Music Retrieval

Kerry Ngan, Yiqiu Huang, Thomas Hwang, Yangjiawen Xu

Section I

Introduction

Given the increasing repository of music tracks, it is important to personalize user discovery and exploration. Music recommendation systems play a vital role in user experience by allowing users to discover more music that they enjoy.

Besides the music recommendation system, we will also conduct a recognizer system including a music recognizer and a genre recognizer. In our daily life, we listen to a lot of music when we are in a restaurant, a supermarket, a coffee shop, etc. Sometimes, the music is so good and we want to download this music to enjoy in the future, but we do not know the name of this music. It is time to have a music recognizer at hand.

All code can be found in [kngan43/Music-Information-Retrieval (github.com)]

Tasks and Importance

We will have two main categories of tasks related to music retrieval - ranking and classification. For each category, we will have two tasks. The following tasks are listed below:

Ranking:

1. Songs embedding

Classification:

1. Music Recognizer
2. Genre classification

Both tasks in the ranking category will take a song as an input and output a list of similar songs with the most relevant at the top of the list and the least at the bottom. Retrieving a relevant list of similar songs serves to enhance user experience through a variety of similar songs they are likely to enjoy and introduce other songs that otherwise may not be found.

In the classification category, our task will be classifying the genre and identifying a song. Our input will take in a song and then output the corresponding label. The importance of these classification tasks is it allows users to identify a feature for a song and lead to better understanding of styles they enjoy.

Query, Search, Evaluation

Since we have different tasks, our queries will generally consist of inputting music tracks with their related features and outputting either similar music tracks or a label classification. The result from the query will also differ based on the individual task.

Music Recommendation using Song Embeddings

The query is a single song id and the output is top k similar songs to the input. The results will be a ranked list of songs in descending order of similarity. The relevant results are songs that are similar to the query. Since similarity in regard to music is subjective, the relevant songs will only make up a fraction of the results and will share some of the same features with the query such as genre and artist. For evaluation, the metric mean average precision will be used to determine how well the music retrieval model is performing since the user will be more likely to look at the top results.

Music Recognizer

For the music recognizer, the query will be a 10 seconds audio clip. After the system receives a sound clip as a query, it will be fingerprinted as hashes, and then the system will match the hashes and return the song which has the most matching hashes.

In terms of evaluation, the query song might exist or not exist in the database. For the query that already existed in the database, there are 3 cases for the result:

- The result matches the query, which is the ideal case
- The result matches an wrong song than query
- Nothing find

Also there are 2 cases of result for the query that not exist in database:

- Nothing find, which is the ideal case
- Return a song in the database

In this project, I will evaluate the recognizer with multiple queries and find the percentage for each case.

Genre Recognizer using K-means and Support Vector Machine

The input is the 30-second sound clip, which is of the genres: blues, classical, country, disco, hip hop, jazz, reggae, rock, metal, or pop. Each sound clip has the features, such as chroma_stft, roll-off, tempo, spectral bandwidth, spectral centroid, etc. After passing into the search engine, we'll get the labeled result, and can be transferred to the exact genre. The relevant results would be whether the classification is correct or not, we need to check whether the genre returned by the model is matched with the genre in the dataset. The evaluation metric is to calculate the accuracy.

Model Overview

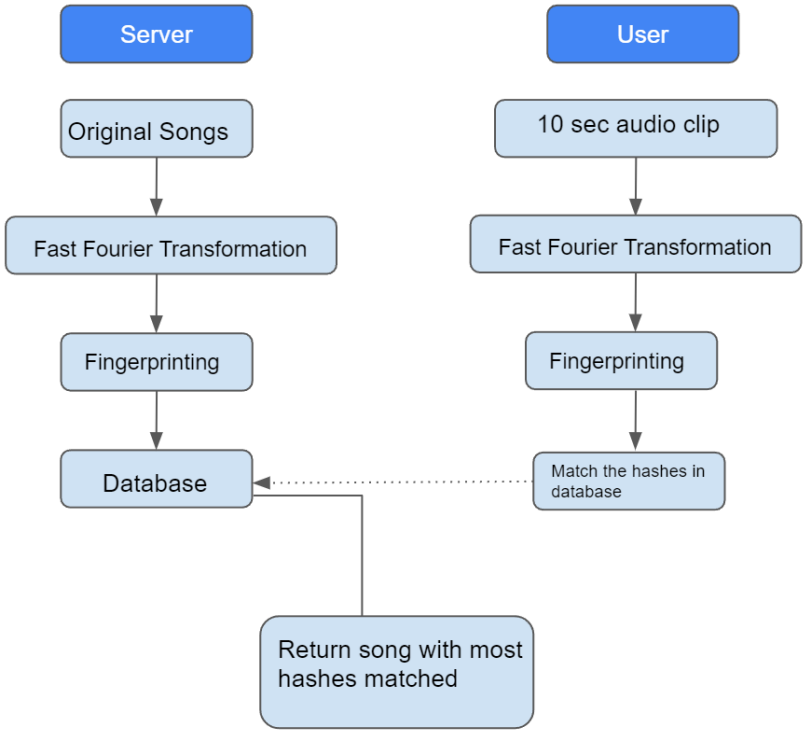
There are 3 tasks for our query - music recommendation with user embedding, music recognition, and genre labeling.

