Second interim report

Topic: Shazam implementation

Team: Dmytro Batko, Anastasiia Petrovych, Yurii Zinchuk

• Explanation of general topic and detailed description of the problem you want to solve

Doing research on the previously chosen topic of the project about Music Generation with Markov Chains, we thought that it would not require a significant amount of LA knowledge and its application; thus, we decided to change the main topic to make our task more interesting for us and with more use of Linear Algebra methods. Currently, our main theme will be discovering the Shazam algorithm and its development.
You hear a familiar song in the club or the restaurant. You listened to this song a thousand times long ago, and the sentimentality of the song really touches your heart. You desperately want to hear it tomorrow, but you can't remember its name. Fortunately, in our amazing futuristic world, you have a phone with music recognition software installed, and you are saved. You can relax, because software tells you the name of the song, and you know that you can hear it again. It is the short description of the Shazam algorithm, which allows you to record your song and get the information about it. Mobile technologies, along with the huge progress in audio signal processing, have given us algorithm developers the ability to create music recognisers. One of the most popular music recognition apps is Shazam. Suppose you capture 20 seconds of a song. In that case, no matter if it's intro, verse, or chorus, it will create a fingerprint for the recorded sample, consult the database, and use its music recognition algorithm to tell you exactly which song you are listening to. Our aim is to try developing our own user-friendly application similar to Shazam, which allows us to find the song which you have heard before. Up to me, Shazam still has some issues with music

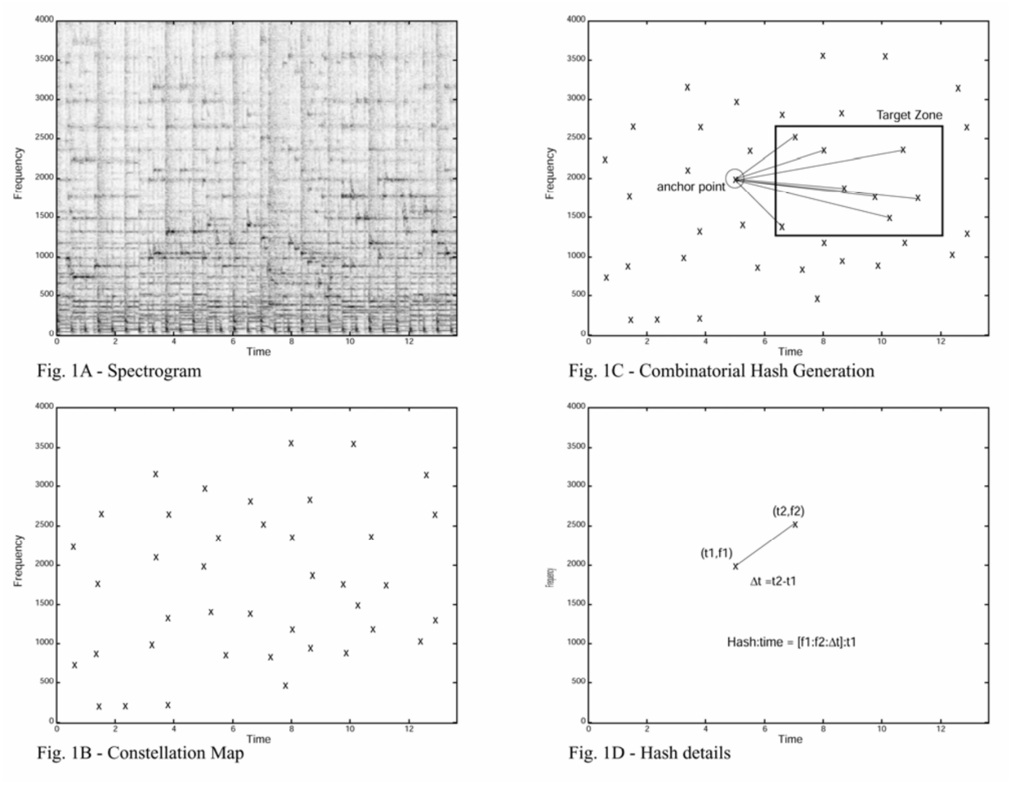
denoising. For instance, if one records a song in a noisy club, this software won't recognise it because the noise doesn't allow one to do it correctly.
We would like to try to develop our software and try different denoising methods to find out how this application works and can be improved.

