MICA

Music Identifying and Classifying Application

by

Jiaxi Li
Weichen Zhang
Yuan Fang

COMP5425
Multimedia Retrieval
Project Final Report

Project Duration: 6 Weeks
Project Start Date: 13/04/2016
Project End Date: 25/05/2016

Contents

1 Introduction

1.1. Motivation

The rapid growth in technology is greater than ever nowadays. In this context, music is getting more and more popular, and an increasing number of music has been infiltrated into lives of people and influencing them in different forms. With the development of the network and digitization technology, people are surrounded by music at all times. However, people may enjoy particular music without knowing what this song is or what it consists of. Therefore, identifying songs of interests becomes an executable and reasonable motivation for developing applications to suit this objective.

1.2. Project

Due the over-variety of music, this project is mainly focusing on Classical Music. There are three major functionality in this deliverable: match incoming music (through files and recordings) with our database, classify instrument played within music (through files and recordings), and rate the music played by yourself.

1.2.1. Searching

Similar to *Shazam*[1], we capture a piece of music clip from either microphone or unrecognized music file, and produce the best matching result in our database.

1.2.2. Instrument Classification

Beside the music name, what instrument played within music interests people. This project provides the ability of identifying instruments through either an unclassified file or microphone on you computer.

1.2.3. Rate Yourself

What more than matching and classifying is having the fun by practicing your own skills. Select the favorite song in the database, and upload personal plays (or through microphone). Then, score will be showing up, and parts that are not played well is pointed out to users on the screen.

[1] `http://www.shazam.com`

2

Project Specification

2.1. Group Work

The three major functionality, searching, instrument classifying and rating, are divided and assigned to Weichen Zhang, Jiaxi Li, Yuan Fang respectively. While everyone in the group is working individually with his part, GUI integration is conducted by all of the members regards to corresponding functionality.

2.2. Working Environment

2.2.1. Source Code

Major functionality is implemented in pure $Python$. GUI is built with python interface dependencies $pygame$[1] and $Tkinter$[2].

2.3. Related Work

Burnett et al. mentioned in [2], "an audio fingerprint is a condensed digital summary, deterministically generated from an audio signal, and then can be used to identify audio samples or locate similar items in audio database in a very short time". With this determination of technique, Kim et al. then has proposed an algorithm for extracting music fingerprints directly from the audio signal in *IEEE International Conference* in 2008[7]. Moreover, she had concluded that this proposed algorithm outperformed the conventional state-of-art identification system. After that, there are various methodologies proposed to augment precision and reliability of audio fingerprints.

In 2013, Lee et al. [8] pointed out the inefficiency in Haitsma's fingerprint system[6]. The audio fingerprinting system provided by Haitsma uses a lookup table to identify the candidate songs in the database, which contains the sub-fingerprints of songs, and searches the candidates to find a song whose bit error rate is the lowest. However, the communication between system and database is extremely frequent which can result in the lost of sub-fingerprint duo to heavily degraded input signal. Lee, on the other had, solves this difficulties by reducing the number of database accesses using an partitioned method on each song.

In the last few years, audio fingerprint has been successfully applied and developed in repeating pattern detection[3], speech recognition [13], and audio content recognition[5].

[1] `http://www.pygame.org/`
[2] `https://docs.python.org/2/library/tkinter.htm`

For the purpose of recognizing instrument, Li et al. compared few approaches of extracting instrument within an audio, including convolutional neural networks (CNN) and Mel-frequency cepstral coefficients (MFCCs) [9].

MFCC computation technique is derived from a type of cepstral representation of the audio clip. It utilizes discrete cosine transform (DCT) for decorrelating log energies of filter bank output [12]. Then resulting log energies and filter banks can be used to calculating a set of coefficients that can best represent an audio. In fact, Bhalke et al. published a paper in *Digital Signal Processing Journal* in 2013 [1] and described the combination of higher order moments and MFCC improves recognition accuracy by more than 8% comparing to convention methodologies.

2.4. Problem Analysis

2.4.1. Feature Extraction

The first step of audio recognition is to extract features. Mel-Frequency Cepstral Coefficents (MFCCs) is one of the most widely used technique for select features in speeches and speakers. As Ganchev et al. mentioned in [4], different implementations may result in different performance. Therefore, the implementation of MFCCs in this project is carried out on our own in following steps:

1. Frame signal into short frames.
2. Calculate power spectrum for each frame.
3. Generate and apply Mel filter banks and energies to power spectrum.
4. Compute logarithm of all filter bank energies.
5. Transform resulting log filter bank energies to DCT[3].
6. Keep 13 coefficients, and discard others.