Music Recommendation using Song Embeddings

In the first model, playlists of songs are used to train a skip-gram retrieval model to rank similar songs. We are stipulating that radio stations are grouping similar songs to target a specific audience. We acquired a dataset from Cornell University which contains playlists from hundreds of radio stations around the US. Each song in a playlist is represented by an index id and serves as an input to a word2vec skip-gram model. The word2vec model trains an embedding vector for each song where the dimension in the vector represents the closest distance from a given song. Using the song embeddings in the skip-gram model, we can make a query by inputting a song id and then retrieve a list of songs that are ranked by similarity in descending order.

Music Recognizer

The first step with the Shazam algorithm is to record and sample the Initial Songs and then convert it to frequency domain by Fourier Transformation. The next step is fingerprinting and storing the song in SQLite. Then, the music recognizer is ready to retrieve the song from the database. The workflow is illustrated below:



Fast Fourier Transform(FFT)

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. Its now commonly used in audio processing.

Fingerprinting

An audio fingerprint is generated from an audio signal that can be used to identify an audio sample.

With the spectrogram generated by FFT(1A), I then use a max filter to iterate over the spectrogram image to make sure that only the peaks are kept (1B). In a specified number of "neighborhood" peaks, a hash will be generated between the peak and each of its neighborhood(1C). The hash is generated by both the peak's amplitude and the time delta between the peaks(1D).