• A short review of related work and possible approaches to solutions

Indeed, there are many solutions that propose Music Recognition, as Shazam and other apps can do (e.g. SoundHound, Genius, Musicxmatch, Google Assistant & Siri). But the basis of it is transforming audio fragments into the frequency domain by using FFT. It will help to extract further the fingerprint or signature for the audio fragment providing a static representation of a dynamic signal. From that point, there is a difference between the algorithm of Shazam and some attempts to imitate it. There are many variants of it, but our aim is to highlight two of them. The one that will be implemented is Shazam's algorithm.

Hashing with Anchors and target zone (Shazam)

Another way to fingerprint the audio fragment is using Shazam's algorithm. As FFT has been performed on the song, one needs to find the strongest frequencies in that signal. It is important that they have to be evenly spaced through the spectrogram of the signal to allow fingerprinting of all fragments. Also, it is common knowledge that the recorded song fragment may contain noise; thus, one has to be confident that those peaks are evenly spaced in frequency so that the algorithm can deal with noise and frequency distortion.[3]



Picture [1]

In the picture above, one can see that having a Constellation map (shows the strongest frequencies), one can now perform our fingerprinting. Fingerprint hashes are created using a constellation map where pairs of time-frequency points are combined in various ways. Anchor points are selected, and each has a target zone. These anchor points are paired with other points in their target zone, resulting in two frequency components and the time difference between the points. This process generates reproducible hashes even when the audio is affected by noise or voice codec compression. The further actions are matching those fingerprints of recorded fragments and the ones in the database of songs' fingerprints.

The other way of doing that is:

1. **Hashing Intervals**

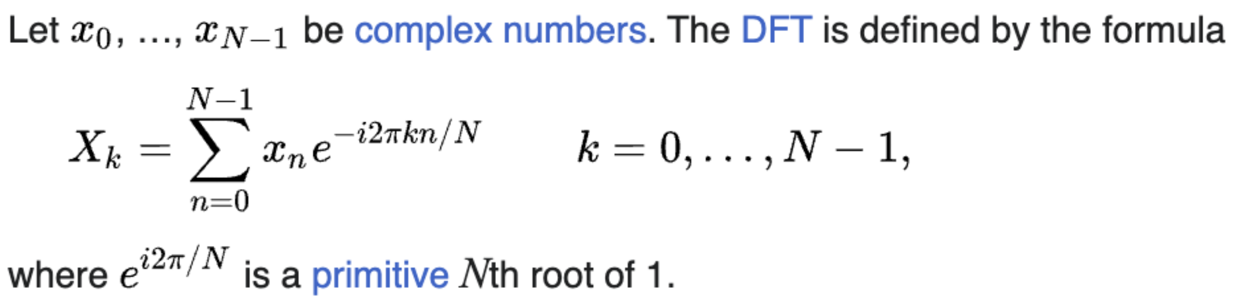
To make Music Recognition more efficient, those algorithms use some kind of sliding window, as frequency domain representation does not provide the most important information about the timing of, for example, some peak frequency [4]. Such sources as [4; 6] make FFT on every chunk of the music (i.e. 44 chunks in one second, given 44100 sample rate) and then, having intervals of frequencies, detect the peaks with the biggest magnitude. *This information forms a* *signature for this chunk of the song, and this signature becomes part of the fingerprint of the song as a whole* [4]. Hashing the data is then stored in the database for further matching.

• A brief explanation of the algorithm chosen and its pros and cons

We start developing our app with the Shazam algorithm, which consists of many parts:
First of all, the given recorded song should be converted to the sound wave with the help of FFT. Then we need to find out the target zone for the song(the interval on which we detected the peak). With the help of peaks, we need to hash the song, which helps to identify the song in a brief period of time. For hashing, we can choose different methods which will impact the speed of the search, reduce the possibility of mistakes and increase the correctness of the software work.
The pros of this algorithm are correctness, easy application, and low time complexity of searching for the match (with appropriate hashing). The cons of this algorithm are missing denoising, which reduces the chances to find out the song in public places and the time complexity of some FFT processes.

• The theoretical part behind the algorithm with stress on the LA methods used

The main part of the Music Recognition Algorithm is Fast Fourier Transform (further FFT), an algorithm used to transform a time-domain signal into its corresponding frequency-domain representation. In the case of audio signals, the FFT can be used to analyse a sound's spectral content, allowing us to visualise and manipulate individual frequencies that make up the sound.

