Assignment 2:

Audio Identification

Dekun Xie | xiedekun@outlook.com

1 Introduction

Audio Identification is a high specificity retrieval task which consists in identifying the particular audio recording that is the source of the query when given a small audio fragment as query[1]. In this assignment, an industrial-strength audio search algorithm which was developed by Wang[2] used in the commercial Shazam will be implemented.

2 Datasets

For audio identification, a subset of 200 classical and pop audio clips from the GTZAN[3] datasets will be used as the target song database and a set of 213 audio tracks that correspond to noisy versions of the same recordings will be input into the algorithm to query.

3 Implementation

The paper in 2003 introduced a method called audio fingerprint which is used for comparison between queries and target song databases. In addition, an indexing technique based on hash is also present to speed up this process. Firstly, a basic concept of fingerprints will be introduced.

3.1 Concept of Audio Fingerprints

Fingerprint is a series of reproducible hash tokens extracted from both database audio and sample audio. The fingerprints from the unknown sample are compared to a huge database of fingerprints produced from music before the candidate matches are evaluated for correctness of match. Similar to the property of uniqueness of human fingerprints, according to some guiding principles, the fingerprint of audio should be temporally localized, translation-invariant, robust, and sufficiently entropic[2].

3.2 Building Audio Fingerprints Based on Spectral Peaks

To implement the attributes of temporal locality, translation invariance and robustness for fingerprints, a spectral-peaks-based method will be applied.

The main idea is to calculate the coordinate by picking the max magnitude in several areas with the same size from the spectrogram (using Short Time Fourier Transform[4] (STFT) in this experiment) which will be converted into a constellation map (Figure 1) remaining the main feature of the audio. As mentioned, the spectral peaks are local in nature with regard to time and frequency, therefore fulfilling the locality requirement. Furthermore, when choosing a small hop size in the STFT computation, the peak features become invariant to signal translations. Finally, the position of most spectral peaks is stable in spite of noise or signal degradations, which shows the robustness[1].

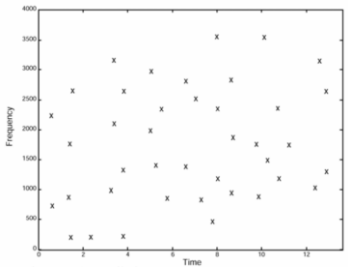


Figure 1. Constellation map[2]

As for the experiment, in the first step, a STFT spectrogram is produced with 1024 FFT size, Hann windows, 1024 window length and 512 hop size, which is shown in Figure 2. Based on it, the constellation map (Figure 3) is able to be calculated with the help of the peak_local_max function (min_distance=5; threshold_rel=0.05)

in python package called skimage[5]. Both methods will be used in database songs and query songs.

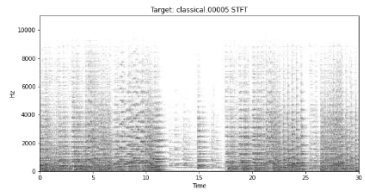


Figure 2. STFT spectrogram of database song, e.g., classical.00005.wav

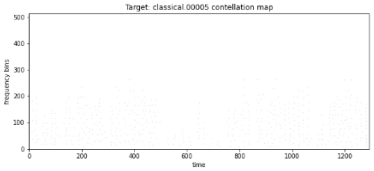
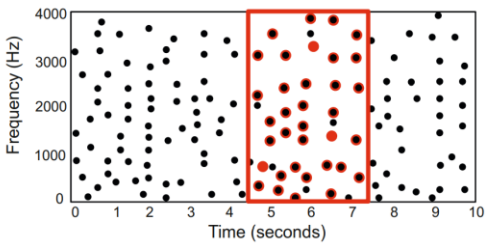


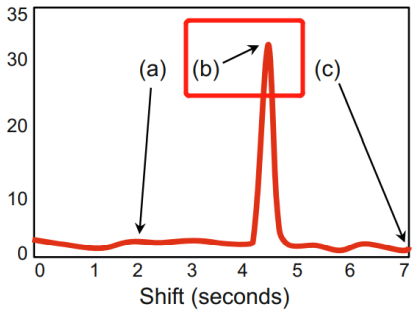
Figure 3. Constellation map of database song, e.g., classical.00005.wav

3.3 Fingerprints Match

After constellation maps of database songs and query songs are acquired, the query map is moved gradually and compared with the database map (Figure 4.a) before only the offset time with the highest matched number is remained (Figure 4.b). This process will be done for every database song and the one with highest matched number (similarity) will be chosen as the best matched result for the query song.