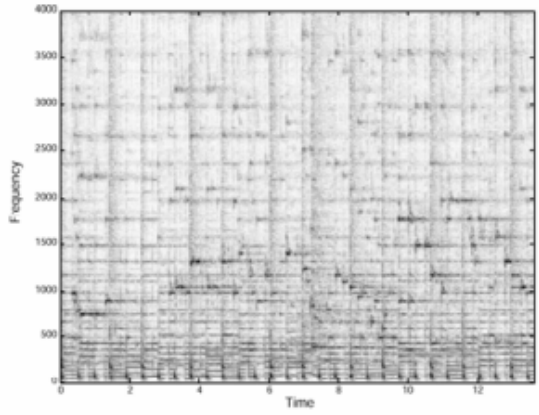


Fig. 1A - Spectrogram

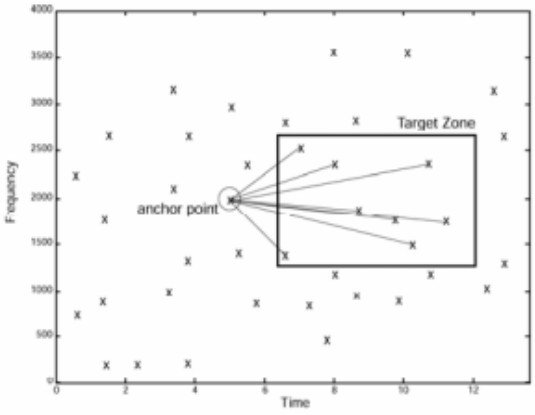


Fig. 1C - Combinatorial Hash Generation

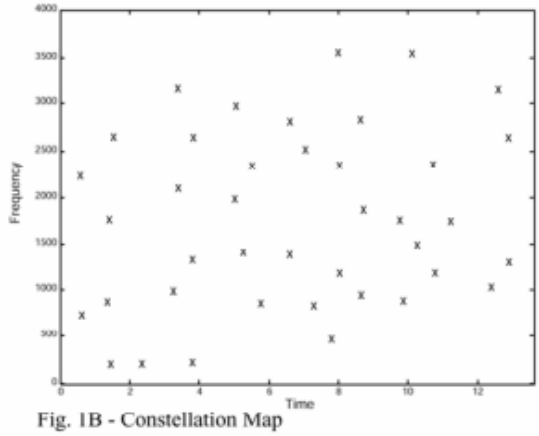


Fig. 1B - Constellation Map

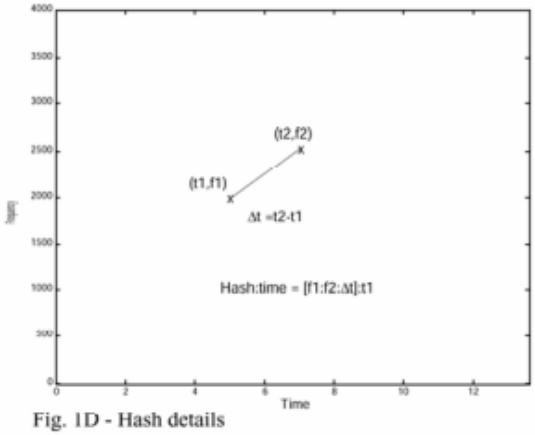


Fig. 1D - Hash details

[2] Fingerprint Algorithm

Genre Recognizer using K-means and Support Vector Machine

We will preprocess the genre collection dataset from GITZAN genre collection dataset. First of all, we get the features of each music by extracting the features from its spectrogram and writing those features into a CSV file for the following analysis. Afterwards, each music will be labeled from 0-9 instead of its genre, meanwhile we will check whether there is null value in our dataset. Then, we'll split the dataset into training dataset and test dataset. Given so many features of each song, we will apply a decision tree classifier and random forest classifier to rank the importance of features across all music.

Afterwards, we will train the dataset using KNN and Support Vector Machine separately. After training the dataset using these two models, we will make a prediction on our test dataset and derive the accuracy of these two models.

Section II

Implementation and Extra Credit

Music Recommendation using Song Embeddings

Sample Queries:
Single song tracks represented by an index id. For example, if we have “Can’t Help Falling in Love with You” by Elvis Presley mapped to index 1, our input query will simply be 1.

Narrative:
Identify songs that a user will enjoy.

Relevant Judgment:
Relevant songs will be songs that share similar features with the query. These indicate a high probability that the user will also enjoy them because they can capture similarity in instruments, rhyme, or beat which correlates to the user preference. Non relevant songs are songs that do not have any similarity features with the query. In our case, there are two additional features, tags and artist. If any songs from the hits share an artist or have the same tag, the song will be scored as relevant.

Results:
On the first run, we queried on one song id and received 10 hits. The title was “You’re Always On my Mind” by SWV with tags (1,10,12,13,15,18,23). One result returned was the song “Time” by Anthrax with tags (0, 2, 8, 19, 23, 26, 28, 51, 66, 81, 102). We can see that both songs have tag 23 so we mark the document as relevant. Another result was the song “Hard Road” by The Shore with tags(0,7,30,101,162) which has no matching tags or artist so this song is marked as not relevant

To evaluate the search, we used mean average precision (MAP). We ran 100 queries, each returning 10 hits. For the result we achieved a MAP score of 0.6075040091962712.

Music Recognizer

Sample query: 10 seconds song clip
Narratives: The user want to find the song name that match the query
Relevance Judgements: The result judgment will be binary. The system will return 1 result and if the result matches the query, it will be a 1 for the judgment. Otherwise, it would be zero**.**

Audio retrieval tasks have a different evaluation metric compared to the document retrieval system since users are trying to find the clip that exactly matches their query. In practice, a track can have multiple versions such as an electronic dance version or a cover version. However, in this project, each song in the 500 will be unique with no other version. Thus, only the top 1 song in the retrieved songs are taken in evaluation.

In this project, I fingerprint around 500 songs and save into the SQLite database.

The system is first tested with original songs as queries. 281 songs are included and each song will generate 3 ten second clips as a query and go through the system. In all 843 queries, 78.4% of the queries are successfully recognized with the exact matching song as the top 1 returned song. 51.9% of the songs are being fully recognized with all 3 queries, and only 1.4% of songs fail to be recognized in all 3 queries.

In practice, users might use the function when they hear a familiar song in the club or the restaurant, so the following metrics are acquired by testing on recorded video clips by myself. Those queries are played by my phone or my sound, and recorded by my PC with all kinds of normal noise.

30 recorded songs are included and each of the recorded songs will generate 5 queries. Among the 30 recorded songs, there are 15 songs in the database and 15 songs not in the database. In the total 150 queries, the result are illustrated below:

	Query Song in Database (75)	**Query Song not in Database(75)**
Retrieved Song Match	**25**	**-**
Retrieved Wrong Song	40	60
Find Nothing	10	**15**

In general, among half of the songs that are in the database, the recognizer achieved an accuracy of 33.3%. 53.3% of the query matched the wrong song in the database. On the other hand, for the other half songs that are not presented in the database, those 75 queries have an 80% chance of returning a wrong song.

Genre Recognizer

Sample queries: 10 30-second sound clips (those are saved in our GitHub repo)
Narratives: Given the query, the user needs to determine its genre.

Relevance Judgements: The relevance will be evaluated by the models – KNN and Support Vector Machine. It is a multi-valued relevance. The result is its numeric label and transfer the numeric label to genre.

In the GTZAN dataset, there are 1600 sound clips. After the dataset is splitted into training data and testing data, we will have 800 sound clips to test the models.
After testing on testing data using KNN and Support Vector Machine, the recognizer gets the accuracy of 63.63% and 72.12%.

Reference

[1] Fast Fourier transform
https://en.wikipedia.org/wiki/Fast_Fourier_transform
[2] An Industrial Strength Audio Search Algorithm.
https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf