate motifs, i.e. repeating subsequences, from multivariate vector sequence, this work focuses on two objectives. The first objective is to improve the efficiency over the previous techniques. To realize efficiency, an online technique is examined to detect candidate motifs in a single pass approach. The second objective is to detect candidate motifs that have long duration. To discover long motifs, robust techniques which can deal with poor intermittent subsequences are required.

To achieve the above objectives, the proposed AMG approach constructs a modified suffix tree to locate the repeating subsequences. Its strategy is to extend Ukkonen's online suffix tree construction algorithm [32] to analyze the input vector sequence without the symbolization process. The Ukkonen's online suffix tree construction algorithm is popular as the algorithm can build a suffix tree at linear computational cost. The algorithm [32] scans the input symbolic sequence sequentially and inserts the new input symbol to a set of active suffix nodes. The set of active nodes are the nodes of the tree that have matched the current input symbol and is updated for each new input symbol. This implies that the active node's branch had matched the existing input subsequence. However, Ukkonen's proposed method only operates on symbolic sequences.

To process multivariate time series, the Ukkonen's tree construction algorithm is modified to accept vectors as input, specifically, the insertion criteria have been modified

to compare similarity measures between vectors with an adaptive threshold for insertion. In addition, the tree's internal node is restricted to have only one child to limit the tree's growth for practical realization.

3.2.1 Adaptive Motif Generation (AMG) algorithm

Figure 3.3 shows the overall structure of a tree constructed by the proposed AMG algorithm. Each circle in the tree is a node W_h^k where h denotes the depth and k the branch number of the tree, and K indicates the total number of branches. To manage the tree's growth, the maximum length of each branch will be set to A where $A << N$ and N is the number of input vectors in the sequence. The value of A is database dependent, and should be large enough to capture segments of actual motifs. The selection of A will be discussed in the experimental work.

In the AMG strategy, each internal node is restricted to have only one child, hence, the tree is a special case of the suffix tree. Corollary, each branch of the tree can also be viewed as an entry in an N-gram table.

To capture the repeating segment statistics, each node W_h^k contains four types of information: the node's template vector $\mathbf{v}_h^k$, the time stamp f_h^k where $\mathbf{v}_h^k$ first occurred, a similarity threshold value α_h^k, and a list Ψ_h^k that contains the time stamps of all input vectors that have been inserted into the node.

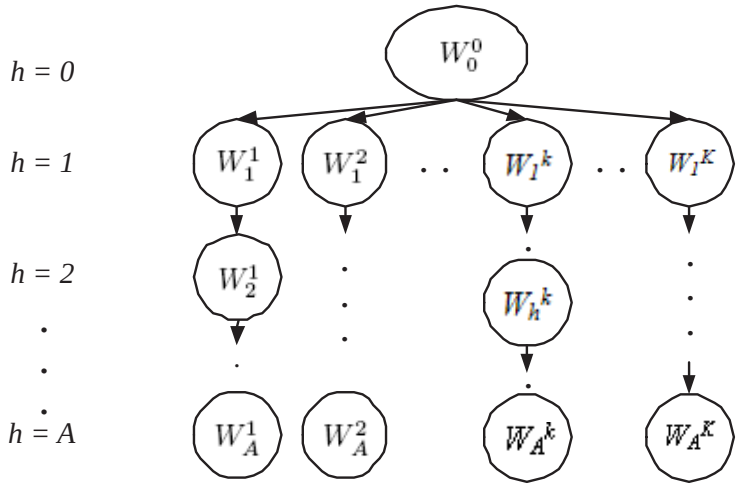


Figure 3.3: The resulting candidate motif representation with AMG algorithm.

Table 3.1: The AMG algorithm used to create a list of K candidate motifs.

// AMG Algorithm
Given input vector sequences $\mathbf{u}_i, i = 1, 2, \ldots, N$

Initialize: create first branch's node W_1^1 using input $\mathbf{u}_1$

(i) User defined values $A, \lambda, \gamma_1, \gamma_2$

(ii) $K = 1, i = 1, h = 1$

(iii) Create W_h^K with $\mathbf{v}_h^K \leftarrow \mathbf{u}_i$, $f_h^K \leftarrow\ i$, $\Psi_h^K \leftarrow \{i\}$, $\alpha_1^k = \gamma_1$.

(iv) $set1 \leftarrow \{W_h^K\}$ // Active branches

(v) $set2 \leftarrow \{\}$

For $i = 2 : N$

$\{\text{modified nodes}\} = f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$
$set2 = \{\text{modified nodes}\}$

For each node $W_h^k \in set1$

- $\{\text{modified nodes}\} = f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$
- $set2 \leftarrow set2 \cup \{\text{modified nodes}\}$

End
$set1 \leftarrow set2$
$set2 \leftarrow \{\}$

End

Before describing the AMG algorithm presented in Table 3.1, the following symbols are defined: $\mathbf{u}_i$ is the input vector at time stamp i, λ is a user defined segment length, γ_1 and γ_2 are user defined threshold values, $set1$ is the set of active nodes and $set2$ is the set of modified nodes during each iteration. The active nodes are nodes that can be extended by the current input vector, and the modified nodes are nodes generated by $f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$ and $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$.

Table 3.1 summarizes the AMG algorithm. The AMG algorithm creates the first node W_1^1 using the first input vector $\mathbf{u}_1$. The 'for loop' (for $i = 2 : N$) is then used to examine subsequent input vectors $\mathbf{u}_i$ to determine if a new branch should be created

as evaluated by $f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$, and/or if the vector is to be appended to existing active nodes as evaluated by $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$. The details of $f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$ and $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$ are discussed in following paragraphs. As the insertion process is similar to [32], the algorithm has linear time complexity.

3.2.1.1 Insertion at root node: $f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$

The function $f_{root}(W_0^0, \mathbf{u}_i, \gamma_2)$ evaluates the input vector $\mathbf{u}_i$ at the root node W_0^0 to determine which of the two hypotheses (**H1**, **H2**) is satisfied. The parameter γ_2 is a user defined threshold to determine similarity decision for **H1**.

- **H1**: $\mathbf{u}_i$ can be used to create a new branch in the tree,
- **H2**: $\mathbf{u}_i$ can be inserted into existing branch(es) of the tree.

The function returns the set of nodes that satisfies either hypothesis.

H1: This condition evaluates if a new candidate motif branch will be created at depth 1. A new branch will be created if no existing branches' template vector is similar to the input vector $\mathbf{u}_i$. In other words, to create a new branch in the tree, the criterion is that the input vector $\mathbf{u}_i$'s similarity to all current depth 1 nodes W_1^k's template vector $\mathbf{v}_1^k$ is less than γ_2. If this condition is satisfied, a new node will be created and this node is returned by the function.

In this work, the Pearson correlation coefficient [96] is used to measure the similarity between two vectors. Specifically, the Pearson correlation between two vectors $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^d$ is defined as

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{r=1}^{d}(x_r - \overline{x})(y_r - \overline{y})}{\sqrt{\sum_{r=1}^{d}(x_r - \overline{x})^2 \sum_{r=1}^{d}(y_r - \overline{y})^2}} \tag{3.1}$$

where x_r, y_r are the r^{th} element of $\mathbf{x}$ and $\mathbf{y}$ respectively, d is the dimensionality of the vectors, $\overline{x} = \frac{1}{d}\sum_{r=1}^{d} x_r$, and $\overline{y} = \frac{1}{d}\sum_{r=1}^{d} y_r$.

H2: To verify if the current input vector $\mathbf{u}_i$ can be inserted into the k^{th} branch of the tree, β_1^k, the similarity between $\mathbf{u}_i$ and the starting node's template vector is evaluated using Eq. 3.2.

$$\beta_1^k = \rho(\mathbf{u}_i, \mathbf{v}_1^k) \tag{3.2}$$

If β_1^k is greater than the node's threshold α_1^k, the corresponding node's Ψ_1^k is updated to include the time stamp i. The set of modified nodes will be returned by the function. The interpretation of the nodes satisfying **H2** is that these branches' starting template vector is similar to the current input vector. The returned set of nodes will become the 'active' nodes that must be verified for the next input vector.

3.2.1.2 Insertion at branch node: $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$

Before discussing the details of $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$, it is better to first discuss the nodes belonging to $set1$ that is used to evaluate $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$. The nodes W_h^k in $set1$ are the 'active' nodes, i.e., the sequence of template vectors at these active nodes' branch from depth of $1 \ldots h$ has matched the recent input vector sequence $\mathbf{u}_{i-h} \ldots \mathbf{u}_{i-1}$. For the current input vector $\mathbf{u}_i$, the function $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$ verifies if this new input vector should be appended to the active branches.

The function $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$ evaluates $\mathbf{u}_i$ for non-root node W_h^k, i.e. $h > 0$ to determine if

- **H3**: $\mathbf{u}_i$ can be used to create a new child node W_{h+1}^k,
- **H4**: $\mathbf{u}_i$ can be inserted into the existing child node W_{h+1}^k.

The function will examine either hypothesis (**H3**, **H4**) and return only one or no modified node.

H3: If W_h^k has no child node and $h < A$, a new child node W_{h+1}^k will be created. This created child node is returned by the function. By growing the branch with a new child, the operation can be viewed as extending the current candidate motif by 1 vector.

H4: If W_h^k has a child node, a test is used to verify if the current input vector $\mathbf{u}_i$ can be inserted into the existing k^{th} branch of the tree. Different to **H2**'s Eq. 3.2 which evaluate only the similarity between the current input vector to one node's template vector, Eq. 3.2 is extended to measure the similarity between the current input vector sequence to the sequence stored in branch k, i.e. in $W_{1\ldots(h+1)}^k$. Specifically, the segment similarity is evaluated by

$$\beta_{h+1}^k = \frac{\sum_{r=0}^{min(h,\lambda-1)} \rho(\mathbf{u}_{i-r}, \mathbf{v}_{h+1-r}^k)}{min(h, \lambda - 1)} \tag{3.3}$$

To verify if $\mathbf{u}_i$ can be inserted into the k^{th} branch of the tree, β_{h+1}^k is compared to threshold α_{h+1}^k. If β_{h+1}^k is greater than α_{h+1}^k, the input vector time stamp information i will be included by Ψ_{h+1}^k and the function $f_{branch}(W_h^k, \mathbf{u}_i, A, \lambda)$ returns the node W_{h+1}^k, otherwise null.

In this system, each α_{h+1}^k is adaptive so that the sequence being analyzed can influence the threshold value. It makes the insertion process more robust as opposed to having a single fixed threshold. The α_h^k is updated by

$$\alpha_{h+1}^k = \alpha_h^k - (\beta_h^k - \beta_{h-1}^k), \tag{3.4}$$

where $h \geq 1$, and $\beta_0^k = \beta_1^k$. The proposed adaptive threshold α_{h+1}^k has the following behaviors: α_{h+1}^k decreases when the past segment similarity β_h^k is better than its predecessor, and α_{h+1}^k increases when β_h^k is poorer than its predecessor. The adaptive threshold is modified thus so that sequences which produce poor intermittent segment similarity will be retained if past segment similarity has been good.

3.2.2 Parameter settings

The following four parameters control the motif discovery process in AMG algorithm: A, λ, γ_1 and γ_2. The parameter A specifies the maximum depth of the tree and its value should be set to a length that can capture a meaningful portion of the repeating segment. The parameter λ specifies the length of the segment to evaluate the similarity measure as specified in Eq. 3.3. The value of λ should be chosen long enough to capture a candidate repeating segment. The threshold γ_1 is used to initialize α_1^k and its setting is important since it determines if each new input vector will be inserted into the branch. An unreasonably high setting will stop a true repeating sequence from being inserted, while an unreasonably low setting will generate too many false alarms. Hence, the value of γ_1 is critical to the success of AMG. In this work, a small training corpus is used to determine γ_1 by evaluating all pairwise similarity between the vectors and selecting the top 10^{th} percentile pairwise similarity value as γ_1. The threshold γ_2 is used in the evaluation of hypothesis **H1**. The value determines whether a new branch should be created for the input vector. The suitable value for γ_2 is $\gamma_1 < \gamma_2 \leq 1$. A higher γ_2 value will allow more branches to be created. In this work, γ_2 is set to the top 5^{th} percentile pairwise similarity value of the training set.

3.2.3 AMG vs. Ukkonen's suffix tree construction

The AMG algorithm emulates Ukkonen's algorithm to perform online construction of a suffix tree. In Ukkonen's original algorithm, each new input symbol will always be inserted into the tree at the active nodes. The insertion criteria for symbolic suffix tree are based on exact match evaluation which compares the input symbol to the node's template. In contrast, the insertion criteria for AMG are not as straight forward since the input are vectors - this implies that the similarity comparison must use threshold for the decision process. However, a simple pairwise similarity evaluation of current input vector to the node's template vector cannot reflect the global correlation of two sequences and hence is not robust. To overcome this shortcoming, AMG employs a segment-to-segment similarity measure with an adaptive threshold to evaluate if the current vector should be inserted into the tree. Hence, the insertion process can tolerate intermittent poor pairwise similarities measured with candidate motifs. Finally, the AMG is also different to the general suffix tree construction as each non-root node is restricted to have only one child - this restricts the tree's growth and allows for a practical realization.

3.3 Candidate Motif Refinement

In Figure 3.3, each branch of the AMG generated structure represents a candidate motif. These candidate motifs must be further processed to extract true repeating patterns because: (i) the candidate motifs will often contain redundant patterns such as trivial matches; (ii) the candidate motifs can be merged to form longer patterns, (iii) the AMG algorithm may miss detecting true motifs, and (iv) the boundaries of the found motifs should be aligned.

To address the above four issues, a direct approach is to capture the candidate motifs' occurrences in an $N \times N$ matrix where N is the number of vectors in the input sequence, and then refine the obtained patterns to generate the final selection. This matrix is named the candidate motif matrix, and the elements of this matrix whose values are '1' indicate the occurrence of a repeating segment. The interpretation of this matrix is similar to the binarized self-similarity matrix used for motif discovery[33]. Hence, existing techniques such as [33, 47] to refine the similarity matrix and pattern merging

can be directly applied. In addition, missing motifs can also be immediately identified from this matrix (see Section 3.3.3).

Different from [33, 47], the proposed candidate motif matrix generated by the AMG is a sparse $N \times N$ matrix as opposed to the conventional method which generates the full $N \times N$ self-similarity matrix. The experimental results (Section 3.4) show that the occupancy of the candidate motif matrix is only about 1% of the full matrix and the memory usage to construct the AMG structure is $O(AK)$.

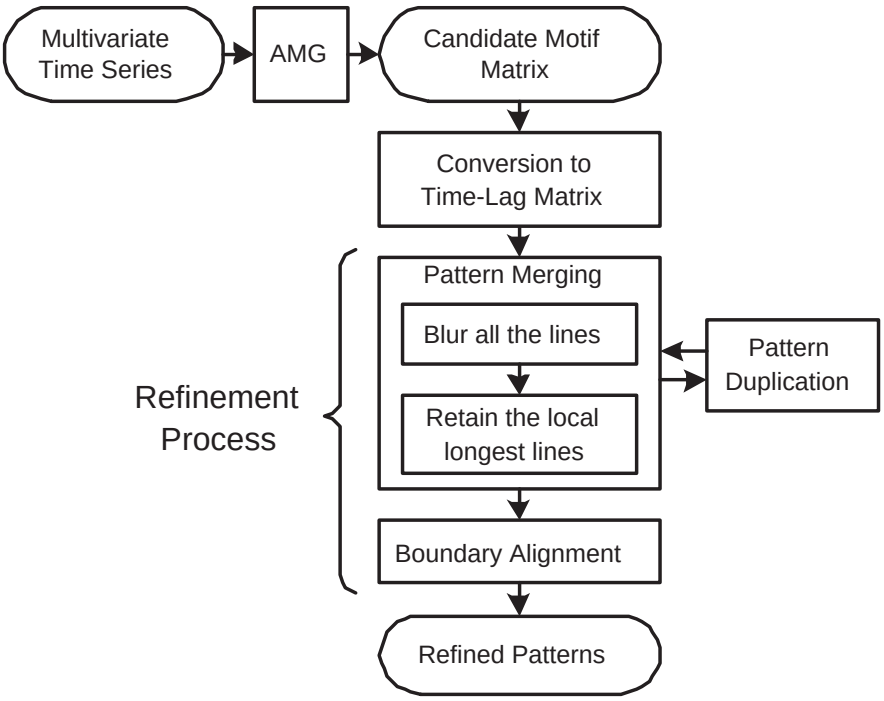


Figure 3.4: The proposed process to refine the candidate motifs obtained by AMG algorithm.

Figure 3.4 illustrates the proposed refinement framework. The candidate motifs generated by AMG is first used to populate the sparse candidate matrix. This matrix is subsequently converted to a time-lag matrix similar to [33, 47] for the refinement process such as: pattern merging, pattern duplication and boundary refinement. The details of the procedure are briefly described in the following sub-sections.

3.3.1 Candidate motif matrix generation

The candidate motifs can be easily retrieved from the structure (Figure 3.3) constructed by AMG by detecting Ω, the set of nodes W_h^k which satisfy the following conditions:

$$h > \lambda, \tag{3.5}$$

$$|\Psi_h^k| > 1, \tag{3.6}$$

$$|\Psi_h^k| \neq |\Psi_{h+1}^k|, \tag{3.7}$$

where "| |" indicates the number of elements in the set. Eq. 3.5 restricts the length of candidate motifs to be at least λ, Eq. 3.6 guarantees that the captured candidate motifs have occurred at least twice, Eq. 3.7 is used to choose the set of detected patterns with the longest length.

These detected patterns are used to populate the candidate motif matrix by assigning '1' to the positions where the patterns occur, specifically, the repeating patterns are at coordinates $(\Psi_m^k(j), f_m^k)$ of $W_{m=1\ldots h}^k$ where $W_h^k \in \Omega$, and $\Psi_m^k(j)$ are the elements of Ψ_m^k

For the convenience of processing, the candidate motif matrix is then converted to a time-lag matrix $\mathbf{T}$ [33], illustrated in Figure 3.2(d). By this conversion, the repeating patterns will be reflected as parallel vertical lines in $\mathbf{T}$ and redundant neighboring patterns can be easily detected and removed.

3.3.2 Pattern merging

During the construction of the AMG structure, the algorithm will record every possible candidate motif. Hence, many neighboring candidates will be generated. These neighboring motifs should be merged to find longer repeating patterns. To merge these neighboring motifs, each vertical line in the time lag matrix is first dilated and then eroded to form a new sparse matrix $\mathbf{T}'$. The objective is to extract the longest vertical line for each neighborhood.

3.3.3 Pattern duplication

If the input sequence is noisy, the AMG algorithm may not be able to capture every repeating pattern of a candidate motif. However, the repetition occurrence in the candidate motif matrix can be further analyzed to sought out missing repeating pairs. To describe it clearly, the following scenario is assumed: if segments Q_1, Q_2 and Q_3 are sequences of the same motif, but due to noise, the sequences $\{Q_1, Q_2\}$ were found under one candidate motif selection, while the sequences $\{Q_2, Q_3\}$ were found in a separate motif selection. Let δ_{jk} denote the frame index differences between pattern $\{Q_j, Q_k\}$. Due to the characteristics of the time-lag matrix, the pattern Q_2 will be marked in two locations, one is a vertical line that is parallel to Q_1 with a distance of δ_{12} along the x-axis, and the other vertical line is along the y-axis with a distance of also δ_{12} to Q_1(see Figure 3.5).

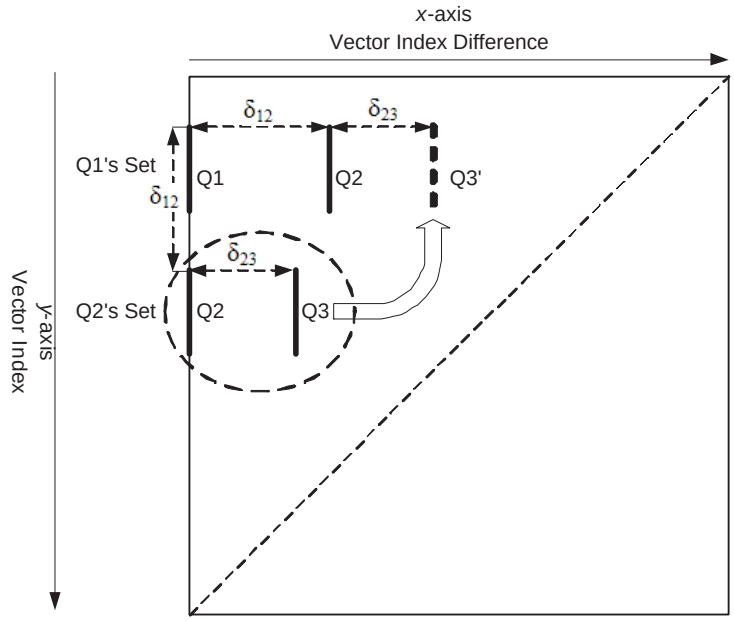


Figure 3.5: An illustration of the repetition occurrence and pattern duplication process in the time-lag matrix.

In this situation, a conclusion can be inferred from the time-lag matrix: Q_3 should also be included in Q_1's set [33]. To exploit this relationship, the refinement process will duplicate all vertical lines in $\mathbf{T}'$ to all possible candidate positions belonging to the same family. The process is illustrated in Figure 3.5.

3.3.4 Boundary refinement

Noise may also cause mis-alignments in the boundaries of the obtained candidate motifs. The boundaries can be aligned by either extending or shrinking the detected patterns. The details of the boundary alignment process can be found in [33]. After the refinement process, the final motifs can be easily retrieved from matrix $\mathbf{T}'$.

3.4 Experiments

To verify the proposed AMG algorithm, two types of data were used to evaluate the performance: 1) a 30 songs music database, and 2) numerical sensory data sets. The performance of the proposed method is compared with related techniques to demonstrate the effectiveness of AMG.

3.4.1 Repeating segments detection in music

3.4.1.1 Experiment setup

The test corpus consists of 30 pieces of English pop songs performed by both male and female singers[1]. The total duration of the 30 songs is about 2 hours. From the test corpus, 63 motifs (i.e. repeating segments) with an average duration 25.6sec are manually identified. The repeating segments are defined as segments that have similar melody in this paper. There are a total of 198 instants for these motifs. These instants are used as the ground truth patterns in this experiment.

The feature used is derived from the 12 dimensional chroma vector [16] calculated from every 250msec music segment with 50msec hop size. In this implementation, the spectrum information from seven musical octaves that span from 32.5Hz to 4000Hz are adopted. Each extracted chroma vector is normalized by dividing each element by the vector's maximum value. As the chroma feature captures information in a 250msec window, slight offset of the analysis window may affect the feature values. As such, researchers [33] have proposed to generate features from a much longer analysis window (1 second). Following their strategy, the extracted feature is the mean and variance of 16

[1]The 30 songs' feature database and repeating segments location information are released at http://www3.ntu.edu.sg/home/aseschng/SpeechTechWeb/projects/projects.htm.

consecutive chroma features with a hop-size of 3 frames. This translates to an analysis window of 1 second and a shift of 150msec to generate a feature vector of 24 dimensions.

This experiment compares the performance of the proposed AMG to the traditional self-similarity matrix approach [33]. The self-similarity approach [16, 33, 47] has been successfully applied to discover music repeating segments by constructing a full $N \times N$ matrix. In this experiment, both approaches will generate the time lag matrix using the same features and are post processed by the same refinement framework as described in Section 3.3. Hence, the proposed approach can be viewed as identical to the self-similarity approach except for the strategy to generate candidate motifs.