(a)



(b)

Figure 4. Match the fingerprints[1]

3.4 Indexing Technique with Hash

3.4.1 Indexing Technique Concept

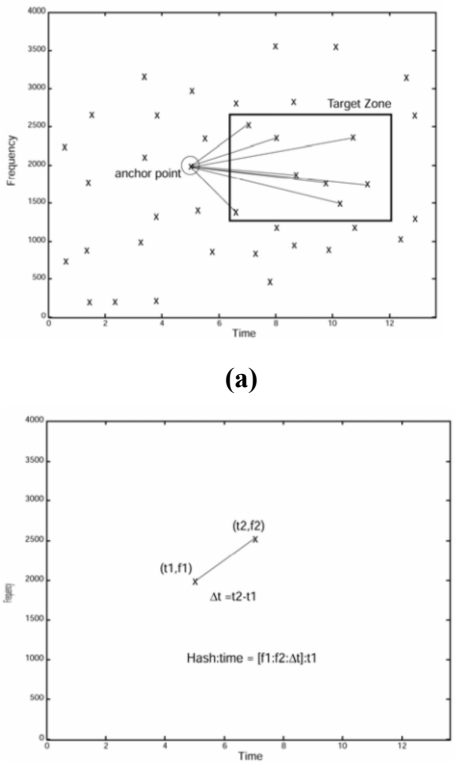
However, finding the correct offset directly from constellation maps can be rather slow, due to raw constellation points having low entropy[2]. Therefore, an indexing technique using hash, which optimizes speed and performance by cutting down the search space through suitable look-up operations is introduced.

In information retrieval, the notion of a hash is used to refer to a fixed-length identifier that acts as a shortened and compact reference to the original, more complex data entity.[1] When compared, the query song only needs to search the database songs with the same hash instead of search all the songs which increase the searching efficiency and the entropy of every search, meeting the requirement of sufficient entropy of audio fingerprints.

3.4.2 Create Hash Fingerprints

The hash is calculated based on a combination of an anchor point and a target zone (Figure 5.a). The anchor point will be combined with each point in the target zone as pairs. If k0 and k1 means the frequency stamp when n0 and n1 means the time stamp for the anchor point and the target zone point respectively, (k0, k1, n1-n0) will be regarded as hash (it can be input into the hash function to calculate) (Figure 5.b). And then, each point will be chosen as the anchor

point. Therefore, one song will contain many hashes which remain the relative features of the points.



(a)

(b)

Figure 5. Anchor point; Target Zone and hash calculation[2]

The paper from Shazam doesn't explain how to generate these target zones. The method of mine is choosing the anchor point from start and after a certain anchor distance (8 time-stamp samples in my experiment), points of target size (also can be called fan-out; it is 28 in my experiment) will be chosen in order. The rule of order relation between the time-frequency points in a spectrogram is that: If two time-frequency points have the same time, the time-frequency point with the lowest frequency is before the other one; If a time time-frequency point has a lower time than another point one then it is before[6]. In the end, every pair will be input into Sha256 hash function to generate hash and the hash of the target database songs will be stored in the local machine with value of ["anchor absolute time", "database song id"]. As this hash is reproducible, one hash may contain different value of anchor absolute time and database song index.

This is the way for the final audio fingerprint to generate and the example fingerprints of database songs are present in Table 1 in Appendix A.

3.4.3 Match Fingerprints

Once there is a song to query, it will be also converted into hash-value fingerprint in the same way mentioned above but the value only records ["anchor absolute time"].

Before finding out the best related result, these two fingerprints should be compared with indicator functions shown in Figure 6 where n means absolute anchor time stamp of the query song when h means the hash of the query song. L(h) means the absolute anchor time stamp of the database finger prints for the same hash of the query one.

Query (*n*,*h*)	*L*(*h*) - *n*	Indicator functions									
		...	-1	0	1	2	3	4	5	6	...
(0,2)	(2,4)	0	0	0	0	1	0	1	0	0	0
(1,3)	(-1,2,4)	0	1	0	0	1	0	1	0	0	0
(1,4)	(2,3)	0	0	0	0	1	1	0	0	0	0
(2,2)	(0,2)	0	0	1	0	1	0	0	0	0	0
(2,4)	(1,2)	0	0	0	1	1	0	0	0	0	0
Matching function		**0**	**1**	**1**	**1**	**5**	**1**	**2**	**0**	**0**	**0**

Figure 6. Indicator functions[1]

For instance, h = 'ebe7b86d83ac729fe452781ba4bd8629d667772bcc5e8e0c39159641b185c11f' in Table 1 and L(h) = [13, 1217, 821, 220]. The offset time between the query song and the database song can be calculated by L(h) – n. The indicator functions will count the show number of each offset and sum all with matching function where the largest counting number of offsets will represent the similarity (score) between the

fingerprints of database songs and query songs. In the end, the scores for the same song will be combined and the song with highest score will be regarded as the searching result.

The example result of the experiment is shown in Table 2 in Appendix A.

4. Result and Evaluation

In this experiment, I use mean average precision (MAP) and Maximal F-Measure to evaluate the algorithm. As shown in Figure 7, the MAP is 47% and 68% for classical and pop music respectively when the Maximal F-Measure is 51% and 70%. It means the pop music is easier to match than the classical music. As the pop music is usually with the stronger drum and higher volume than the classical music, the influence of environment noise will be less, which leads to the higher recognition rate.