Note that above steps are high level introduction for implementation, a more detailed description will be given in Section 3.1.

2.4.2. Semi-Real-Time System

As mentioned in the Section 1.2, the objective of project is to build an application that can retrieve relevant audio in database to match an unknown source and identify instruments played in it. In order to achieve this goal, this application has to perform selected tasks within a reasonable time. Similar to *Shazam*, our application allows user to record a clip of audio and then perform the selected task. Besides, tasks can also be executed on a selected audio file by browsing file system.

[3] Discrete cosine transform

3

Implementation

3.1. Instrument Classification

3.1.1. MFCCs

As the standards mentioned in [11], the signal of audio is framed in to 25 ms and number frame step is 10 ms which allows some overlap between frames. That is, for an audio whose sample rate is 16 kHz, it has 400 samples after the framing process.

Before power spectrum is computed, Discrete Fourier Transform (DFT) is applied to each frames as follow:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{\frac{-j2\pi kn}{N}} \qquad k = 1,2,...,K \tag{3.1}$$

where $s(n)$ is time domain signal and $s_i(n)$ represents segmented frames. $h(n)$ indicates a N sample long analysis window, and K denotes length of DFT. Therefore, power spectrum $P_i(k)$ for each frame is given by:

$$P_i(k) = \frac{1}{N}(S_i(k))^2 \tag{3.2}$$

Mel filter bank is generated using 26 triangular filters (standard), and then resulting filter bank is applied to each power spectrum we had calculated above and filter bank energies are computed through multiplying each filter bank with power spectrum, then add up the coefficients. Once this is calculated, each filter bank will be left with 26 numbers indicating how much energy is in it. The example results of power spectrum and filter banks are illustrated in Figure 3.1. The essential steps for calculating filter bank can be found in [10].

Next step is doing logarithm on 26 filter bank energies. Once log energies are compute, Discrete Cosine Transform (DCT) is applied on resulting 26 energies to generate same number of cepstral coefficients. At last, in this project, only 14 out of 26 lowest coefficients are kept for the purpose of classifying.

As Li pointed out in [9], after MFCCs of each song in database are computed, we train our data with Random Forest which is a widely used classification technique implemented in $sci\text{-}kit\ learn$[1].

3.1.2. GUI

$Insert\_Shots$

[1] `http://scikit-learn.org`

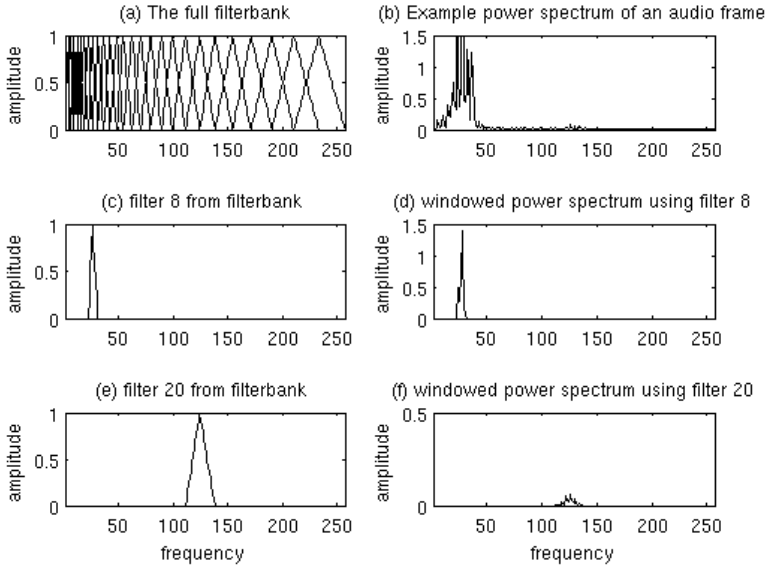

Figure 3.1: Graphical illustration of power spectrum and filter banks.

3.2. Searching

Audio fingerprint is generated using a audio fingerprinting and recognition algorithm implemented in *dejavu*[2], which can memorize audios by listening to it once and fingerprinting it. Then using resulting fingerprints to match input audio against our database (using *mysql*[3]).

3.2.1. GUI

Insert_Shots

3.3. Rating Plays

3.3.1. GUI

Insert_Shots

[2] https://github.com/worldveil/dejavu
[3] https://www.mysql.com

4

Discussion

4.1. Contribution

There are three major contributions have been made within this project.

First of all, not like *c#*, $python$ is a shell language which contains very few GUI abilities. In a short amount of time, an executable and working $Python$ GUI has been built for integrating with functionality described in Section 1.2.

Secondly, referring to Ganchev et al. [4], a different tweak on implementing MFCCs is written from scratch, which suits the purpose of classifying instruments in classical music. Besides, a recreation of combining MFCC with Random Forest on instrument recognition, which mentioned in [9], has been successfully produced.

Last but not least, there are very few music applications are focusing on classical music. Under this circumstance, this project provides an integration of audio processing on this kind of area.

4.2. Reflection and Future Work

While we were developing objectives of this project, we have gained more insight on how audios are being processed and extracted. In both theoretical and practical concepts, searching functions and instrument recognition have been approved in many sources. However, this is an application that lacks of resource supports. That is, the quantity of songs in our database is inadequate (around 40 songs) since we are collecting and labeling audio data by ourselves. As a consequence, matching results have a possibility that produces a large number of false positives in terms of both searching and classifying aspects.

For future implications, it is crucial to expand the size of existing audio resources for minimizing false positives produced by system. Besides, the next step of our application is to identify multiple instruments contained in mixed audios, since current version is detecting single instrument in audio clips. In addition, this application is only focusing on classical music since the workloads for covering multiple kinds of music are exceeding the effort we could make within a certain amount of time. Therefore, current system can be extended for exploring different types of music.

5

Conclusion

In this project, an integrated application for identifying and classifying classical music is delivered in following two aspects, *Shazam*-like audio searching and instrument extraction / classification. In terms of instrument extraction, the combination of MFCCs and Random Forest classification has shown a reasonable approach based on empirical experiments. On the other hand, rapid development on audio fingerprints demonstrated promising matching results. However, due to the limitation of resources, other classification technique such as convolution neural network (CNN) is given up in this version of system. To sum up, both existing empirical techniques, audio fingerprinting and MFCCs, have shown strong strength on audio matching and recognizing. in addition, improving current system with the concept of CNN can be carried out once expanding enough size of training data sets.

Bibliography

[1] DG Bhalke, CB Rama Rao, and DS Bormane. Musical instrument recognition using higher order moments. *Digital Signal Processing*, 5(4):133–138, 2013.

[2] Ian S Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen. *The MPEG-21 book*. Wiley Online Library, 2006.

[3] Mei Chen, Qingmei Xiao, Kazuyuki Matsumoto, and Xin Luo. Detecting repeating pattern in fingerprint feature sequence. In *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, pages 1864–1869. IEEE, 2015.

[4] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*, volume 1, pages 191–194, 2005.

[5] Jeongseok Jo, Sungjun Han, and Jongweon Kim. Improvement of spectral fingerprint for audio content recognition. 2016.

[6] Antonius Adrianus Cornelis Maria Kalker and Jaap Andre Haitsma. Fingerprint database updating method, client and server, April 21 2009. US Patent 7,523,312.

[7] Samuel Kim, Erdem Unal, and Shrikanth Narayanan. Music fingerprint extraction for classical music cover song identification. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 1261–1264. IEEE, 2008.

[8] Sunhyung Lee, Dongsuk Yook, and Sukmoon Chang. An efficient audio fingerprint search algorithm for music retrieval. *Consumer Electronics, IEEE Transactions on*, 59(3):652–656, 2013.

[9] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *arXiv preprint arXiv:1511.05520*, 2015.

[10] Ben Milner and Xu Shao. Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model. In *INTERSPEECH*. Citeseer, 2002.

[11] K Murty and Bayya Yegnanarayana. Combining evidence from residual phase and mfcc features for speaker recognition. *Signal Processing Letters, IEEE*, 13(1):52–55, 2006.

[12] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543–565, 2012.

[13] Matthew Sharifi, Ben Shahshahani, and Dominik Roblek. Unified recognition of speech and music, December 29 2015. US Patent 9,224,385.