

To explore the mechanism that allows music recognition apps
to work.

Cephas Yeung

28/4/2025

Contents

1	Introduction	4
2	Methodology / Design Choice	5
3	Testing method	7
4	Discussion	8

A	Bibliography	9
B	Figures	10
C	Testing results	11
D	Meeting logs	12

Abstract

We recreated an algorithm used to recognise music.

The algorithm first fingerprints all the songs by create a spectrogram, generate peaks point inside it. To create an almost unique hash point where it's inserted inside a database.

When record songs through the interface, the algorithm repeats the process of fingerprinting, but instead compare hash value with matched songs

CHAPTER 1

Introduction

In modern society there are billions of songs created by different artists and different styles of music. Most of us might have already heard or have used the music recognition app Shazam, or something similar on your mobile phone when you are at a restaurant or Café because you want to search for what song it was playing.

From using a very concrete source of a pattern created by the original founder and Chief Scientist of Shazam ¹[1].

This source allows a great understanding of the background information of the program.

My aim for this research is to understand and replicate how modern day music recognition algorithms (explain in footnote) work.

During the creation process, it uses different resources to make the resource and also methods to test the success criteria set out by the algorithm.

In conclusion, the program are successful.

¹Now part of Apple Inc. As they are acquired by Apple.

CHAPTER 2

Methodology / Design Choice


The artefact is separated in different pieces. As illustrated in the figures inside [Appendix A:Figure 1](#).


The software is run inside a container as the pattern suggested that although not limited to any particular system, it is preferred for it to be run in a “Distributed system”¹. But by containerize the program, it allows it to happen.

Next, the program is separated into two parts. The Web app part is for the user to interact with the algorithm graphically.

And the main algorithm are produced using python. Inside the program, the library of NumPy[4] to analysis the data.

The program is replicated as described in the pattern by first recording the audio, this is done through the web app.

All the original audio is first sent to an algorithm to “fingerprint” the piece of music. The algorithm are consist of spectrogram produced from the SciPy signal library where it uses STFT². [6] An array of data illustrated in the figure.  This arrays of data are being filtered through maximum filter in the SciPy library [6]. Which is illustrated in the figures.

The two arrays are compared to use to pick peak points. The peaks are used to construct the pairs shown here. 

The pairs of points are then hashed using the default hashing function³ from python which generate an almost unique hash values in the program.

The hash point is than placed into a database alongside with its song name and where the point A’s real time. The song data (metadata) such as author and copyright detail are placed in a different database.

¹Computer systems whose intercommunicating components are located on different networked computers[2]

²Short-time Fourier Transform

³Hashing function is a set of things to do to make an output that is always the same length from some other input data.[3]

When the user record the sample through the user interface created, the recording sample are send to the server. It is processed where it applies the same algorithm from above, it first construct a spectrogram, create and compare it with a filtered spectrogram, peak point are picked and hash values are created, the hash value are then searched inside the ‘points‘ table.

The list of successfully matched song are than going through the process of ”scoring”. The score are calculated by constructing a histogram with the point A’s real time value. The score are assigned to each songs with the highest value in each histogram. This is because the peak point of some songs are the same for a completely different song. And this includes background noise where the algorithm could mistake it to be a different song. Illustrated in the figure.

[7][5][8]

CHAPTER 3

Testing method

The program is tested using the method described in this paper.[\[8\]](#) We used most of the songs in fingerprinted collection because that reduce the likelihood of a flawed test. We mainly used classical music because there is lower chance for creating copyright issues.

The test have increasing level of difficulty:

- Test 1: original song file (unmodified digital file)
- Test 2: original song file but shifted
- Test 3: original song file but with controlled level of noise. (E.g.: a sine wave)
- Test 4: original song file but played over the air and recorded.
- Test 5: original song but perform by different people.
- Test 6: transposed song

The test results is inside [C.1](#). The program overall worked as expected but test 6 doesn't work because this algorithm is not designed for this.

CHAPTER 4

Discussion

As discussed in [the test conducted](#), this artefact is not capable to detect a transposed song. Because this song only matches the exact frequency that the song produces.

APPENDIX A

Bibliography

- [1] Avery wang - member board of trustees @ SETI institute - crunchbase person profile.
- [2] Distributed computing. Page Version ID: 1266361966.
- [3] Hash function. Page Version ID: 9718140.
- [4] Charles R. Harris, K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. Van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández Del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. 585(7825):357–362.
- [5] Cameron MacLeod. abracadabra: How does shazam work? - cameron MacLeod.
- [6] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: fundamental algorithms for scientific computing in python. 17(3):261–272. Publisher: Nature Publishing Group.
- [7] Avery Li-Chun Wang and Julius O. Smith III. Systems and methods for recognizing sound and music signals in high noise and distortion.
- [8] Cheng Yang. Music database retrieval based on spectral similarity.

APPENDIX B

Figures

Contained Program (Using Docker)

Web app (Build using Svelte)

Back bone of the program (Build using python)

APPENDIX C

Testing results

filepath	passed	SUBTOTAL	filepath	passed	SUBTOTAL
test/test_1.py	46	46	test/test_2.py	46	46
TOTAL	46	46	TOTAL	46	46

Figure C.1: test result table

APPENDIX D

Meeting logs

Meeting with the supervisor at 25 November 2024

Summary:

Improve on the part A of the production logs, with more details on the titles described and what resources I will be using (such as the URLs should be provided), the timescale and what is involve.

As things I should be including to create a great project is the benefit that this might bring to the society and the impact.

And a plan to see what I should be sending or showing to the examiner in the future such as:

- Send product log
- Presentation of the artefact
- Evidence of the artefact
- Critical bibliography
- Place that in the Appendix and write a reference back to the appendix through the production log.

Meeting with a professional at 13 January 2025

Summary:

I explained the background details during the meeting.

I have explained that the program can only detect sounds from following test that I conducted such as:

- A test where there are no modifications to the original sound,
- A test where the song has cropped and reduced to a smaller size.
- A test where there is a constant noise generated by a program with a constant sine wave in the background.
- A test where the songs are mixed with a normal acceptable background noise such as coughing.

All of those above works as expected.

However the test that I did by recording when the sound is played on a music player on a different device doesn't work.

As an additional note, he has mentioned that the program is slow because of a reason which might be caused by the hash point not being an integer, as the hash value using integer is faster than using string as hash. (It was diagnosed with a slow database reading speed prior to the meeting).

The project has concluded as partially successful from the results produced from the searching algorithm.

Summary of Improvement I need to make from the project.

I should use a different hashing algorithm for the fingerprinting part of the program.

Round up the frequency, and the changes in time that are fed into the hashing algorithm.

Additional note:

The meetings are conducted with the consent of the professional. (With regards to the ethical issues)

The professional is my computer science teacher, as he has a decent qualification of a university degree.