The parameter settings used for the AMG approach are: maximum depth of the tree $A = 100$ (this setting allows the system to capture 16sec music segment), the segment length $\lambda = 61$ (approximates 10sec duration). The experimental results show that the values of A and λ are not critical to the success of motif discovery as long as the value of A and λ are sufficiently large to capture a portion of the motif pattern. The critical parameters are the threshold values of γ_1 and γ_2. In this experiment, an independent training corpus of 20 English songs was used to determine suitable values for these two parameters - for the experiment, γ_1 and γ_2 were set to be 0.367 and 0.684 respectively.

3.4.1.2 Evaluation criteria for music data

To evaluate the performances of the two approaches, the repeating segments in $\mathbf{T}'$ generated from Section 3.3 are compared with the ground truth patterns. Let $\mathbf{G}$ denote the ground truth patterns in a sparse binarised time-lag matrix for each song. The vertical lines in matrix $\mathbf{G}$ are duplicated as described by Section 3.3.3. To achieve a robust and stable evaluation, a $\pm$1sec window mismatch among repeating segments is allowed. This is achieved by blurring all the lines in $\mathbf{G}$ to form a new matrix $\mathbf{G}'$ for the $\pm$1sec offset error. Therefore, the correctly detected repeating segments can be found in matrix $\hat{\mathbf{T}}$ by $\hat{\mathbf{T}} = \mathbf{T}' \wedge \mathbf{G}'$, where "$\wedge$" denotes the element-wise logical "AND" operator.

In this paper, the recall and precision are defined as $Recall = \frac{|\hat{\mathbf{T}}|}{|\mathbf{G}|}$ and $Precision = \frac{|\hat{\mathbf{T}}|}{|\mathbf{T}'|}$, where "$|\ |$" denotes the number of non-zero elements in the matrix.

3.4.1.3 Evaluation results for music data

The average recall, precision and F1-measure of all the testing songs are shown in Table 3.2. As the self-similarity approach calculates the similarity between all pairwise vectors, the matrix will contain all similarity information in the sequence for motif detection, hence the ability to extract and find correct motifs relies on the pattern refinement process. Since the refinement process and features used are the same for the two approaches, the experiment measures how well the AMG approach can capture the "correct" candidate motifs. The results in Figure 3.6 and Table 3.2 show that the two approaches have similar performance.

Table 3.2: Average performance comparison between the AMG and self-similarity approach for 30 songs.

	Recall	Precision	F1
AMG	0.590	0.560	0.575
Self-Similarity	0.569	0.558	0.564

To show the efficiency of the AMG approach, the memory usage between the two methods were compared. The AMG structure (Figure 3.3) stores all the similarity information in each node and hence requires $O(AK)$ memory space. However, the self-similarity approach requires $O(N^2)$ memory space to store all pairwise similarity measurements. To illustrate the difference in memory usages between the two approaches, the ratios AK/N^2 and K/N are plotted in Figure 3.7. The element occupancy of the candidate motif matrix was also evaluated for all 30 songs, and the results showed the candidate matrix was 99% sparse.

To compare the computation complexity requirement between the self-similarity approach and the AMG approach, the computational requirements needed to create the sparse time-lag matrix by these two approaches were measured (Figure 3.4). For the self-similarity approach, it requires $N(N-1)/2$ number of pairwise similarity measures (Eq. 3.1). For the AMG approach, the total number of vector-to-vector similarity comparisons for the 30 songs were recorded - The results show that the AMG approach requires an average of 10% computation cost as compared to the self-similarity approach

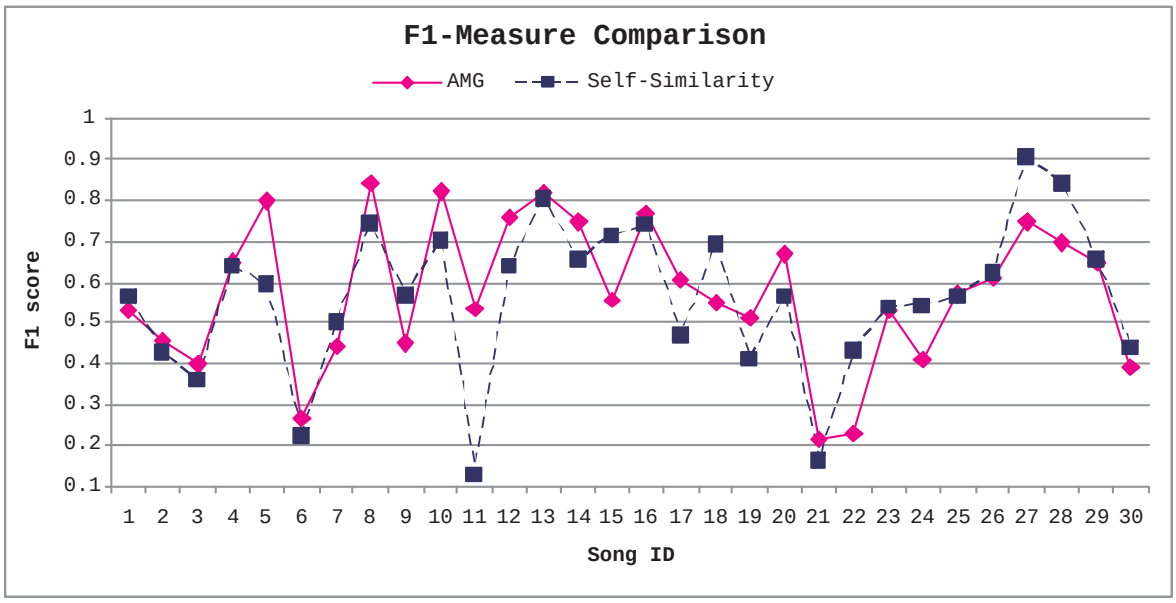


Figure 3.6: The F1-measure for each individual song using the self-similarity approach and the proposed AMG approach.

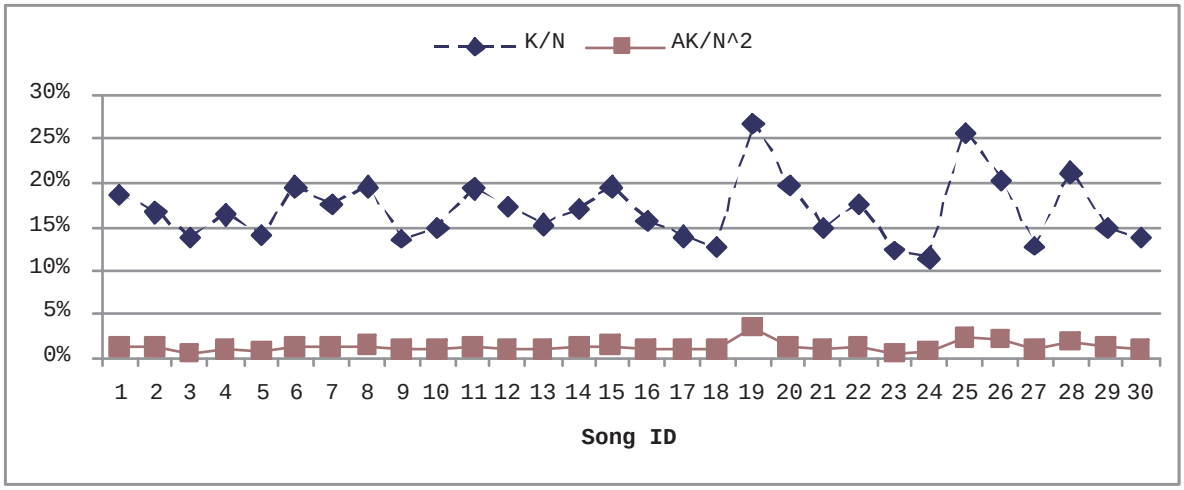


Figure 3.7: Comparison on memory usage requirements between AMG and self-similarity approach for each individual song. The average memory requirement of AMG is approximately 2% of that required by the self-similarity approach. The ratio K/N is plotted to illustrate the number of branches created for each song over the total number of input vectors.

for the music database test (Figure 3.8). Furthermore, the self-similarity approach requires additional processing such as low-pass filtering [47] and other image processing techniques [33] to generate the sparse time-lag matrix.

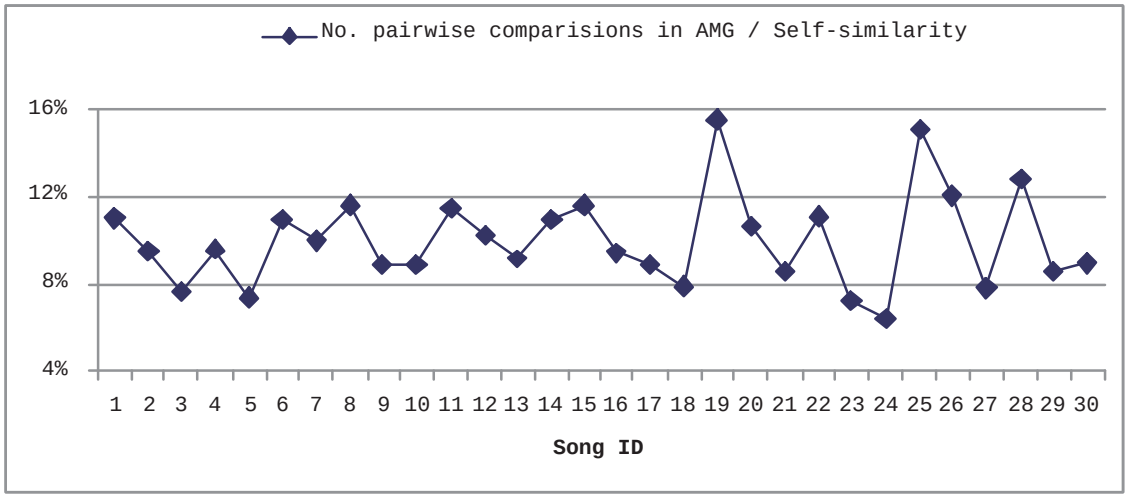


Figure 3.8: Comparison on computational requirements between AMG and self-similarity approach for each individual song.

To further investigate the performance of the proposed method, it was applied to discover motifs from two published data sets.

3.4.2 Motif discovery from sensory data

3.4.2.1 Experiment setup

In this set of experiments, the proposed AMG was applied to automatically detect motifs from two publicly available sensory data sets: fetal ECG data set [101, 138] and shuttle data set [101, 138, 139].

The fetal ECG data is an eight dimensional time series with obvious periodicity and each dimension represents the measurement from one sensor. The AMG algorithm was applied on the raw ECG data directly followed by the refinement process. The parameter settings used for the AMG were selected empirically: maximum depth of the tree $A = 100$ (i.e. 0.4 second of data with sampling rate 250Hz), the segment length $\lambda = 60$, the thresholds γ_1 and γ_2 were set as 0.6 and 0.8 respectively.

The second sensory data set is a six dimensional shuttle time series data. This data set has a motif that appears only twice. The two occurrences of the motif have similar shape in all the dimensions but have different mean values in two of the six dimensions as shown in Figure 3.10. To process this data set, the first derivative (delta) of the features were used as the input. The parameter settings used for the AMG were selected empirically: maximum depth of the tree $A = 100$, the segment length $\lambda = 60$, the thresholds γ_1 and γ_2 were set as 0.6 and 0.8 respectively.

3.4.2.2 Experimental results for sensory data

Figure 3.9 shows the motifs found by AMG algorithm in the ECG data set. To compare with [101], the found motifs' location projected on the first principal component of the data set is plotted. The result shows that 12 motifs can be detected and their time locations are similar to the findings in [101].

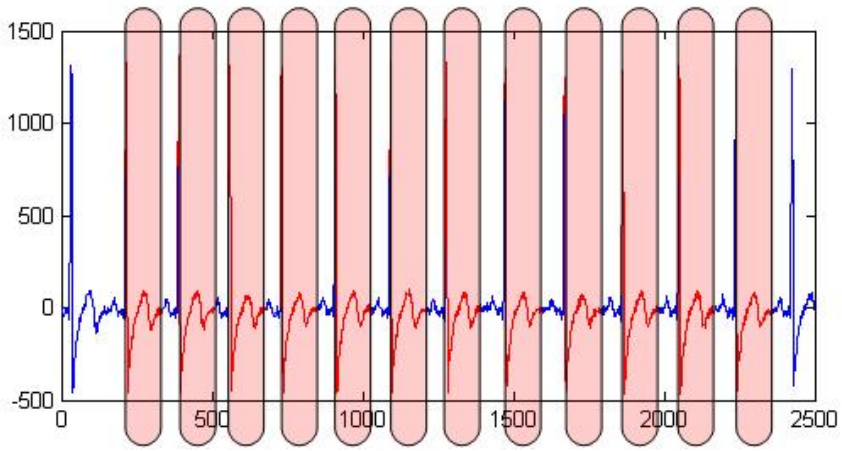


Figure 3.9: The motif discovered in the 8 dimensional ECG data.

Figure 3.10 shows that the AMG can simply find the same motif in shuttle data set as in [101, 139].

Table 3.3 shows that the memory usage requirements for the two data sets. As the ECG data is periodic (Figure 3.9), AMG needed only $K = 67$ branches under the root node to capture all the recurrent patterns. For shuttle data set (Figure 3.10), AMG created only $K = 23$ branches under the root node to capture the two recurrent patterns.

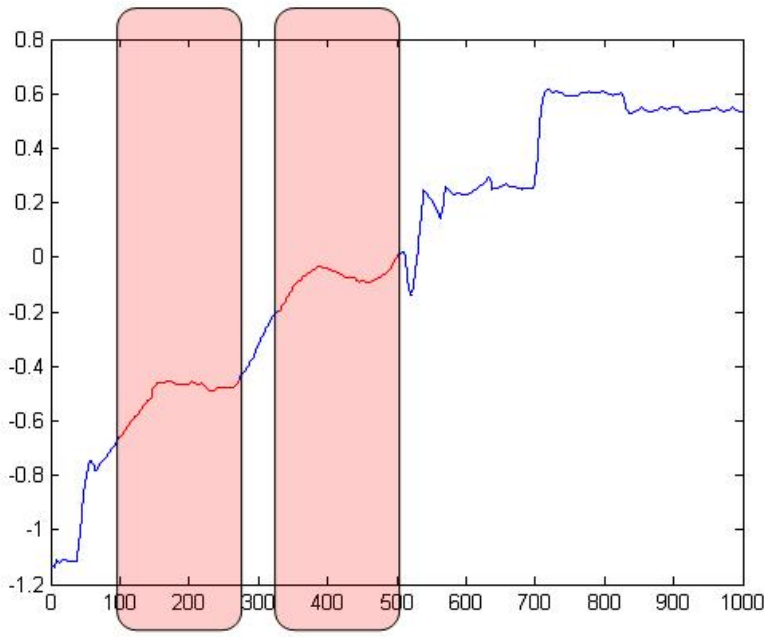


Figure 3.10: The motif discovered in the shuttle data set using all 6 dimensional information. The above plot shows the 2nd dimension of the data; The motif in this dimension has different means. This output is similar to [101, 139].

Table 3.3: Memory usage requirements for the two sensory data sets.

	K/N	AK/N^2
ECG	0.027	0.001
Shuttle	0.023	0.002

Measurement was also done for computational requirements needed to create the sparse time-lag matrix for refinement process. Compared to calculating all the pairwise similarity measurements, AMG requires only 4.1% of the computation cost for both ECG and shuttle data sets.

The above results and observations show that the proposed AMG followed by the refinement process is able to produce similar performances compared to existing studies [101, 139] on the same two data sets.

3.5 Conclusion

In this chapter, a novel approach to detect repeating patterns in a sequence of multivariate vectors is proposed. The online suffix tree construction algorithm is extended to accept vectors as input so that the exact matching criteria are avoided. To make the algorithm more robust, segment-to-segment similarity is used to smooth the outliers in the sequence. The candidate motifs obtained by the AMG method are then refined using a sparse similarity matrix to find the final motifs.

The proposed AMG has been applied to automatically detect repeating segments in music and two time series data sets. The experimental results show that the proposed approach was able to find repeating segments with a similar detection accuracy as that of traditional self-similarity approach [33] but at a considerably lower computation cost and memory requirement.

Due to its success in repeating segments discovery in music, an immediate question could be: Can AMG be applied to discover speech patterns such as words or phrases? Unfortunately, music and speech signals are different in natures. The repeating segments in music usually have similar length and the pitch of a music note is relatively stable, however the duration and pitch of the same speech pattern, e g. an English word, can be very different. Hence, a novel approach that is robust to speech data will be needed. In the following chapter, an iterative approach to model merging to discover speech pattern is examined.

Chapter 4

Automatic Speech Pattern Discovery using Unsupervised Approaches

This chapter focuses on the problem of automatic discovery of recurrent patterns in an untranscribed speech corpus. The recurrent patterns of interest refer to keywords or phrases that occur repeatedly and have been shown [13, 60] that they can be used to detect the salient point of the corpus topic. For example, it is possible to understand the topic of a lecture recording by detecting frequently occurring and long words or phrases. Given an untranscribed spoken document, this chapter examines unsupervised techniques to identify the recurring keywords/phrases without the need of ASR systems, which require transcribed data for training.

The automatic speech pattern discovery is a difficult research problem due to the variations of speech signals. The speech signals of a speech pattern can be affected by different factors such as speakers' characteristics, speaking styles, transmission channels and background noise [140]. As a result, the same words or phrases manifest different acoustic characteristics in different occurrences. An example is demonstrated in Figure 4.1 which shows that the spectrograms of the same English word (in this example: "Four" and "Three") spoken by the same speaker in the same utterance can be different. In this example, the difference can be attributed to context, i.e. the position of the word in a sentence, and to the non-repeatability nature of a human speech generation system. It is clear that robust models and detection techniques are required for reliable speech pattern discovery.

In this chapter, an unsupervised technique is proposed to discover speech patterns in a multi-speaker corpus. The proposed technique is based on statistical modeling of

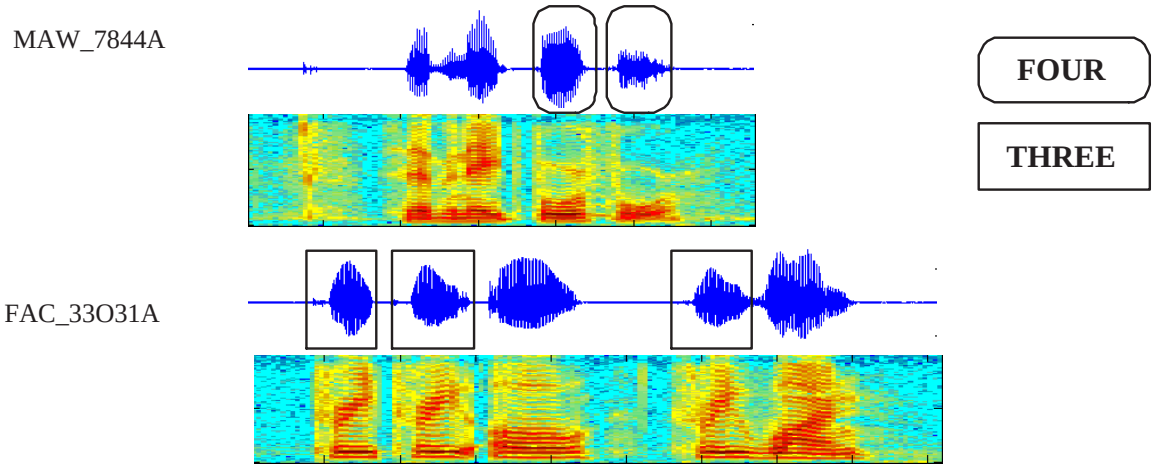


Figure 4.1: An demonstration of variations of speech patterns in clean speech using TIDIGITS database. In the utterance MAW_7844A, the spectrograms of the word "Four" are different although spoken by the same speaker. In the utterance FAC_33O31A, the spectrograms of the word "Three" are also different.

speech features which can address speech variations effectively. The following sections are organized as follows: a brief review of related techniques is provided in Section 4.1. The proposed technique, namely iterative approach to model merging, is described in Section 4.2. Section 4.3 discusses the experimental results, and the conclusion is given in Section 4.4.

4.1 Introduction

In this thesis, the term "speech pattern discovery" refers to the task to automatically identify frequently occurring speech segments in an untranscribed corpus. The speech segments of interest are linguistically meaningful patterns such as words and phrases in human languages. According to a recent study in cognitive science [141], long words tend to carry more information than short words, i.e. the length of words is related to how much information they convey. For instance, in English, shorter words such as "a", "is" and "the" carry less information than longer words such as "president" and "immediately". Since the lengths of written and spoken words are closely related, such finding is also true for spoken documents. For example, the studies [13, 60] show that the topic of a lecture recording is related to a set of selected frequently occurring

words/phrases which are longer than a predefined duration (0.5 seconds). In this work, the objective is to detect not only recurrent patterns but also speech patterns with long durations.

If ASR systems are available for the target speech language, they can be used to transcribe the speech into text to discover repeating patterns (i.e. words or phrases) from the text directly. However, such approach has several limitations. Firstly, the cost of training an ASR system is high as it requires a large amount of transcribed speech data and a well prepared pronunciation lexicon. Secondly, the quality of the recognizer's output, e.g. word recognition accuracy, may be affected by different factors that are difficult to control. For example, speaker variation and environment noise both can seriously affect speech recognition [140]. Thirdly, speech data may contain words that are not in the lexicon of the ASR systems, i.e. OOV words, which cannot be recognized. The OOV words are however important as they can be names of places or people. Due to these limitations, automatic speech pattern discovery techniques which do not rely on ASR systems are desired [13].

To directly search through the speech data to discover recurrent patterns in an unsupervised manner, DTW-based techniques were applied on speech feature vectors to identify recurrent speech patterns by comparing pairwise speech segments [13, 60, 133]. In [133], speech utterances are first segmented by detecting word boundaries, and for each pairwise segments, DTW is applied on the pairwise distance matrix to find the best alignment between them. If the distance between a pair of patterns is smaller than an empirical threshold, the pair is treated as a matched pair. Experimental results in [133] have shown that such approach is able to identify repeating words spoken by the same person. To improve the efficiency in identifying repeating patterns, researchers in [13] proposed segmental DTW. This approach first constructs a distance matrix of two utterances and then divides the distance matrix into diagonal regions for segments alignment, as discussed in Chapter 2. The segmental DTW has been applied to automatically generate keywords in single-speaker lecture recordings for speech summarization application [60]. The DTW-based methods however have only been shown to be effective in finding repeating patterns for a single speaker's speech corpus [13, 60, 133]. While the speech variation of a single speaker is relatively small, the variation between different speakers can be

significant. As DTW lacks robustness in handling such variations in the features, it is unlikely that DTW-based methods will work well on multi-speakers' data [142].

To achieve speaker-independent pattern discovery, data-driven modeling techniques such as ASM [18], fenones [119] and SSS [120–122] as reviewed in Chapter 2 have been suggested. In these work, HMM was used to model the temporal dynamics of speech features. In each HMM state, a GMM is used to model the distribution of speech features of a particular acoustic unit. As there are many mixtures in the GMM, an HMM state is able to represent the features of different speakers and hence achieve speaker independent modeling of speech [142]. For example, in ASM [18], sub-word units are automatically found from untranscribed speech corpus. Each sub-word unit is represented by a 3-state HMM and trained using Baum-Welch algorithm. After convergence, the sub-word units represent phone-like units of a language [18]. For example, similar sounds such as fricatives are represented by a single HMM. As the ASM, fenones and SSS were originally proposed to learn sub-word units for unsupervised speech recognition, the learnt units are too short for our task in this chapter.

Another approach for pattern discovery is to first convert speech data into token sequence, and then apply symbolic pattern discovery techniques on the token sequence to discover repeating patterns. In [106], speech utterances are first divided into short segments and clustered to convert speech utterances into token sequences. DP is then applied on pairwise utterances to detect repeating subsequences. In another study, to overcome the OOV problem of ASR system, Nowell and Moore [94] used a phoneme recognizer to convert speech data into phoneme sequences and then applied local alignment technique used in genomic sequence discovery to detect similar portions from these phoneme sequences. As the tokenization process can be viewed as quantization of speech segments into a limited number of classes, the subsequent symbolic pattern discovery process may suffer from "quantization error" which is difficult to recover.

In the following sections, a technique is proposed to overcome the limitations of the above mentioned 3 speech pattern discovery approaches. The proposed technique is an extension of the modeling approach which is more robust than feature vector matching approach such as DTW-based techniques for speaker-independent speech recognition task [142]. To address the problem of modeling approach, i.e. the learnt units are usually

sub-word units rather than long patterns, sub-word units are then merged to form longer patterns. The merging process is iterative and unsupervised without the aid of linguistic knowledge. In each iteration of model merging, all model parameters are re-estimated by Baum-Welch algorithm so that the models can have better representation of the data. The "quantization error" in symbolic pattern matching approach is hence reduced. To examine the performance of the proposed technique, experiments have been conducted on TIDIGITS corpus to automatically discover the English digits. The experimental results showed that 10 out of 11 digits were detected for both male and female data. This work can be further applied to explore unknown languages by automatically identifying the frequently occurring words or phrases. Hence, it can be used to build a preliminary lexicon for an unknown language.

4.2 Iterative Approach to Model Merging

To discover speech patterns in untranscribed corpus, the proposed technique consists of 3 modules as illustrated in Figure 4.2.

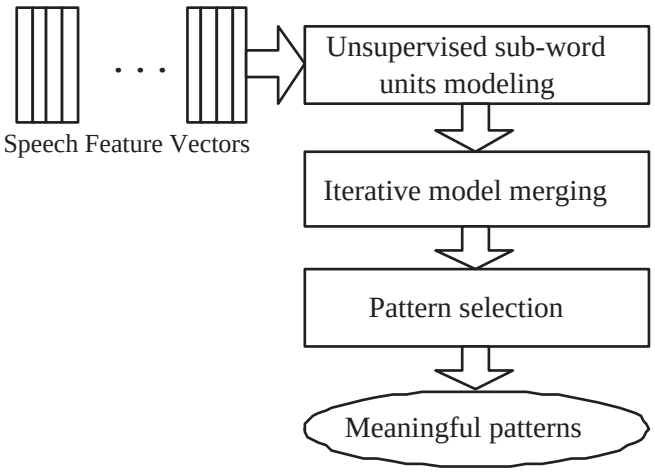


Figure 4.2: A diagram of the proposed technique to discover speech patterns.

In the first module, a set of sub-word acoustic units are first learnt from the speech data using unsupervised modeling technique. These acoustic units should have similar durations as phonemes and are used to represent the entire acoustic space of the speech data. Specifically, in this work, the acoustic units are generated using the ASM approach

which is a data-driven modeling technique [18]. The acoustic units created are modeled by a 3-state left-to-right HMM.

In the second module, the acoustic units are merged to form models representing speech patterns in an iterative process. To achieve this goal, a critical question is how to select sub-word units to merge? In Section 4.2.2, a model merging criterion is examined to merge adjacent models if these adjacent models occur frequently together in the speech data. Merging models means to concatenate the HMM states of the separate models to form a new HMM with more states. In each iteration of the process, only two models are merged and the new model is added to the model set. All the model parameters are then re-estimated using the Baum-Welch algorithm. The iterative process will stop when a stopping criterion is met, and it produces a set of HMMs that model different acoustic patterns such as sub-word units, syllables, words/phrases and silence segments.

In the last step, a pattern selection module is then used to identify the patterns of interest from the set of all patterns. In the following sections, these 3 modules are described in details.

4.2.1 Unsupervised Sub-Word Units Modeling

This section describes the modeling technique to generate sub-word models. As this work aims to discover speech patterns in untranscribed or unknown corpus, the linguistic information of such corpus is unknown. The conventional acoustic units such as phonemes or words are thus unavailable and hence an unsupervised technique is required to generate acoustic models to represent the entire acoustic space of the speech corpus.

The ASM was introduced in [18] to learn data-driven acoustic units in untranscribed speech corpus. The core modeling technique used in ASM is the conventional HMM. The training of the ASM however does not require manual transcription to estimate the parameters of the HMMs. Instead, vector quantization is applied to automatically produce pseudo labels for the training procedure.

The procedure to generate ASM models is discussed in this section. The feature vectors such as MFCC are first extracted from the speech data. The ASM training process is then applied to generate a set of HMMs, denoted as M_i $(i = 1 \ldots m)$ where m is the number of models, in the following manner:

Step 1) Speech segmentation The speech utterances are first divided into short segments by detecting boundaries between two successive acoustic units, e.g. speech phoneme boundaries. In this work, speech segmentation implementation follows the method described in [106]. The method examines each feature frame by measuring the distance between the average feature vectors before and after the frame as shown in Eq. 4.1. The frames with local maximum values are selected to be phoneme boundaries, i.e. the distance value on the boundary frame is higher than both of its neighboring frames.

$$d(\frac{v_{i-2}+v_{i-1}}{2}, \frac{v_{i+1}+v_{i+2}}{2}) \cdot \log(E) > \delta \tag{4.1}$$

where, v_i is the feature vector at time index i. The log energy, $\log(E)$, is used as a weighting factor, and the pre-defined threshold δ is used to avoid the flagging of segments during silent portions so that only the frames with distance values above δ are considered to be boundaries. The distance function $d(v_1, v_2)$ is defined as

$$d(v_1, v_2) = \arccos(v_1' v_2 / (v_1' v_1 v_2' v_2)^{1/2}) \tag{4.2}$$

where, v_i' is the transpose of v_i.

Step 2) Initial label generation Cluster the segments into m clusters using k-means algorithm. In each segment, the mean vector is calculated to represent the segment. The k-means algorithm is then applied on the mean vectors into m clusters, and segments are then labeled with the corresponding cluster identity.

Step 3) Model initialization Create m left-to-right 3-state HMMs. Estimate the parameters of the m HMMs using Baum-Welch algorithm on the training corpus with the labeled cluster identity found in Step 2.

Step 4) New label generation The trained HMMs are used to decode the training corpus. The decoding output is retained as the new labels of the training corpus.

Step 5) Model parameters re-estimation The HMMs are re-estimated using the new labels found after Step 4.

Step 6) Iterative training Repeat Steps 4 - 5 till the average likelihood score of all the feature vectors converges.

Although the ASM training process does not rely on the true transcription of speech data, it is still necessary to manually determine the number of acoustic units. Let $\hat{m}$ represent the actual number of basic acoustic units such as phonemes in the corpus, the pre-defined number of acoustic units, m, should be set as close to $\hat{m}$ as possible. For example, if the language is English, m can be set to 40 as there are 39 phoneme models and 1 silence model in a typical English recognizer. Theoretically, if $m > \hat{m}$, a single phoneme may be represented by multiple acoustic units as there are more acoustic units than phonemes. As there are variations in phonemes due to different factors, e.g. speakers and contexts, different acoustic units may represent variations of the same phoneme. If $m < \hat{m}$, different phonemes could be represented by a single acoustic unit, e.g. similar fricative phonemes /f/ and /s/ may be combined.

In this work, it was found that it may be beneficial to slightly underestimate m due to two reasons. Firstly, if one phoneme is split into more than 1 acoustic units, the same pattern will be represented by different sequences of acoustic units. The redundant representations of patterns are undesired in many potential speech pattern discovery applications such as spoken documents tagging. Secondly, the confusion between phonemes caused by underestimating m can be reduced as the HMMs of the basic acoustic units and merged models are re-estimated in each iteration as discussed in the next section.

After the training process of sub-word units are converged, a set of m HMMs $\mathbf{\Gamma} = \{M_1, \ldots, M_m\}$ are obtained. Each HMM M_i consists of n_i states, and the transition probability from ρth state to ϱth state of M_i is represented by $a^i_{\rho\varrho}$. The model set $\mathbf{\Gamma}$ will be used as the initial models in the second module as shown in Figure 4.2. In the next section, information theory is applied on the decoding output to identify the models which to be merged.

4.2.2 Iterative Model Merging

Given the initial set of sub-word HMMs $\mathbf{\Gamma}$, this work proposes to iteratively merge and grow the HMMs towards longer speech patterns. To simplify the process, a 2-step approach is adopted for this purpose. In the first step, namely tokenization step, each

utterance in the speech corpus is converted into a sequence of model IDs, or called tokens, using the HMMs in $\mathbf{\Gamma}$.

In the second step, statistical information of the token sequences is used to select two models for merging and the newly created model is added to the model set $\mathbf{\Gamma}$. To formulate the problem, the following symbols are defined: let $\mathbf{S} = (S_1, \ldots, S_U)$ denote the token sequences of the speech corpus, where $S_u (u = 1 \ldots U)$ denotes the token sequence for the u^{th} utterance. Each token sequence is represented by $S_u = (\omega_1^u, \ldots, \omega_t^u, \ldots, \omega_{|S_u|}^u)$, where ω_t^u denotes the t^{th} token of the sequence S_u and $|S_u|$ is the number of tokens. Each token ω_t^u takes value from the set of acoustic model names $\mathbf{\Gamma} = (M_1, \ldots, M_m)$. An example token sequence is shown in Figure 4.3. The model selection problem is

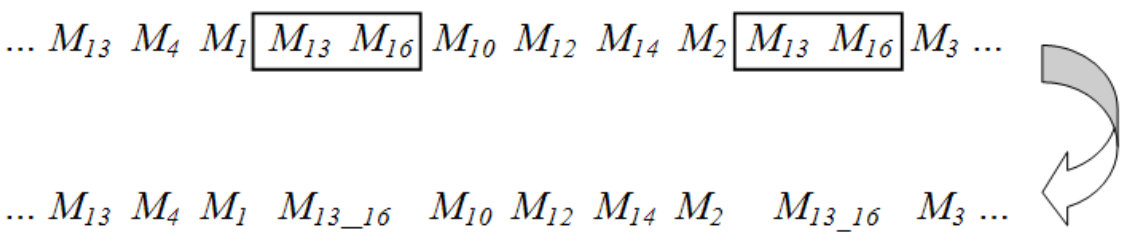


Figure 4.3: A token sequence example. The objective of the model selection criterion is to select two tokens to merge and create a longer model.

to select the most suitable adjacent token pair $(M_j; M_k)$ for merging. Since the token M_j can be followed by all the tokens and M_k can also be preceded by all the tokens as shown in Figure 4.4, a good candidate pair $(M_j; M_k)$ for merging over the entire corpus should satisfy the following conditions: 1) the probability of M_j followed by M_k, i.e. $p(\omega_t = M_k | \omega_{t-1} = M_j)$, should be high; 2) the probability of M_k preceded by M_j, i.e. $p(\omega_{t-1} = M_j | \omega_t = M_k)$, should be high; 3) The token pair $(M_j; M_k)$ should occur frequently in the corpus as the same speech pattern such as a syllable can be found in different words. The conditional probabilities are computed as follows:

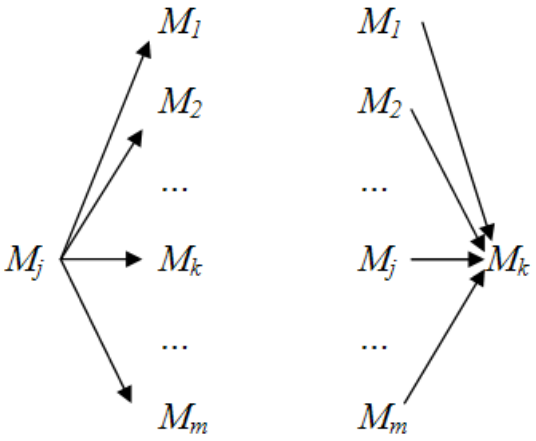


Figure 4.4: All combinations of tokens. Token M_j can be followed by any token and token M_k can also be preceded by any token.

$$\begin{aligned} p(\omega_t = M_k | \omega_{t-1} = M_j) &= \frac{p(\omega_{t-1} = M_j, \omega_t = M_k)}{p(\omega_{t-1} = M_j)} \\ &= \frac{\text{Count}(\omega_{t-1} = M_j, \omega_t = M_k)}{\sum_{l=1}^{m} \text{Count}(\omega_{t-1} = M_j, \omega_t = M_l)} \end{aligned} \tag{4.3}$$

$$\begin{aligned} p(\omega_{t-1} = M_j | \omega_t = M_k) &= \frac{p(\omega_{t-1} = M_j, \omega_t = M_k)}{p(\omega_t = M_k)} \\ &= \frac{\text{Count}(\omega_{t-1} = M_j, \omega_t = M_k)}{\sum_{l=1}^{m} \text{Count}(\omega_{t-1} = M_l, \omega_t = M_k)} \end{aligned} \tag{4.4}$$

where $p(\omega_{t-1} = M_j, \omega_t = M_k)$ represents the joint probability of M_j and M_k and $\text{Count}(\omega_{t-1} = M_j, \omega_t = M_k)$ denotes the number of times M_k is followed by M_j in the entire corpus.

The above mentioned model selection problem is actually similar to the word boundary detection problem in text processing. Given a sequence of symbols, e.g. a sequence of English characters without space, a sequence of Chinese characters without punctuation and a sequence of syllables, the task of textual word boundary detection is to identify the words in the sequence by segmenting the sequence. The input of textual word boundary detection is usually perfect text [143–146] but the problem of this work deals with token sequences which contain noise. The following paragraphs discuss some methods of textual word boundary detection.

To detect textual word boundaries, the probability equations Eq. 4.3 and Eq. 4.4 are examined, namely, forward transitional probability (TP) [143] and backward TP [144] respectively. In [143], researchers proposed to detect word boundaries in a non-space symbolic sequence (e.g. a sequence of English syllables without space, and a sequence of Chinese characters without punctuation) using forward TP. The researchers [143] believe frequently occurred subsequences such as bigrams (i.e. two adjacent symbols) are within words, and subsequences do not belong to a word tend to occur with low frequency. Thus, word boundaries can be identified by unlikely transitions between two successive symbols hence to segment continuous text into words as illustrated in Figure 4.5.

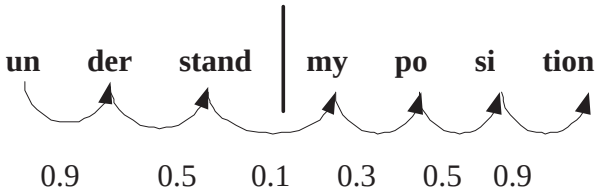


Figure 4.5: An illustration of the use of forward TP in word segmentation [147]. The given symbolic string is a sequence of English syllables and the floating numbers indicate the bigram forward TP between the neighboring syllables. The transition of local minimum value is considered as a word boundary.

In an alternative method, researchers in [144] examined to assess the relations within the bigram pair $(M_j; M_k)$ using backward relationship, i.e. the backward TP as shown in Eq. 4.4, which is the probability that M_k was preceded by M_j. The values of forward and backward TP can differ as they may imply different information. For example, in a sequence of non-space English letters, the forward TP between letter "q" and "u" is very high (near to 1.0) which implies that "q" is followed by "u" frequently; while backward TP is much lower which implies that "u" can be preceded by many other letters. The psychological study in [144] was also carried out by conducting two groups of human listeners to detect word boundaries in an artificial unknown language which the pronunciations of words were synthesized. The synthesized corpus was then played in forward and backward orders to the two groups respectively, and the experimental results shown that the performance of identifying word boundaries by giving only backward information was slightly better than providing only forward information.

The forward and backward TP are asymmetric as the predication is conditional on either predecessor or successor [148]. In another method, a symmetric measure of the adjacent pairs is mutual information (MI), which has also been applied to detect word boundaries [145, 146] in a given sequence of symbols. The mutual information, $I(M_j; M_k)$, of two symbols is defined as

$$I(M_j; M_k) = \log_2 \frac{p(\omega_{t-1} = M_j, \omega_t = M_k)}{p(\omega_{t-1} = M_j)p(\omega_t = M_k)}, \tag{4.5}$$

and it is a quantity that measures the mutual dependence of the two variables. Hence, a larger value of $I(M_j; M_k)$ indicates a higher possibility of that M_j and M_k occur within the same word. By setting a threshold, the given sequence can then be segmented by inserting a boundary between the bigram pair of which the MI value is below the threshold. The study in [148] has shown that MI outperformed TP in detecting word boundaries in textual English syllable sequence.

The above reviewed TP and MI methods can identify the unlikely/likely transition between an adjacent pair in textual word segmentation, however the model merging criterion in this work does not only rely on the relationship between the adjacent tokens but also the frequency of their occurrence. For example, if an adjacent token pair only occurs once in the corpus and each individual token does not occur anymore, the two tokens will be selected by the above mentioned word segmentation methods to merge as the transition between them is the most likely (e.g. the forward TP is 1.0). However, the pair of tokens do not satisfy the condition that they occur frequently so that they should not be selected to merge in this work.

Therefore, the model selection criterion should consist of two factors: relationship factor that measures the dependence of two adjacent tokens and frequency factor that measures the frequency of the token pair's occurrence. In this work, the relationship factor is chosen to be presented by MI as discussed above. The frequency factor is then modeled by the joint probability of the adjacent pair, i.e. $p(\omega_{t-1} = M_j, \omega_t = M_k)$, which indicates the frequency of the occurrence of $(M_j; M_k)$. To combine with MI, frequency factor in log scale is included in Eq. 4.6. As the proposed selection criterion is based on

MI, it is called enhanced mutual information and denoted as EI as shown in Eq. 4.6.

$$\begin{aligned} EI(M_j; M_k) &= \log_2 \frac{p(\omega_{t-1} = M_j, \omega_t = M_k)}{p(\omega_{t-1} = M_j)p(\omega_t = M_k)} + \log_2 p(\omega_{t-1} = M_j, \omega_t = M_k) \quad (4.6) \\ &= \log_2 \frac{p(\omega_{t-1} = M_j, \omega_t = M_k)^2}{p(\omega_{t-1} = M_j)p(\omega_t = M_k)} \\ &= \log_2 \{p(\omega_t = M_k | \omega_{t-1} = M_j) p(\omega_{t-1} = M_j | \omega_t = M_k)\} \quad (4.7) \end{aligned}$$

where $p(M_j, M_k)$ represents the joint probability of M_j and M_k, and $EI(M_j; M_k)$ evaluates enhanced mutual dependence of M_j and M_k. In Eq. 4.7, $EI(M_j; M_k)$ can be re-formulated using the product of forward and backward TP. The idea of the equation can be interpreted as detecting bigram pairs from both forward and backward directions as illustrated in Figure 4.6.

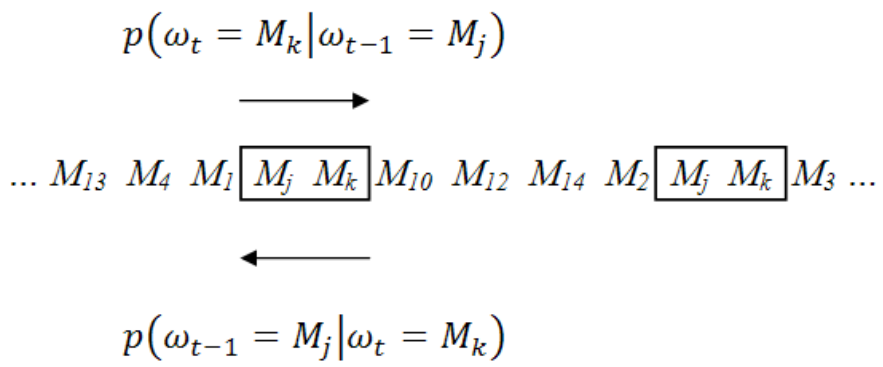


Figure 4.6: Symmetric bigram counting is used in model selection.

To choose the most suitable models to be merged, a greedy strategy is used and the token pair $(M_j; M_k)$ with the highest EI value is selected as shown in Eq. 4.8.

$$(\hat{M}_j; \hat{M}_k) = \arg\max EI(M_j; M_k) \quad (4.8)$$

The selected models are then catenated to form a new HMM and the newly created model is added to the set of HMMs $\mathbf{\Gamma}$ for the subsequent process.

4.2.2.1 Iterative model merging process

By using the above proposed model selection criterion, a framework of iterative model merging process illustrated in Figure 4.7 is applied to discover speech patterns. The

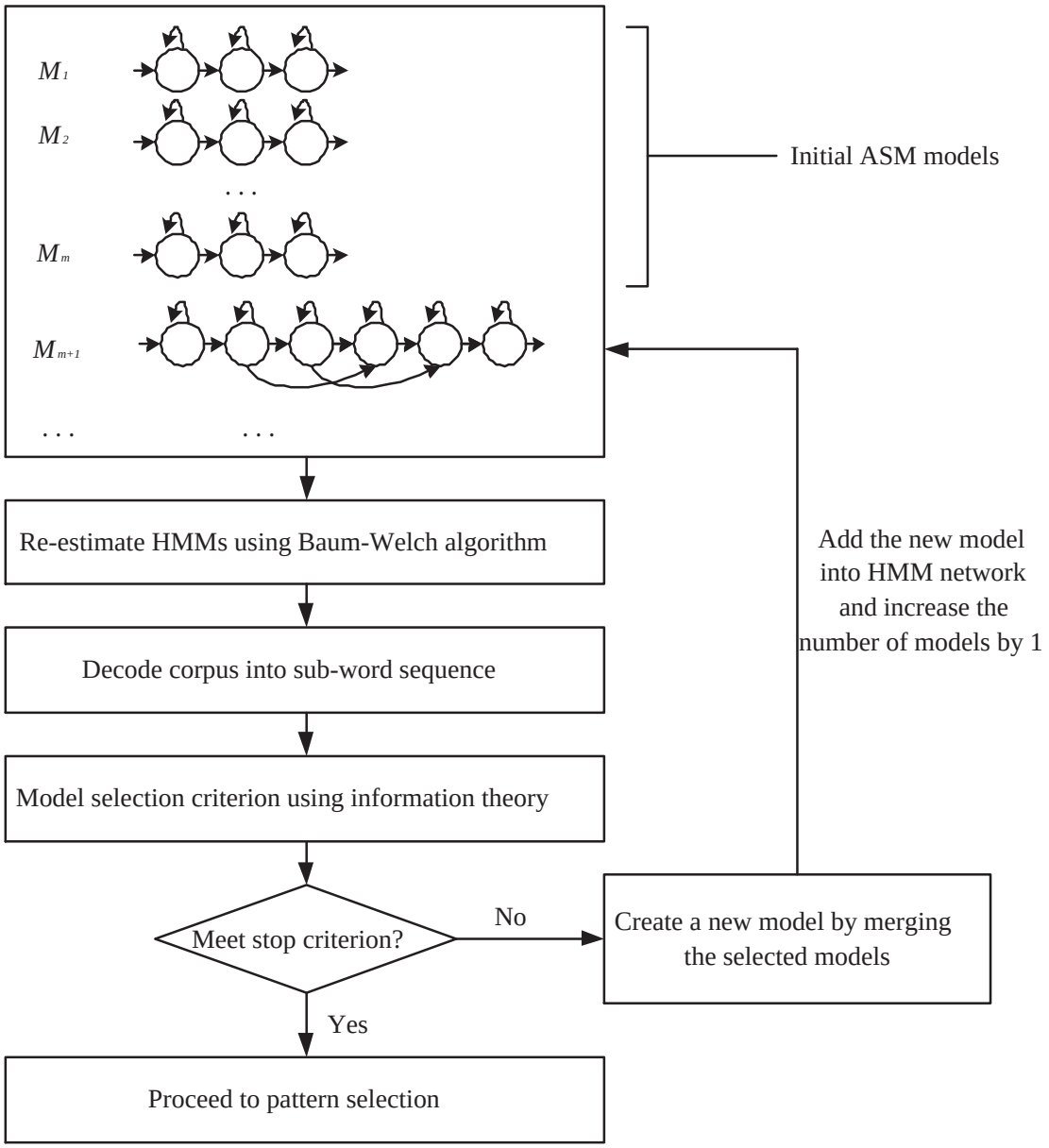


Figure 4.7: A diagram of the iterative approach to model merging.

framework first transcribes the utterances into token sequence using the initial set of sub-word HMMs $\mathbf{\Gamma}$, and the model selection criterion is applied to select two HMMs $\hat{M}_j$ and $\hat{M}_k$ of the highest EI score. The two models are then concatenated to produce a new model M_{m+t} as illustrated in Figure 4.8 where t denotes the t^{th} iteration in the model merging process, in the following ways:

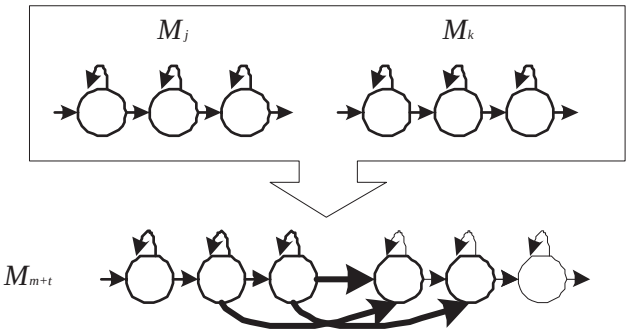


Figure 4.8: An illustration of model merging. The three arrows in bold represent the newly added transition probabilities.

(a) The total number of states, ns_{m+t}, of model M_{m+t} is the number of states of M_j and M_k, i.e.

$$ns_{m+t} = ns_j + ns_k, \tag{4.9}$$

where ns_j and ns_k denote the numbers of states of models M_j and M_k respectively.

(b) In M_{m+t}, the first ns_j states are identical to the states of M_j, and the subsequent ns_k states are identical to the states of M_k.

(c) The following new transition probabilities are added to merge the models for robustness: $a^{m+t}_{s_{ns_j},s_{ns_j+1}}$, $a^{m+t}_{s_{ns_j-1},s_{ns_j+1}}$, $a^{m+t}_{s_{ns_j},s_{ns_j+2}}$, and they are initialized as:

$$a^{m+t}_{s_{ns_j},s_{ns_j+1}} = a^{m+t}_{s_{ns_j},s_{ns_j+2}} = a^{m+t}_{s_{ns_j},s_{ns_j}} = 1/3, \tag{4.10}$$

$$a^{m+t}_{s_{ns_j-1},s_{ns_j+1}} = a^{m+t}_{s_{ns_j-1},s_{ns_j}} = a^{m+t}_{s_{ns_j-1},s_{ns_j-1}} = 1/3. \tag{4.11}$$

The rest of the transition probabilities are retained from the previous models.

(d) Lastly, the newly created model M_{m+t} is added to the HMM set $\mathbf{\Gamma}$:

$$\mathbf{\Gamma} = \mathbf{\Gamma} \cup \{M_{m+t}\} \tag{4.12}$$

After the two models are merged, the label of the utterances are updated to reflect this change. Specifically, all the occurrences of M_j followed by M_k in the label are replaced by their merged model M_{m+t}. For occurrences of M_j not followed by M_k and occurrences of M_k not preceded by M_j, their labels are unchanged. With this new label, the parameters of all HMMs in $\mathbf{\Gamma}$, including the original models and the newly merged model, are re-estimated. After the new model set converges, the entire corpus is decoded by the new model set again to generate a new version of token sequences. The model merging process is then repeated based on the new decoding output. This merge and retrain process is iterated until a stopping criterion is met.

In the framework, the model merging process can stop when $EI(\hat{M}_j; \hat{M}_k) < \gamma$ where γ is a user defined threshold. A higher γ will cause the merging process to stop earlier, and hence may prevent a speech pattern of interest (e.g. words and short phrases) from being formed. On the other hand, a lower γ will allow the merging process to continue for more iterations, and therefore over-merge the patterns. As a result, some long but meaningless patterns may be produced. For example, a long pattern can be a whole word followed by a part of another word.

After the merging process, not every model in the model set represents a pattern of interest. For example, sub-word units are unlikely to be the patterns of interest. Thus, a pattern selection module is introduced to identify the patterns of interest in the next section.

4.2.3 Pattern Selection

The resultant models found by the iterative model merging process is a set of HMMs which represent different acoustic patterns such as sub-word units, syllables, words/short phrases and silence segments. As the objective of this work is to discover speech patterns such as words and short phrases, a further refinement process is to identify the patterns of interest.

Before describing the details of pattern selection criteria, differences between patterns of interest and undesired patterns are clarified here. The patterns of interest should satisfy two conditions: (i) high frequency in the corpus, and (ii) stable models, i.e. the pattern model should have low chance to be further merged with other patterns.

Condition (i) can be simply evaluated by counting the number of occurrences of a pattern in the decoding output. The condition (ii) cannot be measured as we do not have prior knowledge about the corpus. However, it can be achieved by investigating the stability of the number of occurrences of a pattern. Once a pattern of interest such as a complete word is produced, it will not be easily merged with other patterns according to the proposed model merging criterion. Hence, the variance of the number of occurrences in each training iteration can be used to evaluate condition (ii) in this work.

To implement the above ideas, the following symbols are first defined: $n(M_i, t)$ denotes the number of occurrences of model M_i in the decoding output at the t^{th} iteration; A score $\sigma^2(M_i, T)$ is defined for each model M_i to indicate the stability of M_i, as shown in the following equation:

$$\sigma^2(M_i, T) = \begin{cases} \frac{1}{T+1} \sum_{t=0}^{T} \left(n(M_i, t) - \frac{1}{T+1} \sum_{t=0}^{T} n(M_i, t) \right)^2 & (i \leq m) \\ \\ \frac{1}{T+m-i+1} \sum_{t=i-m}^{T} \left(n(M_i, t) - \frac{1}{T+m-i+1} \sum_{t=i-m}^{T} n(M_i, t) \right)^2 & (i > m) \end{cases} \tag{4.13}$$

When $t = 0$, $n(M_i, 0)$ indicates the number of occurrences of M_i in decoding output before the 1st iteration.

The score $\sigma^2(M_i, T)$ indicates the stability of M_i: A small value of $\sigma^2(M_i, T)$ means that M_i does not or rarely has been merged by any other models during the iterative process hence it is considered as a stable pattern in this work; A large value of $\sigma^2(M_i, T)$ indicates M_i is frequently selected to be merged with other models hence it has a high chance to be a sub-word unit.

The pattern selection process that summarized in Figure 4.9 consists of three steps:

Step 1) Silence segments remover: Silence segments occur frequently in speech corpus thus $\mathbf{\Gamma}$ can contain HMMs which model silence segments. A simple VAD filter based on short-term energy can be used to remove the silence segments.

Step 2) Less frequent pattern remover: From the above discussion, the patterns of interest should occur frequently. To remove the less frequent patterns, a user-defined threshold η is used to remove the patterns of which $n(M_i, T) < \eta$, where $n(M_i, T)$ indicates the number of occurrences of M_i at iteration T of model merging.

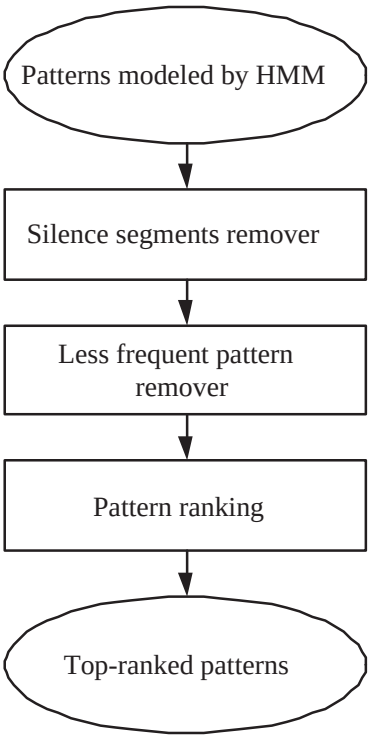


Figure 4.9: Procedures used in patten selection module.

Step 3) Pattern ranking: The scores $\sigma^2(M_i, T)$ are sorted in ascending order and the first number of N ranked models are selected as the final patterns.

By using the above pattern selection process, N patterns are selected as patterns of interest to construct a lexicon for the untranscribed speech corpus.

4.3 Experiments

4.3.1 Corpus

Since there are no benchmarking corpora for automatic speech pattern discovery evaluation, many studies were carried out on ad-hoc corpora [13, 60, 119, 122, 133]. In addition, most of these corpora were recorded by a single speaker. This work aims to discover speech patterns in multi-speaker environment. Therefore, the task of discovery of English words on TIDIGITS corpus is studied in this section.

The TIDIGITS corpus is a publicly available speech corpus used for developing speech recognition technology. The TIDIGITS corpus has a small vocabulary size that consists

of only 11 English words: digits 0–9 and "oh" (an alternative pronunciation of digit '0'). There are 8440 clean utterances spoken by male and female speakers. Each utterance contains either an isolated word or several connected words, and the average length of the utterances is about 2sec. There are two reasons why TIDIGITS corpus was selected for this experiment: i) within the 8440 utterances, each digit occurs about 2520 times on average hence there are sufficient number of repetitions to evaluate the performance of the proposed technique using statistical measurement such as F1 score; ii) the utterances are spoken by multiple speakers so that it can evaluate the robustness of the proposed technique against speaker variations.

4.3.2 Experiment Setup

In the experiments, the corpus was partitioned into gender groups for speech pattern discovery. As speakers' and channel differences are captured by automatic acoustic modeling techniques before different acoustic units are distinguished [122, 149], the existing work either discovered speech patterns within single speaker corpus [13, 122] or multi-speaker corpus of the same gender [106]. This work also separated male utterances from female so that the proposed technique was applied on male and female utterances separately. The utterances in each gender group were further divided into development and evaluation sets. The development sets were used for parameter tuning and the best parameter values are used in the evaluation set. Both the development and evaluation sets of each gender contained 2110 utterances.

The features used in the experiments are raw MFCC features, including the first 13 MFCC features with its delta and double delta versions. The frame size in feature extraction is 25ms and hop size 10ms.

4.3.3 Evaluation Criteria

The objective of the proposed technique is to discover speech patterns of interest in an untranscribed corpus. In this experiment, the patterns of interest are the 11 digits in TIDIGITS corpus. Hence, the proposed technique is evaluated based on whether the discovered patterns match the 11 digits.

Given the discovered patterns, it is necessary to first find the correspondence between the discovered patterns and the reference digits. This is achieved by comparing the detected pattern sequence and the reference digit sequence as illustrated in Figure 4.10. For each utterance, the reference digit sequence was obtained by force-aligning the utterance to its reference transcription using a word-based speech recognition system. The detected pattern sequence is simply the output of decoding the utterance using the finalized models generated by the proposed technique. As the segments in the two sequences usually do not align to each other exactly, an error interval of ±19 frames (i.e. 190ms) was allowed. If the differences between the boundaries of a pattern segment and a reference digit segment are less than 19 frames, the pattern segment is considered as matching the digit segment. As a discovered pattern has many instances in the corpus and they may be matched to different digits, the "identity" of the pattern is determined as the digit to which the instances of the pattern are matched most frequently.

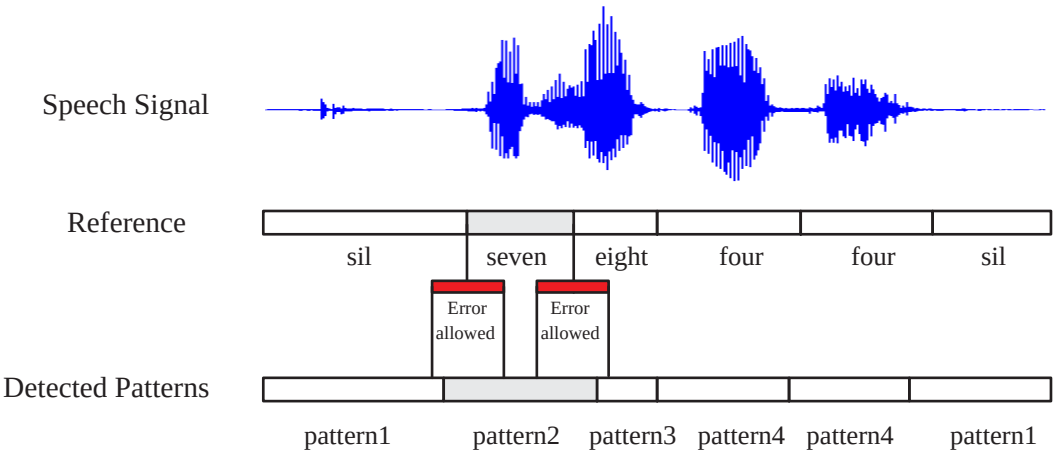


Figure 4.10: An illustration of matching discovered patterns to reference digits. The detected pattern is matched a digit if both its ends fall in the allowed error interval.

After identifying the correspondence between each discovered pattern and the reference digits, evaluation metrics such as recall, precision and F1 score are calculated for each pattern. The recall of the i^{th} discovered pattern is computed as

$$\text{Recall}(i) = \frac{\text{No. of matches between pattern } i \text{ and digit } d(i)}{\text{Total No. of instances of digit } d(i)} \tag{4.14}$$

where $d(i)$ denotes its corresponding digit. The recall indicates whether pattern i matches all the instances of digit $d(i)$. The precision of the i^{th} discovered pattern is computed as

$$\text{Precision}(i) = \frac{\text{No. of matches between pattern } i \text{ and digit } d(i)}{\text{Total No. of instances of pattern } i} \tag{4.15}$$

and precision indicates the purity of the instances of a pattern. The F1 score of pattern i is the harmonic mean of precision and recall:

$$\text{F1}(i) = \frac{2 \times \text{Recall}(i) \times \text{Precision}(i)}{\text{Recall}(i) + \text{Precision}(i)} \tag{4.16}$$

4.3.4 Evaluation Results and Discussion

4.3.4.1 Generation of sub-word models

In this work, the number of initial ASM models, m, should be predefined before applying the proposed technique. If the value of m is larger than the actual number of phonemes in the corpus, the same phoneme may be modeled by more than 1 ASM model so the same pattern of interest will eventually modeled by more than 1 merged models; If the value of m is smaller than the actual number of phonemes, different phonemes may be modeled by the same ASM model. To select a proper m, different values of 8, 16 and 32 were used to generate 3 groups of ASM models respectively. To evaluate the performance of sub-word modeling, sample speech segments modeled by each ASM model were randomly selected and verified manually. From this process, the following observations were obtained:

- $m = 8$: About 88% of the ASM models were found to represent more than 1 phoneme.
- $m = 16$: About 69% of the models were found to represent a single phoneme and the rest represented more than 1 phoneme.
- $m = 32$: Different ASM models were found to represent the same phoneme.

From the above observations, the value of m was eventually set to be 16 as most of the ASM models represent a single phoneme. A further analysis is given in the following paragraphs on the ASM models which represent more than 1 phoneme.

4.3.4.2 Parameters tuning on development sets

In the proposed technique, several parameters need to be tuned, including i) the user defined threshold γ in stopping criterion; ii) the parameter η used to remove less frequently occurring patterns; iii) the parameter N which is the number of selected patterns; and iv) the word insertion penalty in corpus decoding, which is used to add additional value to each model when it transits from end of one word to the start of the next [118]. The proposed technique is applied on the development sets to tune the parameters.

The parameter γ affects the length of the discovered patterns. If γ is too low, the model merging process produces long patterns which concatenated 6 or 7 sub-word models. Such patterns represents more than one digit words. If γ is too high, the iterative model merging process stops early and only short patterns can be formed. Based on initial study on the development sets, γ was empirically set to $\log_2(0.15)$ for all the subsequent experiments.

The parameter η is used to remove the less frequently occurring patterns. As the vocabulary size of TIDIGITS corpus is small and every digit occurs hundreds of times, the parameter η is not sensitive in this case and set to 50. This causes all patterns with less than 50 instances being removed from the pattern list.

To decide the parameter N, the number of patterns to be selected, a series of experiments were carried out on the development sets. Table 4.1 shows an example of mapping between the resulting patterns and the reference words of the female development set. The patterns are ranked using the pattern selection process described in Section 4.2.3, and the table also displays the number of occurrences, recall, precision and F1 score of each individual pattern. In Table 4.1, the top 10 ranked patterns can be mapped to 10 distinct digit words, however, the 11th pattern is mapped to word "oh" which is the same as the 8th pattern. The similar observations were also made from other experiments. Such finding implies that the proposed technique is able to discover 10 of the 11 words of TIDIGITS lexicon, hence the number of retained patterns N was set to be 10 in the subsequent experiments.

The F1 scores of the top 10 patterns were chosen to decide the word insertion penalties used in corpus decoding for female and male data separately. Figure 4.11 shows the plots of the average F1 score of the top 10 selected patterns according to different word insertion

Table 4.1: Examples of mapping model sequence to words.

Rank	Merged sub-word models	References	No. Occur.	Recall	Precision	F1 score
1	M_{12} M_{11} M_{12}	six	630	1.00	1.00	1.00
2	M_{12} M_{11} M_{13}	two	278	0.46	1.00	0.63
3	M_{12} M_{8} M_{13} M_{3}	seven	506	0.79	0.99	0.88
4	M_{15} M_{1}	three	589	0.93	0.99	0.96
5	M_{9} M_{3}	one	994	0.90	0.60	0.72
6	M_{16} M_{8}	five	606	0.92	0.99	0.95
7	M_{2} M_{11} M_{15}	zero	472	0.63	0.86	0.73
8	M_{7}	oh	610	0.66	0.68	0.67
9	M_{4} M_{15}	four	570	0.90	0.99	0.94
10	M_{8} M_{1}	eight	442	0.72	0.99	0.83
11	M_{10}	oh	606	0.31	0.31	0.31
12	M_{3}	—*	312	—	—	—
13	M_{2} M_{11}	zero	154	0.12	0.51	0.20
14	M_{12} M_{11}	two	326	0.52	0.97	0.68
15	M_{9}	nine	250	0.28	0.68	0.39
16	M_{8} M_{13}	seven	124	0.13	0.66	0.22
17	M_{13}	—	166	—	—	—
18	M_{2}	—	170	—	—	—
19	M_{1}	eight	172	0.23	0.83	0.36
20	M_{4}	four	139	0.09	0.41	0.15
21	M_{16}	five	105	0.06	0.34	0.10

Note: * The "—" indicates the model sequence does not match any reference word.

penalties. The best average F1 score occurred at word insertion penalty of -38 and -46 for female and male data, respectively.

4.3.4.3 Further analysis of sub-word units

It is interesting to analyze the obtained patterns in Table 4.1. For instance, sub-word models M_9 and M_3 were found to be merged into a long HMM which represents word "one", and sub-word models M_{12}, M_8, M_{13} and M_3 were merged to represent word "seven". An observation is that model "M_3" occurs at the end of both patterns. As the words "one" and "seven" have the same last phoneme, i.e. /n/, the phonetic equivalent of the original ASM model "M_3" is hypothesized to be /n/. The hypothesis has been verified by aligning the HMM sequences with the reference phoneme boundaries. Similarly, the ASM model "M_{12}" is found to represent the fricative phonemes such as /s/.

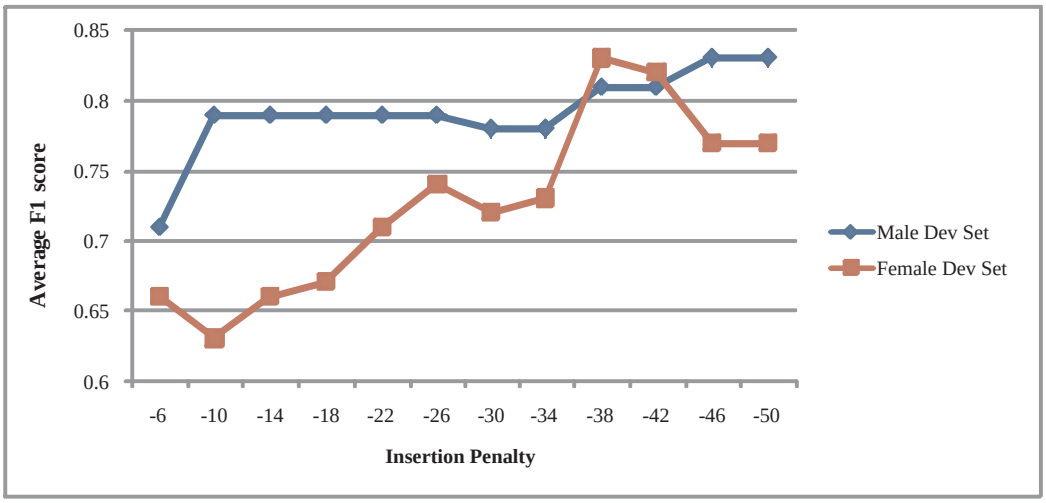


Figure 4.11: Parameter tuning of insertion penalty. The plots show the average F1 score of the top 10 selected patterns. The error interval of ±19 frames was used in evaluation.

4.3.4.4 Evaluation results

With the parameters obtained from the development sets, the proposed technique was applied to the evaluation sets. The average F1 scores of the top 10 patterns are also shown in Table 4.2. The results in Table 4.2 show that the F1 scores produced by the

Table 4.2: Average F1 scores on development and evaluation sets.

	Dev. Set	Eval. Set
Male	0.83	0.80
Female	0.83	0.82

proposed technique are always larger than 0.80. This shows that the proposed method is able to discover most of the digits in the TIDIGITS reliably. In addition, the similar performance on the development and evaluation sets shows that the parameter tuning process discussed in the previous section works well.

4.3.4.5 Discussion on error interval

In the above experiments, the error interval used in the evaluation was set to ±19 frames (i.e. 190ms). As the error interval is an important factor which affects the F1 scores of the patterns, different error intervals from ±5 frames to ±19 frames were used to measure

the F1 scores of the patterns. The experiment was carried out on female development set, and Figure 4.12 shows the average F1 score of the top 10 selected patterns by setting different word insertion penalties and different error intervals. In Figure 4.12, it is obvious that the plots of error intervals ±13 frames to ±19 frames are quite close, and the plots of error intervals ±5 frames to ±9 frames are far from the rest. Such findings imply that about 70% of the discovered speech patterns' boundaries are within ±5 frames interval compared to the reference word boundaries, and about 80% and 87% of the discovered patterns' boundaries are within ±7 frames and ±9 frames intervals respectively.

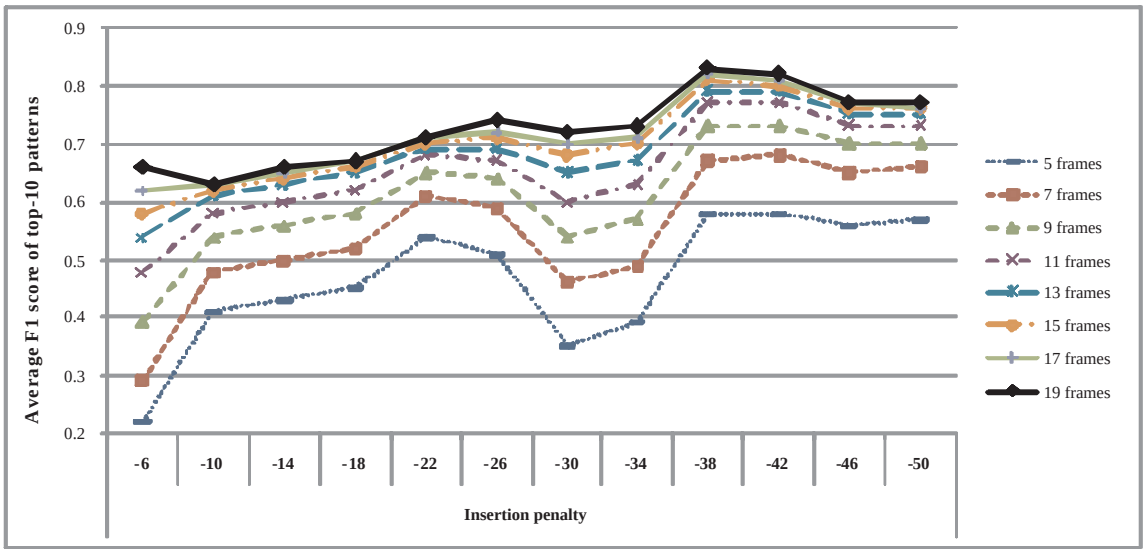


Figure 4.12: Performance of selected patterns measured using different error intervals. The plots show the average F1 scores of top 10 patterns created by setting different word insertion penalties. The error interval of ±19 frames is eventually used in the subsequent evaluation.