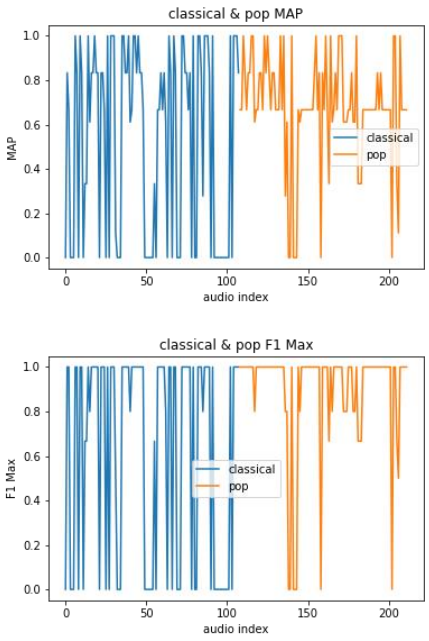


Figure 7. MAP and Maximal F-Measure for Hash Method

The overall result is not excellent but above the chance. To improve the result, I also try comparing all the points from the constellation to calculate the score of each song. The MAPs of the pop music and the classical music increase to 76% and 81% and the Maximal F-measures rise to 67% and 96% respectively (Figure 8).

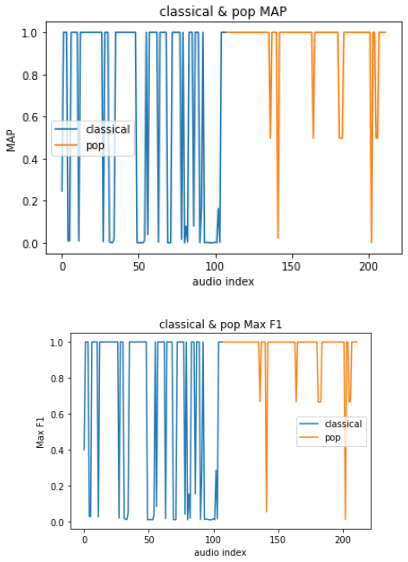


Figure 8. MAP and Maximal F-Measure for Non-hash Method

In spite of the huge improvement of recognition, without hash, it runs really slowly and spends 10min 30s (1min 44s for generating database fingerprints and 8min 46s for query song fingerprints generation and comparison for 200 database songs and 213 query songs; the computer is Surface Pro 7 8GB) to finish the code. Compared with the method with hash, it spends around 3min 13s (1min 16s for generating database fingerprints and 1min 57s for query song fingerprints generation and comparison) which raises 69% speed of the algorithm while reduces around 27% mean average precision. Therefore, it is important to balance the efficiency and correct recognition of the algorithm.

5.Conclusion

The method of Shazam with my parameters works well while the balance of efficiency and correctness of the result should be considered.

Reference

1 Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications* (Vol. 5). Cham: Springer.

2 Wang, A. (2003, October). An industrial strength audio search algorithm. In *Ismir* (Vol. 2003, pp. 7-13).

3 A. Olteanu. Gtzan dataset - music genre classification. [Online]. Available: https://www.kaggle.com/andradaolteanu/gtzan-dataset-musicgenre-classification

4 Cohen, L., & Lee, C. (1989, November). Local bandwidth and optimal windows for the short time Fourier transform. In *Advanced algorithms and architectures for signal processing IV* (Vol. 1152, pp. 401-425). International Society for Optics and Photonics.

5 Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., ... & Yu, T. (2014). scikit-image: image processing in Python. *PeerJ*, *2*, e453.

6 http://coding-geek.com/how-shazam-works/

Appendix A

Table 1. Example hash audio fingerprints of database songs.

Hash	Value (anchor absolute time, song id)
ebe7b86d83ac729fe452781ba4bd8629d667772bcc5e8e0c39159641b185c11f	[(13, 0), (1217, 81), (821, 96), (220, 164)]
fe5ce99b5e8c7c62d62025b18ea380c0da648ef3e258dc74ec6dafc63b2ae685	[(13, 0), (22, 69), (716, 96)]
520e9872b6af29a992a41c6274f053c0ab13219c44d79711e0546057fc3fbdd3	[(13, 0), (633, 37), (821, 96)]
243ca4be926ec3c1bdb0c14259372e5b0ceaacc5358c6736869f2d59a2c39a2e	[(13, 0)]
9034bbfc552c02e4a6dcb925f19125af66402e59d3eb3224f9cde37c28a89e24	[(13, 0), (88, 2), (529, 60), (229, 69)]

Table 2. Example query results

Query song	Result: (song id, score)	Result: song name
classical.00000-snippet-10-10	**(0, 206)** (89, 178) (2, 149) …	**'classical.00000'** 'classical.00089' 'classical.00002'…
classical.00004-snippet-10-10	**(4, 93)** (45, 68) (42, 59) …	**'classical.00004'** 'classical.00045' 'classical.00042' …
classical.00005-snippet-10-10	**(5, 32)** (97, 31) (42, 28) …	**'classical.00005'** 'classical.00097' 'classical.00042' …