4.3.4.6 Discussion on speaker coverage

As the proposed technique aims to discover speech patterns of interest in multi-speaker corpus, it is interesting to investigate whether the discovered patterns are spoken by multi-speakers or not. Hence, the speaker coverage of the top 10 ranked patterns has been computed on the female development set. A speaker is considered as being covered by a pattern if at least 1 instance of the pattern is spoken by the speaker. There are 55 distinct female speakers in the set. The coverage of the top 10 ranked patterns is shown

in Table 4.3. From the table, it is observed that 6 out of 10 patterns occur in all the speakers, and even the pattern with the least coverage has 48 out of 55 (87.3%) speakers. The breakdown of the maximum, minimum and average number of instances found for each speaker is also listed in Table 4.3. The above experimental results show that the proposed pattern discovery technique is effective for multi-speaker corpus.

Table 4.3: Speaker coverage of each top ranked pattern.

Rank	Merged sub-word models	Reference Digit	No. speakers covered/ Total No. speakers	No. occur. for each speaker		
				Max.	Min.	Avg.
1	M_{12} M_{11} M_{12}	six	55/55	18	7	11.5
2	M_{12} M_{11} M_{13}	two	52/55	16	1	5.3
3	M_{12} M_8 M_{13} M_3	seven	53/55	15	1	9.5
4	M_{15} M_1	three	55/55	17	5	10.7
5	M_9 M_3	one	55/55	29	8	18.1
6	M_{16} M_8	five	55/55	16	4	11.0
7	M_2 M_{11} M_{15}	zero	48/55	16	1	9.8
8	M_7	oh	55/55	26	1	11.1
9	M_4 M_{15}	four	55/55	16	4	10.4
10	M_8 M_1	eight	54/55	17	1	8.2

It is difficult to compare the effectiveness of the proposed technique to other work due to the realization of experiments using different corpora. To provide a qualitative comparison, the results reported by Park and Glass [13] are reviewed. In [13], the segmental DTW is proposed to detect repeating speech patterns in single speaker lecture recordings. The experimental results show that the segmental DTW is able to detect on average 66 keywords/short phrases with 89% average purity (i.e. precision) from each of the six lecture recordings. In this work, the proposed technique is able to detect 10 out of 11 digits from TIDIGITS corpus with average precision 90%. Although the task in [13] is different from this task and the results cannot be compared directly, both techniquies achieve high precision. In addition, it should be noted that the study in [13] is performed on single-speaker corpus, while this work can be applied on multi-speaker speech.

4.4 Conclusion

In this chapter, a novel technique is proposed to automatically discover recurrent patterns such as words or phrases from transcription-free multi-speaker speech corpus. The proposed technique adopts a bottom-up strategy to grow sub-word unit models towards longer models to represent repeating patterns. A greedy criterion is used to guide the discovering process in an efficient way. As the technique is based on acoustic modeling, it is more robust in multi-speaker environment. The proposed approach has been applied to discover digital words from TIDIGITS corpus and was able to detect 10 of 11 words in the corpus. This work can be further applied to identify keywords from untranscribed speech or generate lexicon for unknown languages.

In chapters 3 and 4, different techniques have been proposed to address the problem of automatic audio pattern discovery in both music and speech. The algorithms discussed identification of the frequently occurring audio information such as chorus portions in pop songs and keywords in speech automatically. However less frequently occurring information cannot be identified by these algorithms. In addition, if users desire to retrieve audio information on a specific content, such technology is unable to fulfill the requirement. Technology related to query-by-example is highly desired in audio retrieval applications such as Google voice search [21] and spoken document retrieval systems [26]. This motivates me to research on efficient query-by-example techniques in the following chapter.

Chapter 5

Mixed Music and Speech Pattern Retrieval in Broadcast Audio

This chapter focuses on the problem of searching for audio patterns in broadcast audio database that contains both music and speech. In such database, music and speech often occur at the same time. For example, TV and radio broadcast programs contain several types of audio contents such as speech-only, music-only and speech with music background. To process such data, a joint modeling approach is proposed to model the two types of audio simultaneously to enable the search of patterns containing both music and speech. The approach has been applied to the retrieval task of commercials in broadcast audio data.

Searching for mixed music and speech patterns in a broadcast audio database is an information retrieval problem, where the mixed music and speech patterns are the queries. It remains a difficult problem due to several reasons [31]. Firstly, conventional keywords spotting (KWS) techniques [61, 150–152] are not able to be applied on audio with music as they work only on words, phonemes or phonetic lattice generated by speech recognizers, which do not model music data well. Secondly, although template matching techniques [20, 22, 62, 130] are able to search speech and music simultaneously, they are not computationally efficient and vulnerable to variations in audio for the reasons discussed in Section 5.1.

In this chapter, a robust modeling technique is proposed to represent both speech and music for content-based indexing and retrieval. The proposed technique models speech and music signals by conventional phoneme HMM models and ASM models, respectively.

The phoneme HMM models and the ASM models are then combined and used to decode the queries and audio corpus. Based on the decoded output, a vector-based statistical information retrieval method is then used for indexing and retrieval of the commercials. The proposed technique has been applied to search for 100 commercial queries in the 28 hour audio corpus extracted from TRECVID [34] database. The experimental results show that the proposed technique outperforms the conventional speech-only phoneme model and it is also robust and effective to process broadcast audio. The following sections are organized as follows: Section 5.1 briefly reviews related techniques. The proposed technique is presented in Section 5.2. Section 5.3 discusses the experimental results and the conclusion is given in Section 5.4.

5.1 Introduction

Audio pattern retrieval is the task to locate a desired audio pattern in a collection of audio archives. To retrieve an audio pattern, different applications have been developed [20]. For example, query-by-text systems [26–28] return audio archives which contain a given text query; Query-by-semantics systems [10, 29, 30] return audio clips which belongs to a given semantic class such as sound effects (bell, car braking, explosion, etc.) and audio content types (speech, music, background noise, etc.); Query-by-humming systems detect the music/song pieces which contain similar tune hummed by users [43, 44]; Query-by-example systems [22–25, 153] search for an audio segment by the given query example. In this thesis, the audio pattern retrieval techniques are specifically examined for query-by-example applications of mixed music and speech audio content.

The broadcast audio data content usually consists of both music and speech. For example, in most TV commercials, speech is accompanied by background music. As such data is growing exponentially on the Internet, it is necessary to have an effective way to organize and retrieve mixed music and speech patterns. The following paragraphs review some of the existing audio retrieval techniques.

Conventional information retrieval research has mainly focused on textual data such as Google textual queries. Given a textual query that contains several keywords, the text-based documents which contain all or a partial set of keywords are retrieved. Documents

found with greater number of keywords occurrence will be ranked higher in the retrieved list.

Different to text retrieval, audio queries cannot be directly compared to the audio corpus due to the lack of comparable entities such as characters and words [154]. Hence, there is a need to first convert audio data into comparable entities for indexing. For example, automatic speech recognition ASR systems have been used to transcribe spoken queries and spoken documents into word-level text and then information retrieval techniques are applied to retrieve the desired spoken documents [26, 27]. The ASR systems are based on HMMs which are pre-trained using transcribed speech data. If the query is also a speech segment, the query is first recognized into text form for the retrieval process. The text-based information retrieval techniques can then be applied to retrieve the spoken documents. However, ASR-based approach faces several limitations such as the inability to recognize OOV words and the poor recognition performance when there is acoustic mismatch between the ASR training data and the query data [140]. Besides these limitations, ASR approach is also not suitable for mixed speech and music patterns retrieval as music segments cannot be recognized by ASR systems.

Due to the above limitations in transcribing audio data into comparable entities for retrieval, researchers have proposed alternative techniques to search for desired audio patterns. The following paragraphs briefly review 3 categories of these techniques: audio feature vector comparison, audio template matching, and subword/lattice comparison.

Audio features vectors can be extracted directly from audio signals to index audio corpus for retrieval [22, 44, 155, 156]. As audio signals can be represented in the time domain or the frequency domain, different features can be derived or extracted from these two representations. Audio features such as ZCR, loudness, brightness, bandwidth, etc have been examined for content-based characterization of audio signals, and they have been described in Chapter 2 and reviewed in [157]. Each feature has its own discriminative power of usefulness for audio classification, indexing and retrieval. For example, spectral features extracted from frequency domains were used for music genres classification and retrieval problem [155]; Scheirer and Slaney used 13 audio features including spectral centroid and ZCR to discriminate speech from music [156]. Recently, the Spectral Flatness Measure (SFM) [22] was generated from power density spectrum (PDS) of audio

signal. The PDS describes the total average power that distributed over frequency of the audio signal. The SFM measures the flatness of the PDS over a short audio segment. The audio query can be matched to the database by comparing the sequences of SFM values. Although the audio feature vectors can be directly extracted from audio signal at low computational cost, they are easily distorted by noise hence their application is limited.

In the query-by-humming application [44], Lu *et al.* proposed to represent the music melody by combining pitch contour, pitch interval and rhythm (i.e. duration) of the notes in a music piece. The pitch contour is a symbolic feature that indicates the condition that a music note is higher or lower than the previous note. The melody representations are extracted from both song database and a hummed query. A hierarchical matching process is then used to first align the query's pitch contour to the candidate segments in the database using DP. The candidate segments are retained for subsequent comparison if their alignment distances are higher than a threshold. Similarity of pitch interval and rhythm is finally computed between the pairs of note sequence according to the matched paths for ranking. Their experimental results shown that the correct song given a 10 seconds query can be retrieved in the top-10 results among 1000 song collection with accuracy 88%. Such query-by-humming technique however is not applicable for the mixed content broadcast audio as pitch change in speech does not change the meaning of spoken words.

In another approach, template matching approach has been examined to avoid the shortcomings of ASR systems. The conventional DTW [130, 158, 159] can be used to detect an audio query in audio database by identifying the best alignment between the query and a segment of audio data. Such feature matching approach requires high computation power and may degrade when there is distortion in the audio signals. Recently, posteriorgram feature vectors [159] are extracted from audio segments using Artificial Neural Network (ANN). The posteriorgram feature is a vector with components that are the probability vector are posterior probabilities of English phonemes at any given frame of speech. Template matching using DTW is then used to compare the query's posteriorgram vector against the database' posteriorgram vectors to identify the closest audio segments using distance metric.

Another template matching approach uses HMM to model audio query, which is similar to the strategy applied in KWS [61, 150–152] technology. The KWS task is to detect desired words or phrases in speech corpus for applications such as topic identification. Most of the KWS systems are based on two HMMs as reviewed in [160]: one represents the keyword and another one is a background/garbage model. Given a speech utterance, Viterbi algorithm is used to decode the utterance using the two models. If the likelihood on the the keyword model is higher than the garbage model, the keyword is considered as detected. To build a keyword model, different approaches have been examined in different KWS spotter systems: i) the word-based [150] spotter uses a collection of the keyword samples to train a keyword model; ii) phoneme-based [151] spotter directly concatenates phoneme models to form a keyword model; and iii) LVCSR-based [61, 152] spotter also concatenates existing phoneme/sub-word models to represent a keyword, and its garbage model only captures the words in the lexicon except the keywords. Hence, such KWS systems create one HMM for every keyword by either modifying from a well trained recognizer or training a new model uses enough samples of the keyword.

The KWS techniques have been modified and applied on general audio segments which contain both speech or music for audio retrieval. Velivelli *et al.* [20] proposed to create an HMM using an audio query and two background HMMs using randomly selected data. The models are then concatenated by having the query HMM between the two background HMMs to decode the database. The audio segments corresponding to the states of the query HMM are then selected as potential instances of the query. The queries used in their application have duration of 40 seconds. Such duration of audio provides sufficient data to estimate the query model. However, the length of shorter queries such as a commercial last only 10-30 seconds, and is not sufficient to train models for this suggested method.

The third category of audio pattern retrieval techniques use speech recognizers to transcribe audio data into sub-word units or phonetic lattice instead of the conventional top-1 sequence. To address the OOV problems inherent in LVCSR systems, the vocabulary-independent approach [24, 25, 63, 161–163] to speech indexing has been proposed. The speech data is first transcribed into either phonetic lattice or sub-word sequences for subsequent processing. E.g., multiple hypotheses captured in a phonetic lattice have been

examined [164]. To index the speech corpus for efficient retrieval, researchers [24, 25, 161] have proposed to index the speech corpus using information such as the expected word counts vector [162] extracted from the lattice. To search for a spoken query, the same information is extracted from the query and used to find similar indexed vector by computing their distance. While the acoustic lattice can be used to locate a spoken query, searching through multiple hypotheses may generate many false-positives [63]. Hence, Logan *et al.* [63] proposed to index speech corpus using sub-word units which are in the form of within-word sequences of characters. Such sub-word units are automatically generated from phonetic transcription of words and may represent phonemes, several adjacent phonemes, or a word. The collection of the sub-word units found are then used as a dictionary in the phoneme-based speech recognizer to convert a spoken query into sub-word sequence for retrieval. Their experiments show that such approach is able to retrieve OOV queries in monolingual corpus.

The above mentioned techniques to retrieve audio patterns deal with either speech data or music data. To process mixed content audio, the HMM template matching approach can be used. However, sufficient samples of the queries are required to robustly train the query model.

Motivated by the previous work, this chapter proposes a method for retrieval of mixed music and speech patterns by extending the vocabulary-independent speech indexing and retrieval framework [25, 162] using ASM models and Vector Space Model (VSM) [124] to index the corpus for retrieval. The ASM creates models in a data-driven approach and can be used to process music data. A set of ASM models are trained from music data and combined with phoneme models to decode audio data containing both speech and music into the top-1 transcription or lattice output. The statistical information generated from the decoded output is then used to index the audio data and retrieve the audio patterns using the VSM approach as described in [124, 162].

5.2 Mixed Music and Speech Pattern Retrieval Framework

The proposed framework to retrieve mixed music and speech audio pattern in broadcast audio database is illustrated in Figure 5.1. There are 4 major steps in the system: 1)

Generation of the acoustic models for both speech and music; 2) Decoding of audio signals into token sequence or lattice; 3) Building of n-gram vectors using VSM to index the audio archives; 4) Retrieval of audio query using the n-gram vectors. In the following sections, each step is described in details.

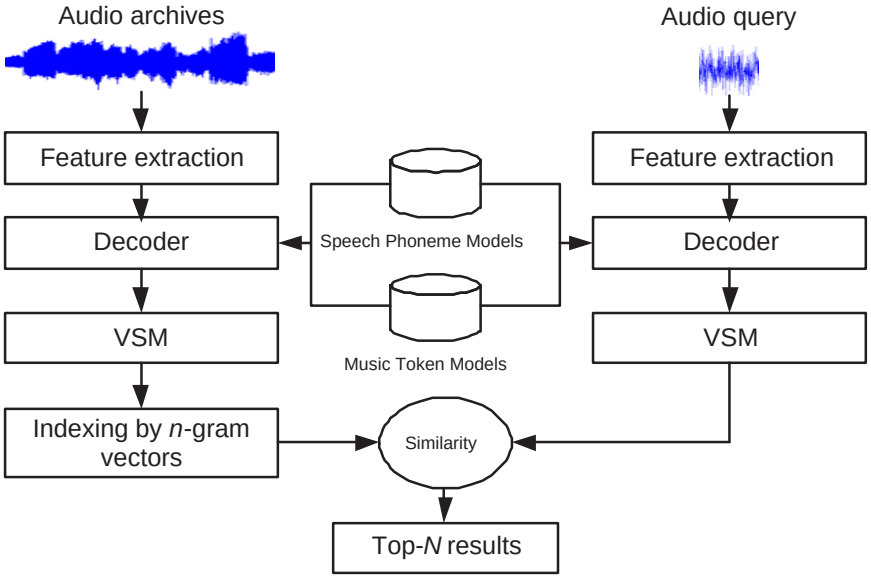


Figure 5.1: Overview of audio indexing and retrieval system.

5.2.1 Generation of Speech Acoustic Models and Music Token Models

The acoustic model of a conventional ASR system is the HMM [114] which is used to model the acoustic units such as phonemes [79, 81] of a language. As the data of interest in this work is mixed speech and music audio, conventional ASR system cannot directly process music data into reliable token sequence/lattice. It is necessary to extend the acoustic model to cover music data. Hence, one focus of this work is to examine the use of an equivalent acoustic model for music. The music model and the conventional ASR acoustic model can then be combined to decode mixed content audio data.

The training of HMMs in ASR is carried out in a supervised manner and is therefore language and task dependent [114]. As such, the speech data needs to be manually transcribed. However, there is no simple or effective way to generate music transcription to train music models.

To remove the need to define the music acoustic models and manual transcription for music, this work proposes to use data-driven acoustic modeling technique, ASM [18], to train the music HMMs. These music HMMs are denoted as music token models. These token models are combined with a well-trained phoneme recognizer to form a general decoder to decode broadcast audio.

This section describes the decoder used to process the general broadcast audio data. The decoder consists of well-trained speech acoustic models and data-driven music token models.

5.2.1.1 Speech acoustic models

The basic acoustic units of a speech recognizer can be words [165], subwords [166] and phonemes [79, 81]. Depending on the size of the speech recognition task, from small, medium to large vocabulary, the acoustic models used by the recognizer may be different. For example, for LVCSR task, the existing state-of-the-art recognition systems use HMM to model phonemes of a language. This is because HMM is able to capture the statistical behavior of the speech features of a phoneme and has the ability to compensate for some of the variability of speech production [81].

In this work, English broadcast news corpus is examined. Inspired by [81], 39 HMMs to model the 39 phonemes of the English language with an additional silence model are defined. To generate the acoustic models, the widely used large vocabulary Aurora-4 speech corpus [167] and HTK [118] were used to train the HMMs. Each HMM consists of 3 left-to-right states and each state has 16 Gaussian mixtures. The speech features used are the MFCC features as well as their first and second derivatives with mean variance normalization. By applying the recognizer together with a bigram language model and vocabulary size of 5000 words to a continuous speech recognition task of 20 minute clean speech data, the recognizer achieved 52% mono-phone recognition accuracy and 81% word accuracy.

Since the procedure to train HMMs requires transcribed data, it cannot be easily applied to train music HMMs due to the lack of transcription. In the following section, an unsupervised technique to create music token models is decribed.

5.2.1.2 Music token models

In Chapter 4, ASM has been examined to generate acoustic models on untranscribed speech data. In this work, it is used to automatically train 40 token models for music. The number of music tokens is chosen to be 40 so that the final system will have the same number of speech and music HMMs.

The training procedure of ASM is modified to train music token models as music does not have obvious boundaries which can be used to divide it into short segments. To initialize the models, music data is first divided into equal length segments, and then clustering is performed. The ASM training process, illustrated in Figure 5.2, is applied

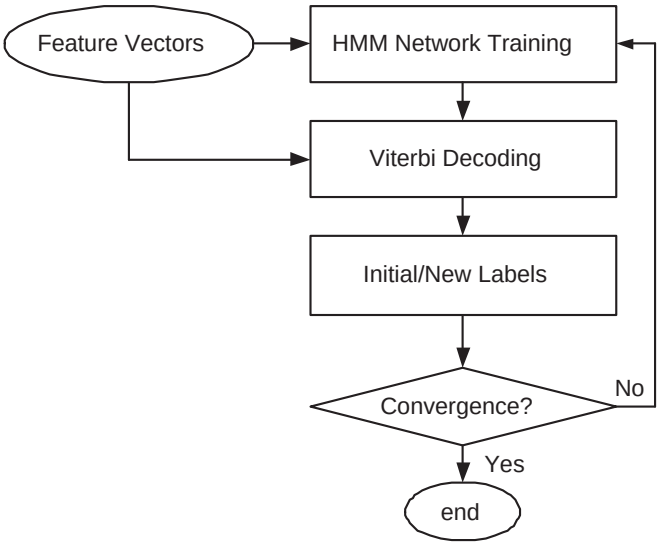


Figure 5.2: Iterative training procedure of music ASMs.

to generate 40 HMMs for music data. The ASM training procedure is revised as follows:

Step 1) Audio feature vectors are generated from music signal and then divided into equal length segments without overlapping.

Step 2) The mean vector in each segment is extracted and used to group the segments into 40 clusters using k-means clustering. The segments in the entire corpus are labeled with the found cluster identity. The initialization step is illustrated in Figure 5.3.

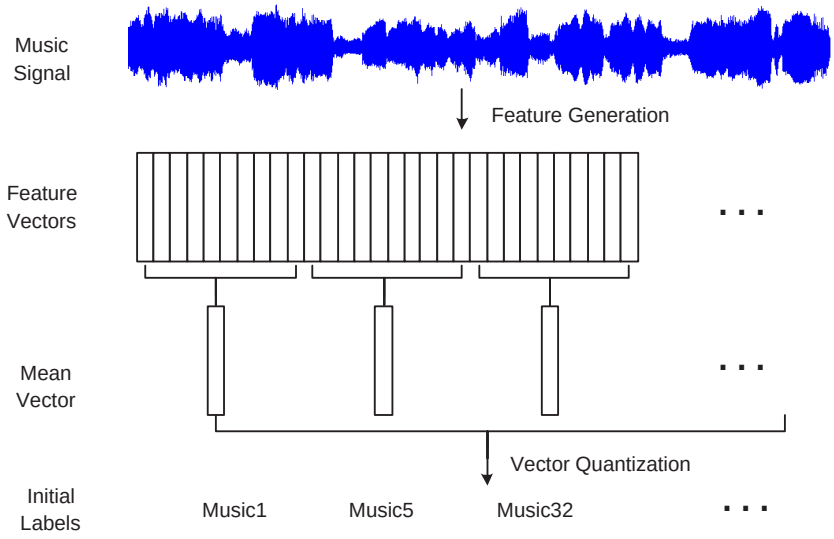


Figure 5.3: Initialization of the training of music ASMs.

Step 3) A set of 40 initial HMMs are created to model the segments with the cluster identities found in Step 2. The HMM parameters are first estimated using Baum-Welch algorithm using the extracted feature vectors belonging to the assigned labels.

Step 4) The trained 40 HMMs are used to decode the music corpus using Viterbi decoding.

Step 5) The HMM parameters are then re-estimated with the new labels found in Step 4.

Step 6) Repeat steps 4 and 5 till convergence.

The above approach iteratively generates better HMMs to represent the music data. Each HMM can be interpreted as a model to represent a particular sound class just like k-means clustering. Its ability to capture temporal information of sound class, however makes HMM more robust than k-means clustering. In the implementation, the music training corpus consists of 46 pieces of music played by five different instruments and 20 pieces of English pop songs performed by both male and female singers. All the music pieces are converted from MP3 format and down-sampled to 16KHz to be consistent with

the speech training data. At the end of the training procedure, the total log-likelihood score of the training feature vectors on the HMMs converges as shown in Figure 5.4 and 40 music HMMs are obtained.

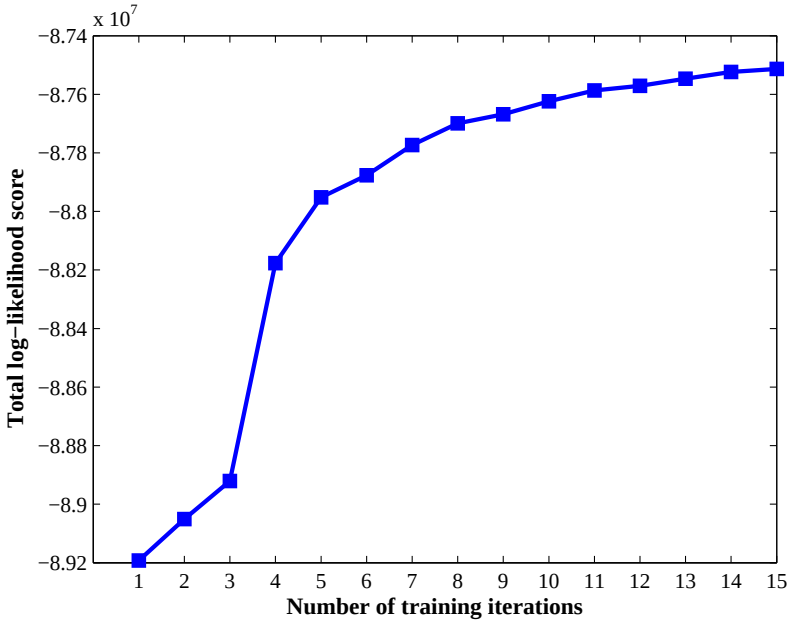


Figure 5.4: The total log-likelihood score of the training data feature vectors on the HMMs converges after 15 training iterations.

With the trained 40 English phoneme models and 40 music token models, the system now is able to convert a mixed speech and music audio signal into text-like transcripts.

5.2.2 Vector Space Model (VSM)

The HMM decoder can generate results in different formats [118], e.g. top-1 acoustic sequence, top-n acoustic sequences, and lattice. The top-1/top-n sequences imply the best path/the best n alternative paths generated from decoding process; The lattice is an intermediate representation of the decoding process which contains multiple hypotheses of the decoding output. This system considers two of the output formats: the top-1 acoustic sequence and the lattice. Phoneme co-occurrence sequence information will be extracted from these two forms of decoding output to generate indexed vectors using VSM. The VSM technique is discussed in the following paragraphs.

The VSM [168] was first introduced to represent text documents as vectors of terms in the text retrieval task. The elements of the vectors are counts of occurrence of respective words in a given text. Hence the vector captures the frequency of occurring word in the given text. This idea has been widely applied and extended in different areas such as text retrieval [169], language identification [124], music retrieval [170], music genres classification [171], etc.

In the language identification task [124], a VSM-based approach was proposed to extract the n-gram statistics of the phonemes for a given speech segment to generate a vector termed as *bag-of-sounds* vector. Following this method, the decoded acoustic sequence or lattice can similarly be processed to generate an n-gram vector to index the audio segment. In other words, the VSM is used to transform the decoder's output into a vector containing the segment's feature n-gram statistics. Figure 5.5 illustrates the generation of the n-gram vector from the decoding output.

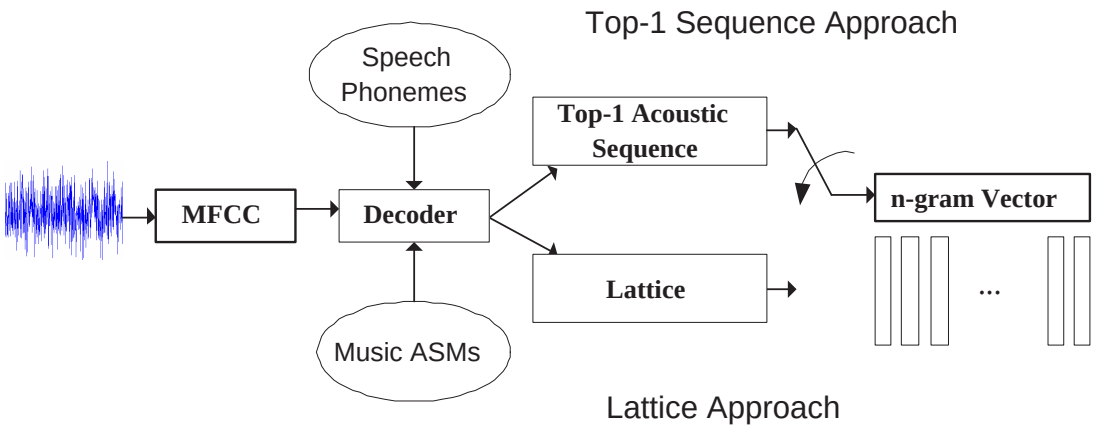


Figure 5.5: Two approaches to generate n-gram vector from decoded output.

5.2.2.1 Top-1 acoustic sequence

The top-1 decoded output is a sequence of phonemes and music token identities with timing boundary to represent the given audio segment based on maximum likelihood decoding criterion [114] using the Viterbi algorithm. The top-1 sequence can be used to generate an n-gram vector by counting the frequencies of each n-gram terms. For

example, the 1-gram (unigram) vector can be generated by counting the occurrences of the acoustic/music models' occurrence for the given audio segment. In this work, the total number of HMMs is 80, i.e. 40 music, 39 speech plus 1 silence model. Hence the vector generated is of length 80. The elements of the vector is either a count or a normalized count of the model's occurrence for the given audio segment. Such vectors have been referred to as "hard-counting" of the decoded output in [170]. Alternatively, bigram VSM, i.e. a vector of 81×81 (one additional dummy model is introduced to represent the starting/ending point of the top-1 sequence) can be also generated. Each term in the vector registers the frequency of occurrence for the corresponding bigram term.

5.2.2.2 Phonetic lattice

Another possible output from the decoder is the phonetic lattice. A lattice is a connected directed acyclic graph in which each edge is labeled with a phoneme hypothesis and a likelihood value. Each path in the acyclic graph provides a hypothesis of the sequence of phonemes spoken in the utterance [164]. Hence, the lattice compactly stores multiple hypotheses for the given audio segment.

Phonetic lattice has been widely used in many speech processing areas such as speech indexing and retrieval [25, 164], language recognition [172], etc. There are several reasons to use lattice: 1) Lattice can provide more accurate information than top-1 decoding sequence. 2) Similar to the use of phonemes and sub-words to represent the corpus [63], its application also allows OOV to be detected.

The lattice generated by the decoder can also be used to generate n-gram vector. The SRILM Toolkit [173] can be used for this purpose. The expected count of one n-gram term can be computed using the lattice posterior probabilities from all the arcs corresponding to the desired n-gram term. Details of generating n-gram vector from lattice can be found in [172]. This process is sometimes referred to "soft-counting" of the decoded output [170].

In the next two sections, the n-gram vectors generated from both top-1 acoustic sequence and lattice are incorporated in the proposed audio indexing and retrieval framework.

5.2.3 Audio Archives Indexing

This section describes the broadcast audio indexing process. Given an audio archive to be indexed, the above mentioned n-gram vector is first created for each segment in a sliding window of t seconds duration with s seconds shift. These vectors are then stored as the indexed vectors of the audio archive. The retrieval process can simply be a Brute-Force similarity comparison to find the closest N indexed vectors to the query segment's n-gram vector. The details of the implementation are described in the following paragraphs.

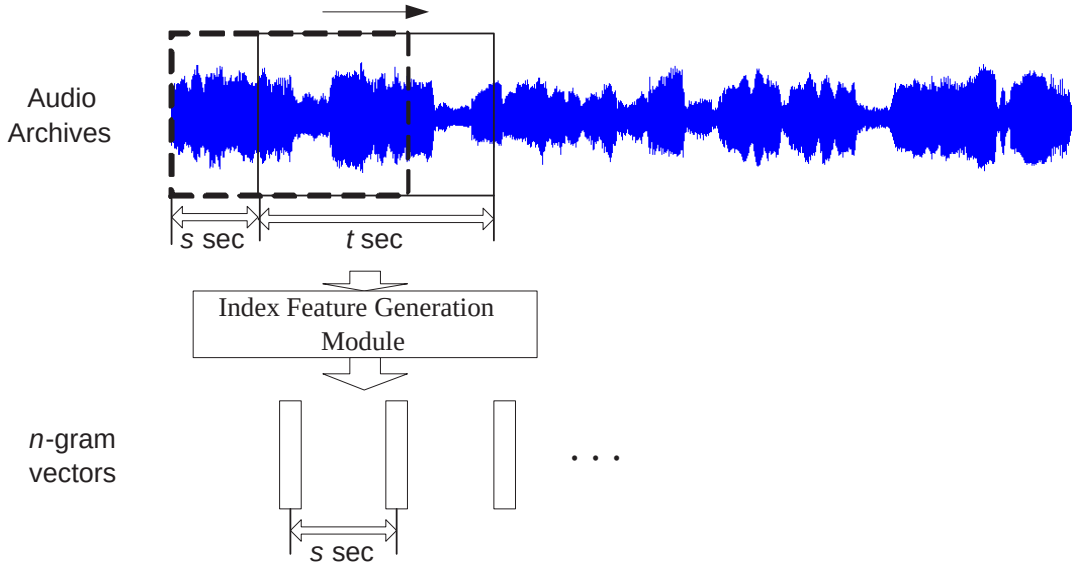


Figure 5.6: Indexing of audio archives. The n-gram vectors can be generated from either top-1 acoustic sequence or lattice.

Figure 5.6 shows that an indexed vector is created every s seconds. The n-gram vector is extracted from a t seconds window. The audio signal in each t-second window is transcribed using the speech and music models jointly to generate the top-1 sequence or lattice. It is clear that there will be decoding errors for audio segments which contain both speech and music. However, as the main goal of the framework is to represent the speech segment into a token sequence, the accuracy is not as crucial as LVCSR systems whose goal is to generate a transcript.

To index the corpus, the decoding output is converted to n-gram statistics [173] and the n-gram vector is used as the indexed term. The details of converting the top-1 and

lattice n-gram statistics into vector representation have been discussed in Section 5.2.2. The indexed vectors are generated for the entire audio database and stored for audio query retrieval in the next section.

5.2.4 Audio Retrieval

Given an audio query segment, the indexed vector generation method as discussed in Section 5.2.3 is applied to generate the query's n-gram vector. To find similar audio segments to the query, the proposed framework simply compares the similarity of the query' vector against the entire archive's indexed vectors to retrieve the top-N closest matches. Figure 5.7 illustrates the audio query retrieval process. As this work primarily emphasizes on measuring accuracy, more efficient methods of retrieval such as inverted index [174] and hierarchical clustering [105] are not considered for this work.

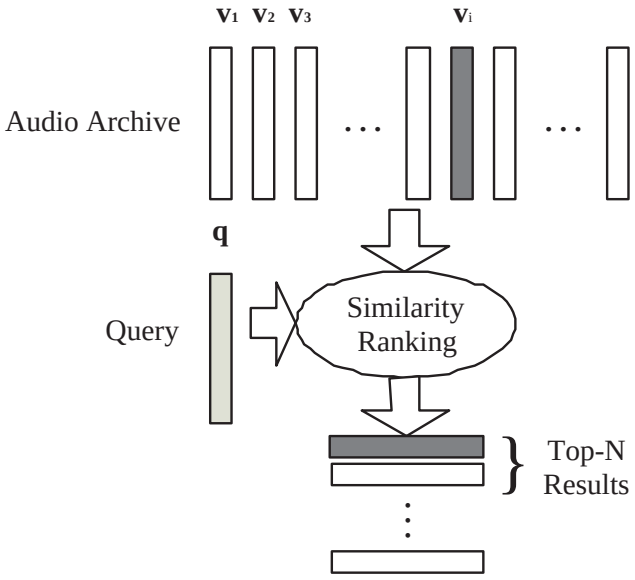


Figure 5.7: Audio query retrieval process. A brute force approach is adopted to find the N-nearest matches.

The given query segment should have a duration longer than t-seconds, where t-second is the duration window used to generate the feature vectors for the indices. For a query segment, the feature extraction module extracts an n-gram feature vector $\mathbf{q}$. If the query duration is sufficiently long, more query vectors $\mathbf{q}$ can be generated. The query vectors $\mathbf{q}$ are compared with all the audio archive feature vectors $\mathbf{v}$ to find the closest N matches

using the Pearson correlation coefficient measure [96]. The Pearson correlation coefficient between vector $\mathbf{q}$ and $\mathbf{v_i}$, where i denotes the i^{th} index vector, is given by Eq. 5.1.

$$r_i = \frac{\sum_{k=1}^{n}(q_k - \overline{q})(v_{ik} - \overline{v_i})}{\sqrt{\sum_{k=1}^{n}(q_k - \overline{q})^2 \sum_{k=1}^{n}(v_{ik} - \overline{v_i})^2}} \tag{5.1}$$

In this way, the segments of the archive are ranked in terms of similarity to the query.

5.3 Experiments

5.3.1 Corpus

The proposed audio retrieval system is evaluated using an audio corpus extracted from TRECVID 2003 and 2004 [34] database. The TRECVID video database was recorded from the ABC and CNN network news in 1998. The content of the videos is mainly broadcast news and TV commercials, where the news include both studio news and in field reports. A total of 28 hours of audio signal was extracted from selected videos and down-sampled to 16KHz which captures most of spectral information of both speech and music signal.

As the proposed system focuses on retrieval of mixed speech and audio data, TV commercial are used as queries as they usually contain both speech and music. All the commercials were manually labeled in the 28 hour database, and 100 unique commercials were selected as the query commercials. In the database, there are 242 additional instances of the 100 unique commercials. The task is to find the correct instances of commercials that are the same as the query.

5.3.2 Experiment Setup

MFCC features are used as the speech features for acoustic modeling. The dimension of the feature vectors is 39, including the first 13 MFCC features as well as their delta and double-delta features. The duration of each frame is 25ms with a hop size of 10ms. As there are three corpus involved and they are recorded using different equipments and under different environments, it is necessary to normalize the feature vectors such that the speech model trained from Aurora-4 and the ASM model trained from music corpus matches the TRECVID data well. Utterance-based cepstral mean and variance

normalization (MVN) [175, 176] is used to normalize the mean of each feature dimension to 0 and variance to 1. It has been shown in [176, 177] that utterance-based MVN is able to improve the performance of LVCSR significantly in mismatched training/testing conditions as this technique effectively compensates the channel effect.

To evaluate the effect of query's duration t on performance, t was chosen to be 3sec, 5sec and 7sec with shift s was set to 0.5sec. The system performance was studied using both top-1 sequence and lattice approaches to generate index vectors. Both unigram ($n = 1$) and bigram ($n = 2$) vectors were generated as index vectors. Since there are a total of 80 HMMs including 40 English phoneme models and 40 music ASM models, the unigram vector has 80 dimensions. The bigram vector has 6561 (81×81) dimensions as the starting point and ending point of the sequence are considered as one extra token.

To evaluate the effect of combining speech phoneme models and music token models on performance, the proposed framework is compared to a baseline system which developed following the speech indexing and retrieval techniques proposed in [25, 162]. The baseline system uses only 40 speech phoneme HMMs trained from Aurora-4 corpus to recognize the audio data. In the baseline system, the unigram vector has 40 dimensions and bigram vector has 1681 dimensions.

5.3.3 Evaluation Criteria

To evaluate the retrieved results, retrieved boundary of the segment within ± 1 second of the actual boundary is considered as correct answer. In the experiments, the top-10 results were retained for evaluation since no queries in this database has more than 10 instants of the query.

The recall and Mean Average Precision (MAP) are used to evaluate the system performance and they are defined by the equations below:

$$\text{Recall} = \frac{Number\ of\ relevant\ results\ retrieved}{Total\ number\ of\ actual\ relevant\ results} \tag{5.2}$$

$$\text{MAP} = \frac{1}{M}\sum_{m=1}^{M} AP_m \tag{5.3}$$

$$AP_m = \frac{1}{K}\sum_{i=1}^{K} \frac{i}{r_i} \tag{5.4}$$

where M is the total number of queries, K is the number of correctly retrieved results for query m, and r_i is the rank of relevant result i.

5.3.4 Evaluation Results on Clean (Original) TRECVID Data

The recall and MAP on the clean data are shown in Table 5.1 and Table 5.2 respectively. The combined models are denoted as "comb." and speech-only recognizer is denoted as "speech" in the results. From the two tables, the following observations are made:

Table 5.1: Recall on clean broadcast audio data.

t	Top-1 Sequence				Lattice			
	Unigram		Bigram		Unigram		Bigram	
	comb.	speech	comb.	speech	comb.	speech	comb.	speech
3sec	0.93	0.69	0.95	0.87	0.98	0.93	0.98	0.97
5sec	0.95	0.76	0.97	0.92	0.98	0.96	0.99	0.98
7sec	0.93	0.82	0.94	0.91	0.97	0.96	0.98	0.97

Table 5.2: MAP on clean broadcast audio data.

t	Top-1 Sequence				Lattice			
	Unigram		Bigram		Unigram		Bigram	
	comb.	speech	comb.	speech	comb.	speech	comb.	speech
3sec	0.93	0.61	0.92	0.87	0.95	0.92	0.96	0.95
5sec	0.96	0.74	0.96	0.89	0.97	0.95	0.97	0.96
7sec	0.92	0.77	0.92	0.91	0.96	0.96	0.98	0.97

(i) The system using proposed combined model consistently outperforms the baseline system using only the speech model. When top-1 sequence and unigram counts are used, using combined model produces significantly better results than using speech model only. The results show that the proposed combined model is effective in representing audio data that contains both speech and music. When lattice and/or bigram counts are used, the differences between the two systems become smaller. This is due to the fact that the lattice provides more stable n-gram counts than than the top-1 sequence and bigram counts provide even more discriminative information than unigram counts in indexing and retrieval tasks [24, 25, 161].

(ii) Using lattice outperforms using top-1 sequence approach in all the test cases. This can be easily explained by the fact that the lattice representation captures more statistics than just the top-1 decoder output and hence is more robust and accurate. For example, an incorrectly recognized phoneme/token in the top-1 output will cause the corresponding elements in the n-gram statistics to be in error; While the lattice representation will still provide "soft-counting" of the probability of occurrence for the corresponding elements of the n-gram statistics.

(iii) As both approaches utilize the temporal information of the signal, the window size t is an essential parameter to the systems. The performance of the combined model is not affected by t very much; while the speech-only model with $t = 5, 7$ performs better than $t = 3$. The performance is stable when the length of query is more than 3 seconds. Such finding overcomes the limitation of [20] which requires the query is more than 40 seconds.

The experimental results on lattice bigram approach show that the recall and MAP of both speech-only system and combined model framework are more than 95%, and the combined model slightly outperforms the speech-only model in all the test cases. This surprising result may be due to the fact that the test data is clean. To further investigate the robustness of the proposed combined model framework, noisy data was generated and used in the following experiments.

5.3.5 Evaluation Results on Noisy TRECVID Data

To show the robustness of the proposed framework, the original (i.e. clean) TRECVID data was corrupted by adding recorded babble noise into the clean audio signal. The babble noise used is from the Aurora-2 database [178], which was recorded in real world environments. The noise is added to the audio signal of the corpus to achieve two different signal-to-noise ratios (SNRs), i.e. 15dB and 5dB. While the audio archives are corrupted with noise, the queries are left unchanged, i.e. clean condition, to examine the effect of mismatch between the query and the instances.

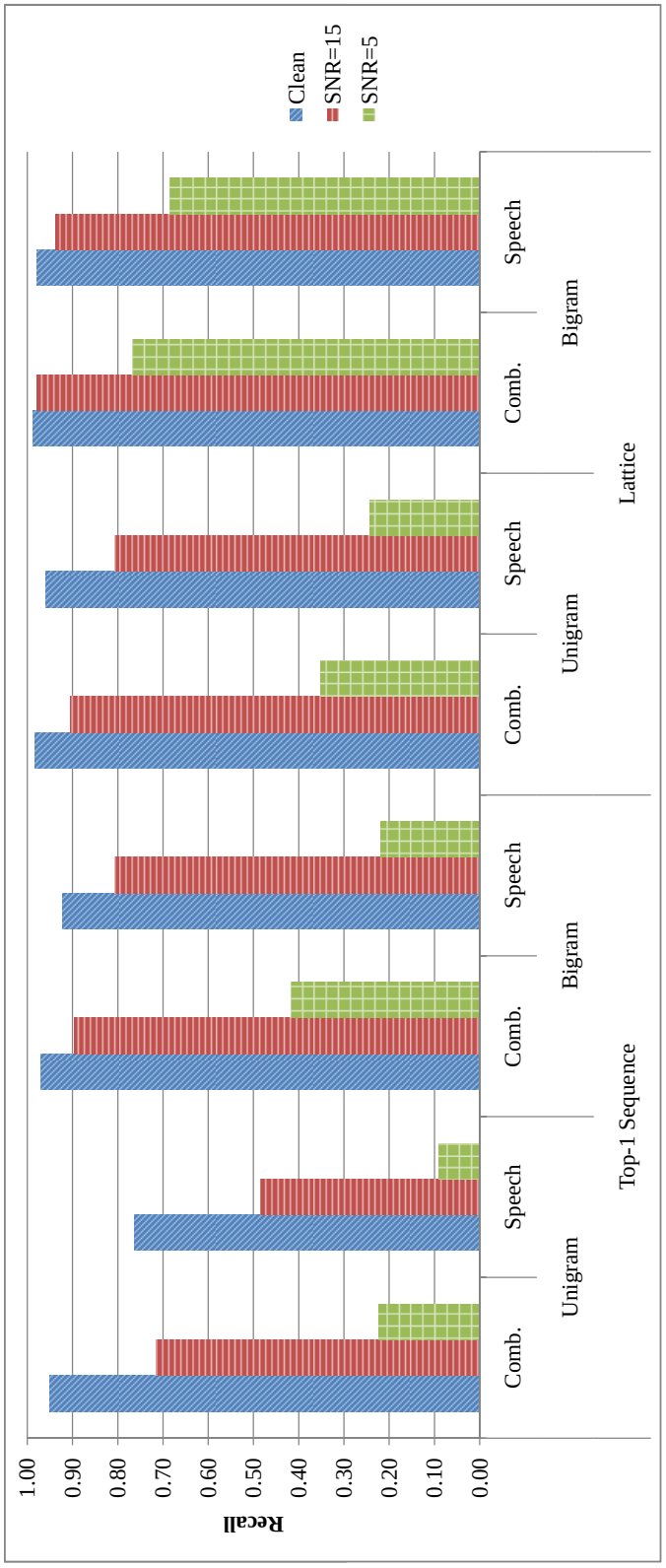


Figure 5.8: Performance comparison (Recall) under different noise conditions.

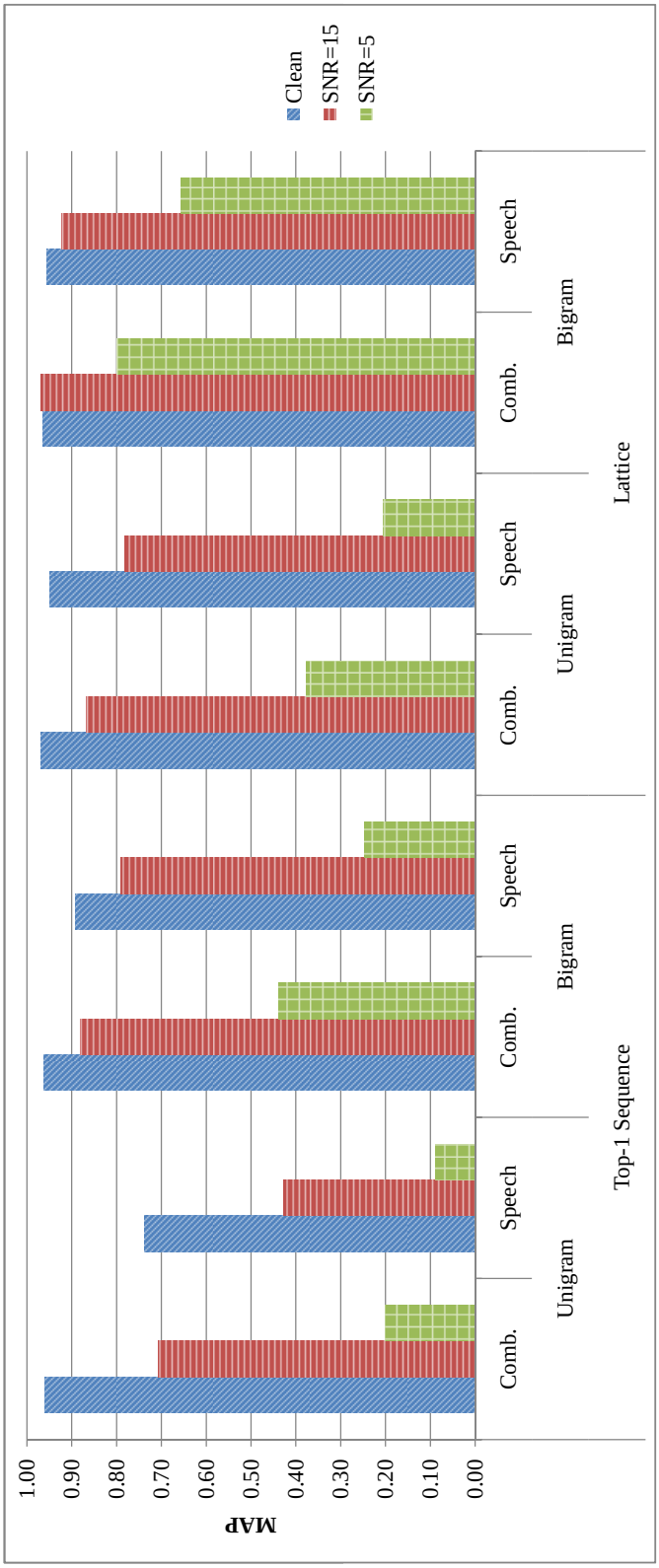


Figure 5.9: Performance comparison (MAP) under different noise conditions.

The audio indexing and retrieval is carried out on the corrupted archives using both speech-only and combined model, separately. The previous experimental results in Section 5.3.4 show that the overall retrieval performance is not sensitive to window size, i.e. the length of the query, for $t >= 5$. Hence, t was set to 5 seconds in this experiment.

The retrieval performance of the two systems measured by recall and MAP are shown in Figure 5.8 and Figure 5.9 respectively. In addition, by comparing to the performance on clean data, the relative degradation (in percentage) on recall and precision is shown in Table 5.3 and Table 5.4, respectively. From the results, it is observed that the performance of both systems are affected significantly by noise, and the degradation is more serious in lower SNR level. However, the system using combined models is more robust than the system using speech-only model. Compared to the results in the previous experiments, the performance difference between the two systems becomes more obvious in noisier conditions. The reason for more degradation of speech-only model than combined model may be that the speech-only model is less discriminative for music data as it was trained to represent only the phonemes of English. In clean condition where the query and the instances are almost identical, the decoded output of the query and instances will be similar even if speech-only model is used. However, when the instances are corrupted and becomes different from the query, the less discriminative speech-only model will generate more unstable output than the combined model which covers both speech and music information. Hence, the performance of using speech-only recognizer degrades more than the combined models in noisy conditions.

Table 5.3: Relative degradation on recall.

	Top-1 Sequence				Lattice			
	Unigram		Bigram		Unigram		Bigram	
	comb.	speech	comb.	speech	comb.	speech	comb.	speech
SNR=15	25%	37%	8%	13%	8%	16%	1%	4%
SNR=5	77%	88%	57%	76%	64%	75%	22%	30%

5.4 Conclusion

This chapter describes an audio pattern retrieval framework which can process speech and music simultaneously. The data-driven ASM technique is used to generate the music

Table 5.4: Relative degradation on MAP.

	Top-1 Sequence				Lattice			
	Unigram		Bigram		Unigram		Bigram	
	comb.	speech	comb.	speech	comb.	speech	comb.	speech
SNR=15	26%	42%	9%	11%	11%	17%	0%	3%
SNR=5	79%	88%	54%	72%	61%	78%	17%	31%

token models followed by incorporating them with the speech phoneme models to decode general broadcast audio signal into both top-1 acoustic sequence and lattice. In addition, VSM is introduced to extract statistics from the two types of decoding output, and the statistical information which is in the form of n-gram vector is used to index the audio archives. In such case, the query segment can be easily compared to the audio archives by measuring the similarities between the n-gram vectors in brute-force manner. Therefore, the audio segments which are most similar to the query can be retrieved.

The experiments have also been carried out to examine the use of joint music/speech HMMs to decode audio data. By applying the framework on both clean and noisy TRECVID data, the experimental results show that the proposed combined model outperforms the speech-only model which is lack of music information. The higher recall and MAP scores in all 3 testing cases show that the combined model is robust and effective to transcribe broadcast audio data containing both music and speech.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis explored several unsupervised techniques for audio pattern discovery and retrieval. There are 3 major contributions, namely the unsupervised discovery of repeating segments in music, the automatic discovery of recurrent patterns in speech, and the retrieval of audio patterns containing both speech and music.

The first contribution is the proposal of an efficient method to find repeating segments in music automatically. The AMG is used to detect recurrent patterns in a sequence of multivariate vectors such as music feature vectors. The online suffix tree construction algorithm is extended to accept vectors as input so that exact matching criterion is avoided. To make the algorithm more robust, segment-to-segment similarity is used to smooth outliers in the sequence. Finally, the candidate motifs obtained by the AMG method are refined using a sparse similarity matrix to locate the final motifs. To evaluate the performance, the proposed AMG was applied to automatically detect repeating segments in music and two time series data sets. The experimental results show that the AMG method can significantly reduce the computational cost and memory requirements without sacrificing the detection accuracy compared to the traditional self-similarity approach [33].

The second contribution is the proposal to automatically discover recurrent patterns such as words and short phrases in a multi-speaker speech corpus using an iterative model merging approach. The proposed approach begins with a set of data-driven acoustic unit models generated by the ASM technique. An information theory based criterion is then

used to select two models with high co-occurrence for merging. The model merging process continues till a predefined stopping criterion is met. The obtained patterns are ranked according to their frequency of occurrence in the corpus and stability. The top-ranked patterns are then eventually selected as the final patterns. Experimental results on the TIDIGITS corpus show that the proposed approach is able to automatically discover 10 out of 11 digits with high F1 score.

In the third contribution of this thesis, an audio pattern retrieval technique is examined to simultaneously process mixed music and speech audio data. To model both music and speech, a combined model is built by incorporating two separate acoustic models: one of the acoustic models is trained from the Aurora-4 corpus to model the English phonemes and the other acoustic model is trained from a collection of music to represent music sounds. Using this combined model, the mixed speech and music audio archives are decoded to top-1 token sequences/lattice and indexed by n-gram statistics of the decoded output. The proposed framework has been applied on a query-by-example task for broadcast audio corpus which contains both music and speech. Experimental results show that using the proposed combined model improves both the recall and precision of pattern retrieval over a baseline system which uses the speech-only model.

In this thesis, the problem of "Audio pattern discovery and retrieval" has been addressed for different types of audio content. I believe that the presented unsupervised approaches can be further explored to benefit many applications such as language acquisition and automatic audio content summarization task. The following section presents some suggestions for future work.

6.2 Future Work

As discussed in Chapter 2, applications to automatically analyze and summarize audio content are useful and important. In addition, the amount of audio data is growing rapidly and currently there is no robust way to retrieve them. Existing approaches to organize audio data first generate a basic transcription of the spoken audio using an LVCSR system and then index the generated transcription [2, 11, 38, 48, 49]. However, if the LVCSR system is not available or the data is from an unknown language, existing

techniques will not help. In addition, if the audio content has a mix of speech, music and sound effects, these non-speech sounds will also pose a problem to existing techniques which basically only focus on speech.

The suggested future work includes:

(i) **Feature enhancement**

To further improve the performance of the proposed techniques in this thesis, feature enhancement techniques can be examined to generate more robust features. For example, music features which address transposition [179] can be applied to the work in Chapter 3. Music transposition refers to the process of moving a collection of notes up or down in pitch by a constant interval so that transposition to a common key can help the proposed technique in Chapter 3 to find more repeating segments. Moreover, feature compensation techniques such as cepstral variance normalization and power term can be used to reduce speakers' variations in speech data before applying the proposed technique in Chapter 4.

(ii) **Language acquisition**

The research work on language acquisition using computational methods has been explored since the early 2000s [59]. Computational models [106, 107, 180, 181] have been proposed to acquire human verbal communication behavior such as how an infant learns languages. The existing work was inspired by phycological study that shows how an infant begins to understand the sematic meaning of a spoken term by associating the spoken term to an object [182]. For example, when an infant is being taught how to speak, the caregivers usually repeat the same word or sentence with additional visual or tactile information such as pictures and body movements to communicate the semantic meaning.

A possible extension of the speech pattern discovery research is to assign semantic meanings to identified recurrent patterns. To assign semantic meanings to the patterns, visual information can be included as an input. For example, an image can be associated with each utterance in the corpus according to the scenario of the utterance. By discovering speech patterns in these utterances, the corresponding

images can also be mined to identify the most frequently occurred shapes, colors or objects to associate with the speech pattern.

One possible scenario for such research is the task to train human-like auditory perception system for social robots. The current auditory system used by robots is the conventional speech recognizer which is pre-defined and trained using transcribed speech data. Hence, this auditory system suffers from all the limitations that the given speech recognizer may encounter. Hence, the proposed research work may enable robots to learn languages from scratch given both auditory signals and visual signals from its own operating environment.

(iii) **Audio content summarization**

Audio content summarization system is an important application that is not yet mature. By audio summarization, I encourage the scenario presented in Figure 6.1 in which an automatic system is used to process unstructured broadcast audio corpus to generate a useful table of contents for users to browse. The technology described in this thesis is applicable to build such system. For example, to automatically detect repeating content in broadcast audio programs, the AMG technique proposed in Chapter 3 can be applied on the audio data's n-gram vectors extracted using the technique proposed in Chapter 6. By detecting the repeating segments, a rough content structure of the audio programs can be inferred using prior knowledge. For instance, an LVCSR engine can be used to transcribe the speech portion in order to get a brief news description.

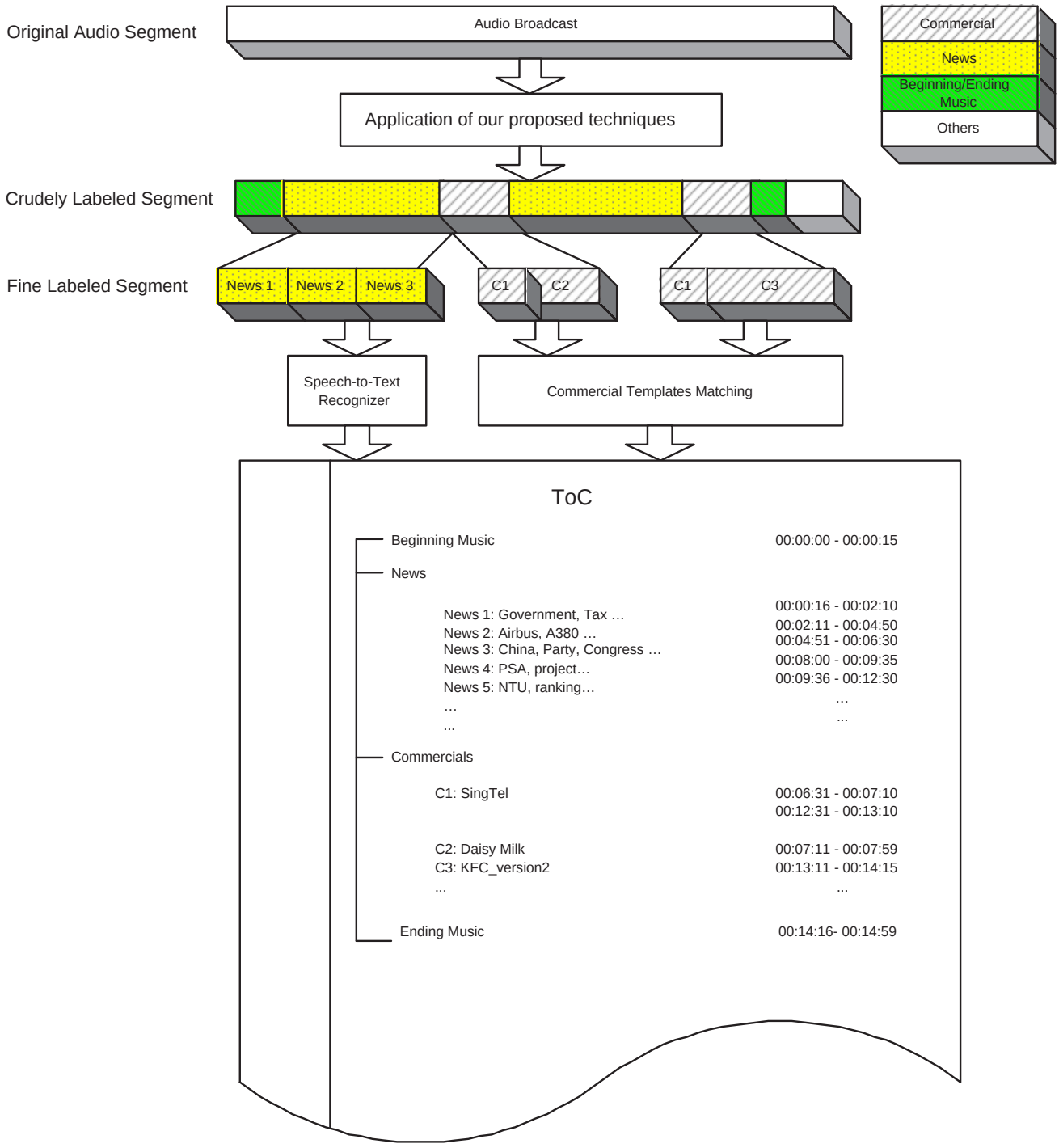


Figure 6.1: A overview of the proposed application of audio content summarization.

Appendix A

Publication

Journal Paper

(i) **Lei Wang**, Eng Siong Chng and Haizhou Li, "A tree-construction search approach for multivariate time series motifs discovery," *Pattern Recognition Letters*, vol. 31, no. 9, pp. 869-875, 1 July 2010.

Conference Paper

(i) **Lei Wang**, Eng Siong Chng and Haizhou Li, "An iterative approach to model merging for speech pattern discovery," in *Proceedings of 2011 Asia-Pacific Signal and information Processing Association Annual Summit and Conference (APSIPA ASC 2011)*, Xi'an, China, October 18 - 21, 2011.

(ii) **Lei Wang**, Eng Siong Chng and Haizhou Li, "Efficient repeating segments discovery in music using adaptive motif generation algorithm," in *Proceedings of 2009 Asia-Pacific Signal and information Processing Association Annual Summit and Conference (APSIPA ASC 2009)*, Sapporo, Japan, October 4 - 7, 2009.

(iii) **Lei Wang**, Eng Siong Chng and Haizhou Li, "Efficient sparse self-similarity matrix construction for repeating sequence detection," in *Proceedings of 2009 IEEE International Conference on Multimedia and Expo (ICME 2009)*, New York City, USA, June 28 - July 3, 2009.

(iv) **Lei Wang**, Haizhou Li and Eng Siong Chng, "A vector-based approach to broadcast audio database indexing and retrieval," in *Proceedings of IEEE International Conference on Multimedia and Expo 2007 (ICME 2007)*, Beijing, China, July 2-5, 2007, pp. 512-515.

(v) Tomi Kinnunen, Chin Wei Eugene Koh, **Lei Wang**, Haizhou Li, and Eng Siong Chng, "Temporal discrete cosine transform: Towards longer term temporal features for speaker verification," in *Proceedings of 5th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, COLIPS, Q. Huo et al.(Eds.), pp. 547-558, 13-16 December 2006, Singapore.

(vi) K. A. Lee, H. Sun, R. Tong, B. Ma, M. Dong, C. You, D. Zhu, C. W. E. Koh, **L. Wang**, T. Kinnunen, E. S. Chng, and H. Li, "The IIR submission to CSLP 2006 speaker recognition evaluation," in *Proceedings of 5th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, LNAI 4274, Q. Huo et al.(Eds.), pp.494-505, 13-16 December 2006, Singapore.

References

[1] N. Dimitrova, H.-J. Zhang, B. Shahraray, I. Sezan, T. Huang, and A. Zakhor, "Applications of video-content analysis and retrieval," *IEEE Multimedia*, vol. 9, no. 3, pp. 42–55, July-Sept. 2002.

[2] S. Furui, T. Kikuchi, Y. Shinnaka, and C. Hori, "Speech-to-text and speech-to-speech summarization of spontaneous speech," *IEEE Trans. Speech and Audio Processing*, vol. 12, no. 4, pp. 401–408, July 2004.

[3] C. Herley, "ARGOS: Automatically extracting repeating objects from multimedia streams," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 115–129, Feb. 2006.

[4] W. Chai, "Semantic segmentation and summarization of music: methods based on tonality and recurrent structure," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 124–132, Mar. 2006.

[5] J. Makhoul, F. Kubala, T. Leek, D. Liu, L. Nguyen, R. Schwartz, and A. Srivastava, "Speech and language technologies for audio indexing and retrieval," *Proc. IEEE*, vol. 88, no. 8, pp. 1338–1353, Aug. 2000.

[6] A. Park, T. J. Hazen, and J. R. Glass, "Automatic processing of audio lectures for information retrieval: Vocabulary selection and language modeling," in *Proc. ICASSP '05*, Philadelphia, USA, Mar. 2005, vol. 1, pp. 497–500.

[7] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, "The Rich Transcription 2006 Spring Meeting Recognition Evaluation," in *Machine Learning for Multimodal Interaction*, pp. 309–322. May 2006.

[8] Z.-Y. Zhou, P. Yu, C. Chelba, and F. Seide, "Towards spoken-document retrieval for the internet: lattice indexing for large-scale web-search architectures," in *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the ACL*, New York, June 2006, pp. 415–422.

[9] D. P. W. Ellis and K. Lee, "Minimal-impact audio-based personal archives," in *Proc. ACM workshop on Continuous Archival and Retrieval of Personal Experiences '04*, New York, NY, USA, 2004, pp. 39–47.

[10] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE Multimedia*, vol. 3, no. 3, pp. 27–36, July-Sept. 1996.

[11] C. Hori and S. Furui, "A new approach to automatic speech summarization," *IEEE Trans. Multimedia*, vol. 5, no. 3, pp. 368–378, Sept. 2003.

[12] J. Foote, "Visualizing music and audio using self-similarity," in *Proc. ACM MM '99*, 1999, pp. 77–80.

[13] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, Jan. 2008.

[14] *http://www.music-ir.org/mirex/wiki/mirex_home/.*

[15] *http://hlt.i2r.a-star.edu.sg/starchallenge.*

[16] M. A. Bartsch and G. H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics '01*, Oct. 2001, pp. 15–18.

[17] J. Wolfe, "Speech and music, acoustics and coding, and what music might be 'for'," in *Proc. 7th International Conference on Music Perception and Cognition*, 2002, pp. 10–13.

[18] C.-H. Lee, F. K. Soong, and B.-H. Juang, "A segment model based approach to speech recognition," in *Proc. ICASSP '88*, New York, NY, 1988, pp. 501–504.

[19] P. W. Jusczyk and R. N. Aslin, "Infants' detection of the sound patterns of words in fluent speech," *Cognitive Psychology*, vol. 29, no. 1, pp. 1–23, 1995.

[20] A. Velivelli, C. Zhai, and T. S. Huang, "Audio segment retrieval using a short duration example query," in *Proc. ICME '04*, Taipei, Taiwan, June 2004, pp. 1603–1606.

[21] *https://www.google.com/voice*.

[22] J. Herre, E. Allamanche, and O. Hellmuth, "Robust matching of audio signals using spectral flatness features," in *Proc. IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics '01*, New Paltz, New York, Oct. 2001, pp. 127–130.

[23] D. Abberley, S. Renals, and G. Cook, "Retrieval of broadcast news documents with the THISL system," in *Proc. ICASSP '98*, Seattle, WA, May 1998, pp. 3781–3784.

[24] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *Proc. HLT/NAACL 2004*, Boston, MA, USA, May 2004.

[25] P. Yu, K. Chen, C. Ma, and F. Seide, "Vocabulary-independent indexing of spontaneous speech," *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, pp. 635–643, Sept. 2005.

[26] J. S. Garofolo, E. M. Voorhees, V. M. Stanford, and K. S. Jones, "TREC-6 1997 Spoken Document Retrieval Track Overview and Results," in *Proc. 1997 TREC-6 Conference*, Gaithersburg, MD, Nov. 1997, pp. 83–91.

[27] M. G. Brown, J. T. Foote, G. J. F. Jones, K. Sparck Jones, and S. J. Young, "Automatic content-based retrieval of broadcast news," in *Proc. ACM Multimedia '95*, San Francisco, California, United States, 1995, pp. 35–43.

[28] J.-M. V. Thong, P. J. Moreno, B. Logan, B. Fidler, K. Maffey, and M. Moores, "SPEECHBOT: An experimental speech-based search engine for multimedia content in the web," Tech. Rep. CRL 2001/06, Cambridge Research Laboratory, July 2001.

[29] S. Kiranyaz, A. F. Qureshi, and M. Gabbouj, "A generic audio classification and segmentation approach for multimedia indexing and retrieval," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 1062–1081, May 2006.

[30] R. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H. Cai, "A flexible framework for key audio effects detection and auditory context inference," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 1026–1039, May 2006.

[31] P. Piamsa-Nga et al., "Content-based audio retrieval using a generalized algorithm," in *Advances in Intelligent Systems: Concepts, Tools, and Applications*, 1998, pp. 231–242.

[32] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, Sept. 1995.

[33] L. Lu, M. Wang, and H.-J. Zhang, "Repeating pattern discovery and structure analysis from acoustic music data," in *Proc. ACM MIR '04*, New York, NY, USA, 2004, pp. 275–282.

[34] *http://www-nlpir.nist.gov/projects/trecvid*.

[35] B. Pardo, "Finding structure in audio for music information retrieval," *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 126–132, May 2006.

[36] B.-H. Juang and S. Furui, "Automatic recognition and understanding of spoken language - a first step toward natural human-machine communication," *Proc. IEEE*, vol. 88, no. 8, pp. 1142–1165, Aug. 2000.

[37] C.-L. Huang and C.-H. Wu, "Spoken document retrieval using multilevel knowledge and semantic verification," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2551–2560, Nov. 2007.

[38] K. Zechner, "Automatic summarization of open-domain multiparty dialogues in diverse genres," *Computational Linguistics*, vol. 28, no. 4, pp. 447–485, Dec. 2002.

[39] X. Zhu and G. Penn, "Summarization of spontaneous conversations," in *Proc. Interspeech '07*, Antwerp, Belgium, Aug. 2007, pp. 1531–1534.

[40] N. Morgan, D. Baron, J. Edwards, D. Ellis, D. Gelbart, A. Janin, T. Pfau, E. Shriberg, and A. Stolcke, "The meeting project at ICSI," in *Proc. HLT '01*, San Diego, 2001, pp. 1–7.

[41] Y. Fujii, N. Kitaoka, and S. Nakagawa, "Automatic extraction of cue phrases for important sentences in lecture speech and automatic lecture speech summarizaiton," in *Proc. Interspeech '07*, Antwerp, Belgium, Aug. 2007, pp. 2801–2804.

[42] T. Shinozaki, C. Hori, and S. Furui, "Towards automatic transcription of spontaneous presentations," in *Proc. Eurospeech '01*, Aalborg, Denmark, Sept. 2001, pp. 491–494.

[43] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: Musical information retrieval in an audio database," in *Proc. ACM MM '95*, San Francisco, California, USA, Nov. 1995, pp. 231–236.

[44] L. Lu, H. You, and H.-J. Zhang, "A new approach to query by humming in music retrieval," in *Proc. ICME '01*, Tokyo, Japan, Aug. 2001, pp. 776–779.

[45] T. De Mulder, J.-P. Martens, S. Pauws, F. Vignoli, M. Lesaffre, M. Leman, B. De Baets, and H. De Meyer, "Factors affecting music retrieval in query-by-melody," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 728–739, Aug. 2006.

[46] R. B. Dannenberg and N. Hu, "Pattern discovery techniques for music audio," *Journal of New Music Research*, vol. 32, no. 2, pp. 153–163, June 2003.

[47] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1783–1794, Sept. 2006.

[48] R. Valenza, T. Robinson, M. Hickey, and R. Tucker, "Summarisation of spoken audio through information extraction," in *Proc. ESCA workshop: Accessing information in spoken audio*, 1999, pp. 111–116.

[49] R. Jin and A. G. Hauptmann, "Title generation for spoken broadcast news using a training corpus," in *Proc. ICSLP '00*, Beijing, China, Oct. 2000.

[50] S. R. Maskey and J. Hirschberg, "Automatic summarization of broadcast news using structural features," in *Proc. Eurospeech '03*, Geneva, Switzerland, Sept. 2003, pp. 1173–1176.

[51] G. Murray and S. Renals, "Towards online speech summarization," in *Proc. Interspeech '07*, Antwerp, Belgium, Aug. 2007, pp. 2785–2788.

[52] K. Koumpis and S. Renals, "Automatic summarization of voicemail messages using lexical and prosodic features," *ACM Trans. Speech Lang. Process.*, vol. 2, no. 1, pp. 1–24, Feb. 2005.

[53] S. Furui, "Recent progress in spontaneous speech recognition and understanding," in *Proc. IEEE Workshop on Multimedia Signal Processing '02*, Dec. 2002, pp. 253–258.

[54] G. Williams and S. Renals, "Confidence measures for hybrid HMM/ANN speech recognition," in *Proc. Eurospeech '97*, Rhodes, Greece, Sept. 1997, pp. 1955–1958.

[55] C.-L. Huang, C.-H. Hsieh, and C.-H. Wu, "Spoken document summarization using acoustic, prosodic and semantic information," in *Proc. ICME '05*, July 2005.

[56] K. Ohtake, K. Yamamoto, Y. Toma, S. Sado, S. Masuyama, and S. Nakagawa, "Newscast speech summarization via sentence shortening based on prosodic features," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003, pp. 167–170.

[57] A. Inoue, T. Mikami, and Y. Yamashita, "Improvement of speech summarization using prosodic information," in *Proc. Speech Prosody*, Japan, 2004.

[58] J. Hirschberg, "Communication and prosody: Functional aspects of prosody," *Speech Communication*, vol. 36, no. 1-2, pp. 31–43, Jan. 2002.

[59] *http://www.acorns-project.org/.*

[60] X. Zhu, G. Penn, and F. Rudzicz, "Summarizing multiple spoken documents: finding evidence from untranscribed audio," in *Proc. the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Singapore, Aug. 2009, pp. 549–557.

[61] I. Szoke, P. Schwarz, and P. Matejka, "Comparion of keyword spotting approaches for informal continuous speech," in *Proc. Eurospeech '05*, Lisbon, Portugal, Sept. 2005, pp. 633–636.

[62] S. E. Johnson and P. C. Woodland, "A method for direct audio search with applications to indexing and retrieval," in *Proc. ICASSP '00*, Istanbul, Turkey, June 2000, pp. 1427–1430.

[63] B. Logan, P. Moreno, and O. Deshmukh, "Word and sub-word indexing approaches for reducing the effects of OOV queries on spoken audio," in *Proc. Human Language Technology Conference '02*, San Diego, CA, 2002.

[64] Z. Liu and Q. Huang, "Content-based indexing and retrieval-by-example in audio," in *Proc. ICME '00*, New York City, NY, USA, July-Aug. 2000, pp. 877–880.

[65] Y. Rui, A. Gupta, and A. Acero, "Automatically extracting highlights for TV baseball programs," in *Proc. ACM MM '00*, 2000, pp. 105–115.

[66] Z. Xiong, R. Radhakrishnan, A. Divakaran, and T. S. Huang, "Audio events detection based highlights extraction from baseball golf and soccer games in a unified network," in *Proc. ICASSP '03*, Hong Kong, China, Apr. 2003, vol. 5, pp. 632–635.

[67] B. Zhang, W. Dou, and L. Chen, "Ball hit detection in table tennis games based on audio analysis," in *Proc. ICPR '06*, 2006, vol. 3, pp. 220–223.

[68] S. Nepal, U. Srinivasan, and G. Reynolds, "Automatic detection of 'Goal' segments in basketball videos," in *Proc. ACM MM '01*, Ottawa, Canada, 2001, pp. 261–269.

[69] J. Wang, *Content-based sports video analysis and composition*, Ph.D. thesis, Nanyang Technological University, 2006.

[70] C. M. Taskiran, Z. Pizlo, A. Amir, D. Ponceleon, and E. J. Delp, "Automated video program summarization using speech transcripts," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 775–791, Aug. 2006.

[71] Y. Li, S.-H. Lee, C.-H. Yeh, and C.-C. J. Kuo, "Techniques for movie content analysis and skimming: tutorial and overview on video abstraction techniques," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 79–89, Mar. 2006.

[72] M. Xu, L.-T. Chia, and J. Jin, "Affective content analysis in comedy and horror videos by audio emotional event detection," in *Proc. ICME '05*, Amsterdam, Netherlands, July 2005.

[73] H.-W. Chen, J.-H. Kuo, W.-T. Chu, and J.-L. Wu, "Action movies segmentation and summarization based on tempo analysis," in *Proc. ACM SIGMM international workshop on Multimedia information retrieval '04*, New York, NY, USA, 2004, pp. 251–258.

[74] S. Moncrieff, C. Dorai, and S. Venkatesh, "Affect computing in film through sound energy dynamics," in *Proc. ACM MM '01*, Ottawa, Canada, 2001, pp. 525–527.

[75] Z. Rasheed and M. Shah, "Movie genre classification by exploiting audio-visual features of previews," in *Proc. ICPR '02*, Aug. 2002, vol. 2, pp. 1086–1089.

[76] W.-H. Cheng, W.-T. Chu, and J.-L. Wu, "Semantic context detection based on hierarchical audio models," in *Proc. ACM MIR '03*, Berkeley, California, 2003, pp. 109–115.

[77] Y. Wang, Z. Liu, and J.-C. Huang, "Multimedia content analysis -using both audio and visual clues," *IEEE Signal Processing Magazine*, vol. 17, no. 6, pp. 12–36, Nov. 2000.

[78] L. Lu, H.-J. Zhang, and H. Jiang, "Content analysis for audio classification and segmentation," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 7, pp. 504–516, Oct. 2002.

[79] L. Rabiner and B.-H. Juang, *Fundamental of Speech Recognition*, Prentice-Hall International, Inc., 1993.

[80] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[81] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A guide to theory, algorithm and system development*, Prentice Hall PTR, 2001.

[82] J. C. Segura, C. Benitez, A. de la Torre, A. J. Rubio, and J. Ramirez, "Cepstral domain segmental nonlinear feature transformations for robust speech recognition," *IEEE Signal Processing letters*, vol. 11, no. 5, pp. 517–520, May 2004.

[83] T. Kinnunen, C. W. E. Koh, L. Wang, H. Li, , and E. S. Chng, "Temporal discrete cosine transform: Towards longer term temporal features for speaker verification," in *Proceedings of 5th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Singapore, Dec. 2006, pp. 547–558.

[84] T. L. Bailey and C. Elkan, "Unsupervised learning of multiple motifs in biopolymers using expectation maximization," *Machine Learning*, vol. 21, no. 1-2, pp. 51–80, Oct. 1995.

[85] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert, "Approaches to the automatic discovery of patterns in biosequences," *Journal of Computational Biology*, vol. 5, no. 2, pp. 279–305, 1998.

[86] J.-L. Hsu, C.-C. Liu, and A. L. P. Chen, "Discovering nontrivial repeating patterns in music data," *IEEE Trans. Multimedia*, vol. 3, no. 3, pp. 311–325, Sept. 2001.

[87] M. M. Gaber, A. Z., and S. Krishnaswamy, "Mining data streams: a review," *ACM SIGMOD Record*, vol. 34, no. 2, pp. 18–26, 2005.

[88] E. M. McCreight, "A space-economical suffix tree construction algorithm," *Journal of the ACM*, vol. 23, no. 2, pp. 262–272, 1976.

[89] D. He, "Using suffix tree to discover complex repetitive patterns in DNA sequences," in *Proc. of EMBS '06*, Aug. 2006, pp. 3474–3477.

[90] A. Chattaraja and L. Paridab, "An inexact-suffix-tree-based algorithm for detecting extensible patterns," *Theoretical Computer Science*, vol. 335, no. 1, pp. 3–14, May 2005.

[91] M. Vingron and P. Argos, "Motif recognition and alignment for many sequences by comparison of dot-matrices," *Journal of Molecular Biology*, vol. 218, no. 1, pp. 33–43, Jan. 1991.

[92] M.-F. Sagot, A. Viari, and H. Soldano, "Multiple sequence comparison: A peptide matching approach," in *Proc. 6th Annual Symposium on Combinatorial Pattern Matching '95*, Espoo, Finland, July 1995.

[93] I. Karydis, A. Nanopoulos, and Y. Manolopoulos, "Finding maximum-length repeating patterns in music databases," *Multimedia Tools and Applications*, vol. 32, no. 1, pp. 49–71, Jan. 2007.

[94] P. Nowell and R. K. Moore, "The application of dynamic programming techniques to non-word based topic spotting," in *Proc. Eurospeech '95*, Madrid, Spain, Sept. 1995, pp. 1355–1358.

[95] G. Aimetti, "Modelling early language acquisition skills: Towards a general statistical learning mechanism," in *Proc. the EACL 2009 Student Research Workshop*, Athens, Greece, Apr. 2009, pp. 1–9.

[96] J. Pevsner, *Bioinformatics and Functional Genomics*, John Wiley & Sons, Inc., 2003.

[97] *http://blast.ncbi.nlm.nih.gov/Blast.cgi*.

[98] J. W. Picone, "Signal modeling techniques in speech recognition," *Proc. IEEE*, vol. 81, no. 9, pp. 1215–1247, Sept. 1993.

[99] L. Xie, *Unsupervised pattern discovery for multimedia sequences*, Ph.D. thesis, Columbia University, 2005.

[100] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multidimensional time-series," *The VLDB Journal*, vol. 15, no. 1, pp. 1–20, Jan. 2006.

[101] Y. Tanaka, K. Iwamoto, and K. Uehara, "Discovery of time-series motif from multi-dimensional data based on MDL principle," *Machine Learning*, vol. 58, no. 2-3, pp. 269–300, Mar. 2005.

[102] D. Minnen, T. Starner, I. Essa, and C. Isbell, "Discovering characteristic actions from on-body sensor data," in *Proc. the 10th IEEE International Symposium on Wearable Computers*, Oct. 2006, pp. 11–18.

[103] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. ACM SIGMOD workshop on Research issues in data mining and knowledge discovery '03*, San Diego, California, 2003, pp. 2–11.

[104] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons Inc., 2nd edition, 2001.

[105] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, Sept. 1999.

[106] L. ten Bosch and B. Cranen, "A computational model for unsupervised word discovery," in *Proc. Interspeech 2007*, Antwerp, Belgium, Aug. 2007, pp. 1481–1484.

[107] H. Van hamme, "HAC-models: a novel approach to continuous speech recognition," in *Proc. Interspeech 2008*, Brisbane, Australia, Sept. 2008, pp. 2554–2557.

[108] O. J. Räsänen, U. K. Laine, and T. Altosaar, "A noise robust method for pattern discovery in quantized time series: the concept matrix approach," in *Proc. Interspeech 2009*, Brighton, UK, Sept. 2009, pp. 3035–3038.

[109] O. J. Räsänen, U. K. Laine, and T. Altosaar, "Self-learning vector quantization for pattern discovery from speech," in *Proc. Interspeech 2009*, Brighton, UK, Sept. 2009, pp. 852–855.

[110] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 68–86, Jan. 1971.

[111] C. J. V. Rijsbergen, *Information Retrieval*, London: Butterworths, 2nd edition, 1979.

[112] E. Keogh, J. Lin, and W. Truppel, "Clustering of time series subsequences is meaningless: implications for previous and future research," in *Proc. ICDM '03*, Nov. 2003, pp. 115–122.

[113] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel, "Finding motifs in time series," in *Proc. the Second Workshop on Temporal Data Mining*, Edmonton, Alberta, Canada, July 2002.

[114] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[115] B. Logan and S. Chu, "Music summarization using key phrases," in *Proc. ICASSP '00*, Istanbul, Turkey, June 2000, vol. 2, pp. 749–752.

[116] J.-J. Aucouturier and M. Sandler, "Segmentation of musical signals using hidden Markov models," in *Proc. the Audio Engineering Society 110th Convention*, Amsterdam, Netherlands, May 2001.

[117] S. Abdallah, K. Noland, M. Sandler, M. Casey, and C. Rhodes, "Theory and evaluation of a bayesian music structure extractor," in *Proc. the 6th ISMIR Conference*, London, UK, 2005, pp. 420–425.

[118] S. Young et al., *The HTK Book (for HTK Version 3.2.1)*, Cambridege University Engineering Department, 2002.

[119] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer, and M. A. Picheny, "A method for the construction of acoustic markov models for words," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 443–452, Oct. 1993.

[120] J. Takami and S. Sagayama, "A successive state splitting algorithm for efficient allophone modeling," in *Proc. ICASSP '92*, San Francisco, CA, Mar. 1992, pp. 573–576.

[121] H. Singer and M. Ostendorf, "Maximum likelihood successive state splitting," in *Proc. ICASSP '96*, Atlanta, GA, May 1996, pp. 601–604.

[122] B. Varadarajan, S. Khudanpur, and E. Dupoux, "Unsupervised learning of acoustic sub-word units," in *Proceedings of ACL-08: HLT*, Columbus, Ohio, USA, 2008, pp. 165–168.

[123] K. K. Paliwal, "Lexicon-building methods for an acoustic sub-word based speech recognizer," in *Proc. ICASSP '90*, Alburquerque, NM, Apr. 1990, pp. 729–732.

[124] H. Li, B. Ma, and C.-H. Lee, "A vector space modeling approach to spoken language identification," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 271–284, Jan. 2007.

[125] M. Ostendorf, "Moving beyond the 'beads-on-a-string' model of speech," in *Proc. ASRU 1999*, Keystone, Colorado, 1999, pp. 79–84.

[126] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 96–104, Feb. 2005.

[127] T. Zhang and R. Samadani, "Automatic generation of music thumbnails," in *Proc. ICME '07*, 2007, pp. 228–231.

[128] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Proc. ICME '00*, July-Aug. 2000, pp. 452–455.

[129] Y. Shiu, H. Jeong, and C.-C. J. Kuo, "Similarity matrix processing for music structure analysis," in *Proc. ACM workshop on Audio and music computing multimedia '06*, Santa Barbara, California, USA, 2006, pp. 69–76.

[130] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

[131] L. R. Rabiner and S. E. Levinson, "Isolated and connected word recognition–theory and selected applications," *IEEE Trans. Communications*, vol. 29, no. 5, pp. 621–659, May 1981.

[132] *http://labrosa.ee.columbia.edu/matlab/dtw/*.

[133] M. Cevik, F. Weng, and C.-H. Lee, "Detection of repetitions in spontaneous speech in dialogue sessions," in *Proc. Interspeech 2008*, Brisbane, Australia, Sept. 2008, pp. 471–474.

[134] T. Oates, "PERUSE: An unsupervised algorithm for finding recurring patterns in time series," in *Proc. ICDM '02*, Dec. 2002, pp. 330–337.

[135] A. Muscariello, G. Gravier, and F. Bimbot, "Audio keyword extraction by unsupervised word discovery," in *Proc. Interspeech 2009*, Brighton, UK, Sept. 2009, pp. 2843–2846.

[136] C. Wang, J. Li, and S. Shi, "N-gram inverted index structures on music data for theme mining and content-based information retrieval," *Pattern Recognition Letters*, vol. 27, no. 5, pp. 492–503, Apr. 2006.

[137] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, "A multiresolution symbolic representation of time series," in *Proc. ICDE '05*, Apr. 2005, pp. 668–679.

[138] Eamonn Keogh, *The UCR Time Series Data Mining Archive [http://www.cs.ucr.edu/∼eamonn/TSDMA/index.html]*, University of California - Computer Science & Engineering Department, Riverside, CA, 2006.

[139] Bill Chiu, Eamonn Keogh, and Stefano Lonardi, "Probabilistic discovery of time series motifs," in *KDD '03: Proc. the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, D.C., Aug. 2003, pp. 493–498.

[140] Y. Gong, "Speech recognition in noisy environments: A survey," *Speech Communication*, vol. 16, no. 3, pp. 261–291, 1995.

[141] S. T. Piantadosi, H. Tily, and E. Gibson, "Word lengths are optimized for efficient communication," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 108, no. 9, pp. 3526–3529, 2011.

[142] L. R. Rabiner and S. E. Levinson, "A speaker-independent, syntax-directed, connected word recognition system based on hidden markov models and level building," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 33, no. 3, pp. 561–573, June 1985.

[143] J. R. Saffran, R. N. Aslin, and E. L. Newport, "Statistical learning by 8-month-old infants," *Science*, vol. 274, no. 5294, pp. 1926–1932, 1996.

[144] P. Perruchet and S. Desaulty, "A role for backward transitional probabilities in word segmentation?," *Memory & Cognition*, vol. 36, no. 7, pp. 1299–1305, 2008.

[145] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational Linguistics*, vol. 16, no. 1, pp. 22–29, Mar. 1990.

[146] K. Church, W. Gale, P. Hanks, and D. Hindle, "Using statistics in lexical analysis," in *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pp. 115–164. Erlbaum, 1991.

[147] *http://www.socsci.uci.edu/l̃pearl/courses/psych215L_2010fall/index.html*.

[148] M. R. Brent, "An efficient, probabilistically sound algorithm for segmentation and word discovery," *Machine Learning*, vol. 34, pp. 71–105, 1999.

[149] W. Shen and D. Reynolds, "Improving phonotactic language recognition with acoustic adaptation," in *Proc. Interspeech '07*, Antwerp, Belgium, Aug. 2007, pp. 358–361.

[150] J. R. Rohlicek, W. Russell, S. Roukod, and H. Gish, "Continuous hidden Markov model for speaker independent word spotting," in *Proc. ICASSP '89*, Glasgow, Scotland, May 1989, pp. 627–430.

[151] A. Manos and V. Zue, "A segment-based wordspotter using phonetic filler models," in *Proc. ICASSP '97*, Munich, Germany, Apr. 1997, pp. 899–902.

[152] 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 Trans. Acoustics, Speech and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, Nov. 1990.

[153] *http://labs.google.com/gaudi*.

[154] J. Foote, "An overview of audio information retrieval," *Multimedia Systems*, vol. 7, no. 1, pp. 2–10, 1999.

[155] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, July 2002.

[156] E. Scheirer and M. Slanry, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. ICASSP '97*, Munich, Germany, Apr. 1997, pp. 1331–1334.

[157] G. Lu, "Indexing and retrieval of audio: A survey," *Multimedia Tools and Applications Journal*, vol. 15, no. 3, pp. 269–290, Dec. 2001.

[158] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on gaussian posteriorgrams," in *Proc. ASRU 2009*, 2009.

[159] V. Gupta, J. Ajmera, A. Kumar, and A. Verma, "A language independent approach to audio search," in *Proc. Interspeech 2011*, Florence, Italy, Aug. 2011, pp. 1125–1128.

[160] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, Apr. 2009.

[161] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. ACM SIGIR 2007 on Research and development in information retrieval*, New York, NY, USA, 2007, pp. 615–622.

[162] T. K. Chia, K. C. Sim, H. Li, and H. T. Ng, "A lattice-based approach to query-by-example spoken document retrieval," in *Proc. ACM SIGIR '08 on research and development in information retrieval*, 2008, pp. 363–370.

[163] B. Matthews, U. Chaudhari, and B. Ramabhadran, "Fast audio search using vector space modelling," in *Proc. ASRU 2007*, 2007.

[164] D. A. James and S. J. Young, "A fast lattice-based approach to vocabulary independent wordspotting," in *Proc. ICASSP '94*, Adelaide, Australia, Apr. 1994, vol. 1, pp. 377–380.

[165] C.-H. Lee and L. R. Rabiner, "A frame-synchronous network search algorithm for connected word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 37, no. 11, pp. 1649–1658, Nov. 1989.

[166] R. Singh, B. Raj, and R. M. Stern, "Automatic generation of subword units for speech recognition systems," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 2, pp. 89–99, Feb. 2002.

[167] N. Parihar and J. Picone, *Aurora working group: DSR front end LVCSR evaluation AU/384/02*, Institute for Signal and Infomation Processing, Mississippi State Univ., MS, Tech. Rep., 2002.

[168] G. Salton, *The SMART Retrieval System*, Prentice-Hall, Inc., 1971.

[169] I. Mani and M. T. Maybury, *Advances in Automatic Text Summarization*, Cambridge, MA:MIT Press, 1999.

[170] N. C. Maddage, H. Li, and M. S. Kankanhalli, "Music structure based vector space retrieval," in *Proc. ACM SIGIR '06*, Seattle, Washington, USA, Aug. 2006, pp. 67–74.

[171] K. Chen, S. Gao, Y. Zhu, and Q. Sun, "Music genres classification using text categorization method," in *Proc. IEEE Workshop on Multimedia Signal Processing '06*, Oct. 2006, pp. 221–224.

[172] J.L. Gauvain, A. Messaoudi, and H. Schwenk, "Language recognition using phone lattices," in *Proc. ICSLP '04*, Jeju, South Korea, Oct. 2004.

[173] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proc. ICSLP '02*, Denver, USA, Sept. 2002.

[174] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley Longman, 1999.

[175] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 254–272, Apr. 1981.

[176] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, pp. 133–147, 1998.

[177] X. Xiao, E. S. Chng, and H. Li, "Normalization of speech modulation spectra for robust speech recognition," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1662–1674, Nov. 2008.

[178] D. Pearce and H.-G. Hirsch, "The AURORA experimental framework for the performance evaluation of speech recogntion systems under noisy conditions," in *Proc. ICSLP '00*, Beijing, China, Oct. 2000, pp. 29–32.

[179] J. Serra, E. Gomez, and P. Herrera, "Transposing chroma representations to a common key," in *Proc. IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, 2008, pp. 45–48.

[180] G. Aimetti, R. K. Moore, L. ten Bosch, O. J. Räsänen, and U. K. Laine, "Discovering keywords from cross-modal input: ecological vs. engineering methods for enhancing acoustic repetitions," in *Proc. Interspeech 2009*, Brighton, UK, Sept. 2009, pp. 1171–1174.

[181] V. Stouten, K. Demuynck, and H. Van hamme, "Discovering phone patterns in spoken utterances by non-negative matrix factorization," *IEEE Signal Processing letters*, vol. 15, pp. 131–134, 2008.

[182] J. F. Werker and H. H. Yeung, "Infant speech perception bootstraps word learning," *TRENDS in Cognitive Sciences*, vol. 9, no. 11, pp. 519–527, 2005.