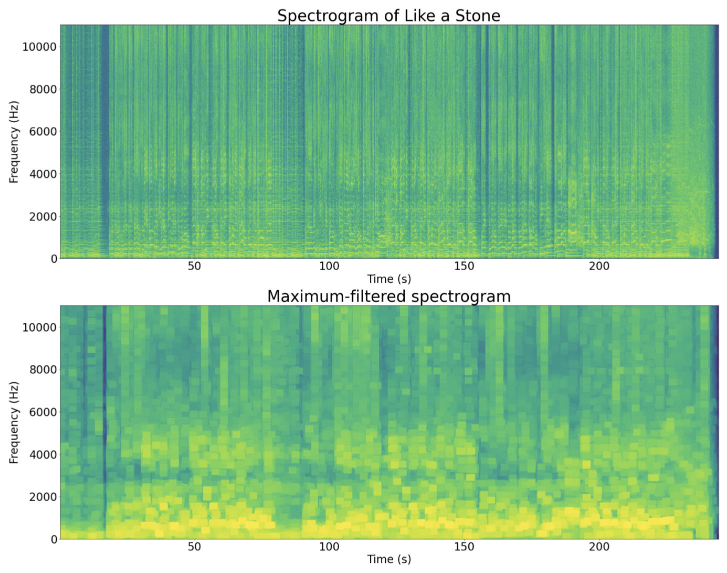


Let $x_0, \ldots, x_{N-1}$ be complex numbers. The DFT is defined by the formula

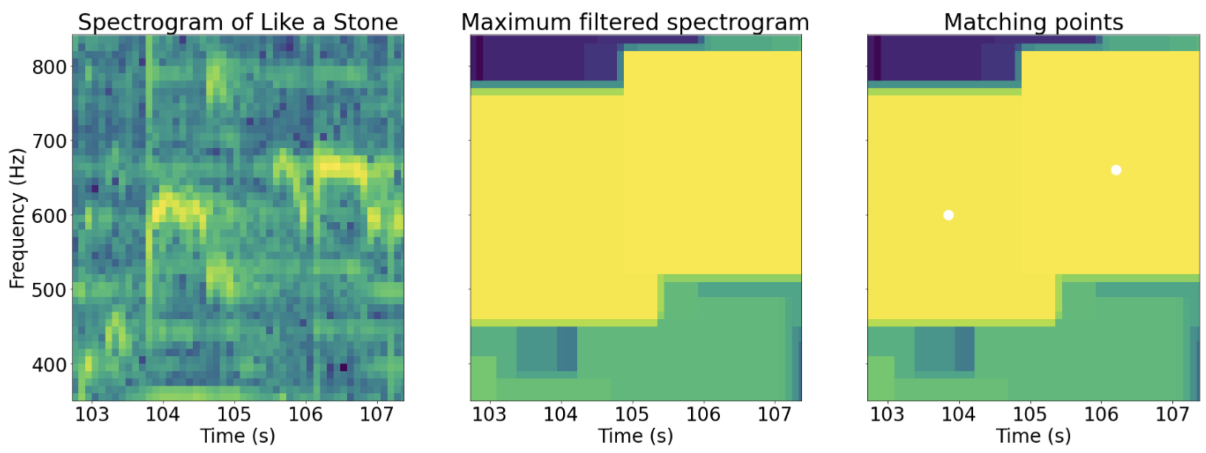
$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1,$$

where $e^{i2\pi/N}$ is a primitive Nth root of 1.

The basic idea behind the FFT is that any signal can be represented as a sum of sinusoids with different frequencies, amplitudes, and phases. The FFT algorithm takes a time-domain signal as input and produces a frequency-domain representation of that signal as output. The output of the FFT is a set of complex numbers which represent the amplitude and phase of each frequency component of the signal.
Also, going further into the algorithm, a Maximum filter algorithm is performed on our obtained frequencies. In a nutshell, every pixel of the spectrogram is being set to the local maximum by looking at its neighbourhood pixels. The example is given in the picture:

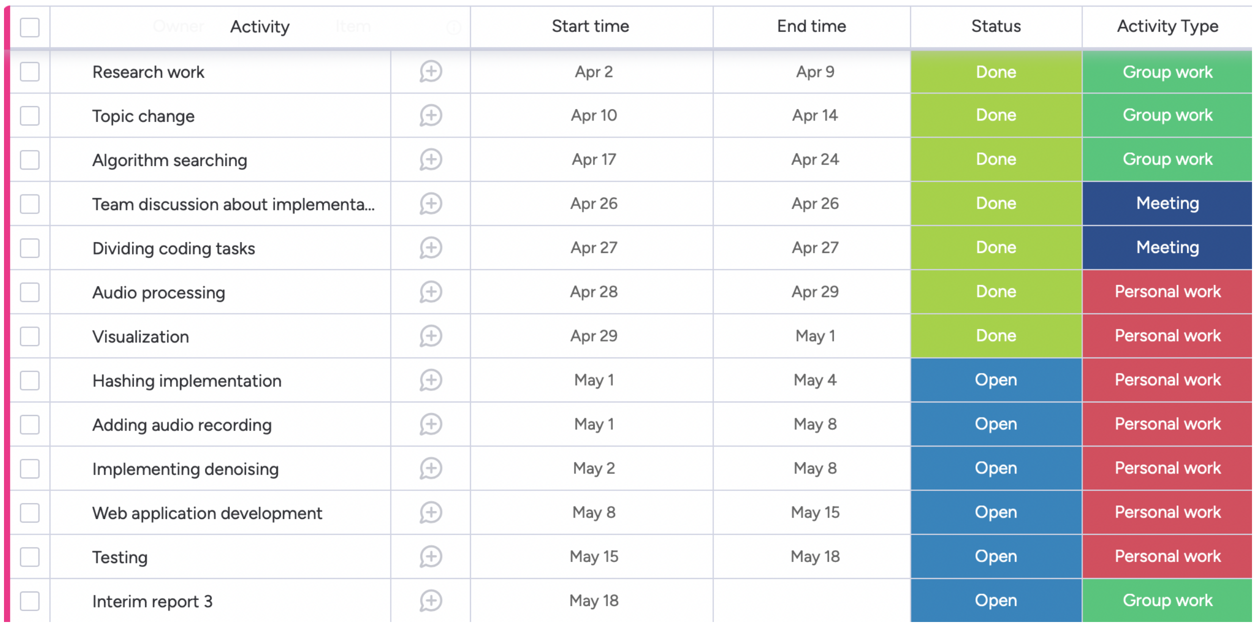


Then one has to find the location of the local peaks, as the Maximum filter algorithm only emphasises them. The concept behind this technique is that the local peaks have been used to replace all the non-peak points in the spectrogram, resulting in a modification of their values. The peaks are the only points in the spectrogram that remain unchanged; therefore, they are candidate peaks for the Constellation map. Below one can see how those peaks (white ones are chosen from a zoomed section of the above spectrogram).



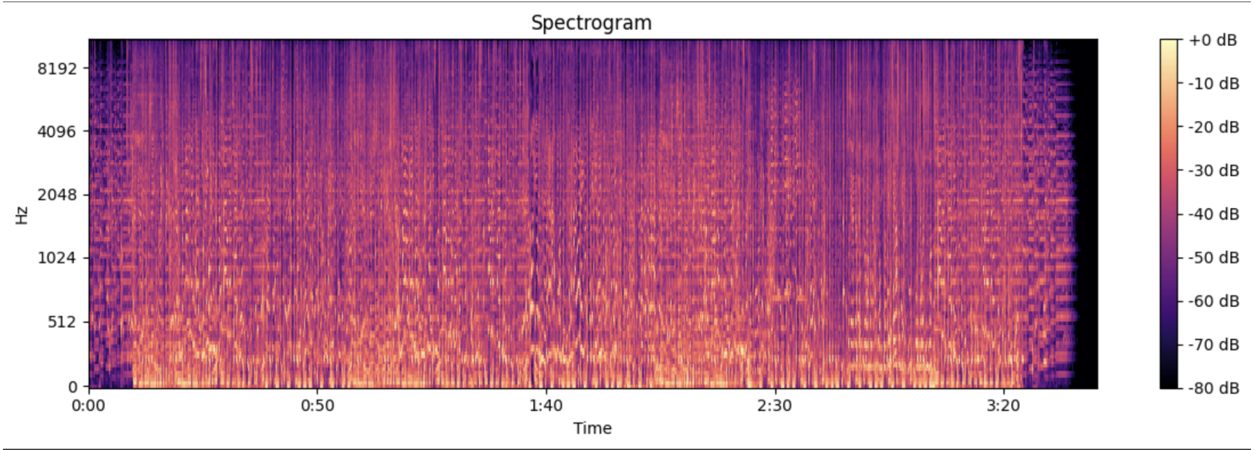
Then there is a need to hash that data and perform its matching with other song's fragments. (pictures here were taken from [3])

- The description oof the implementation pipeline (what you already did and what is planned in the time left)

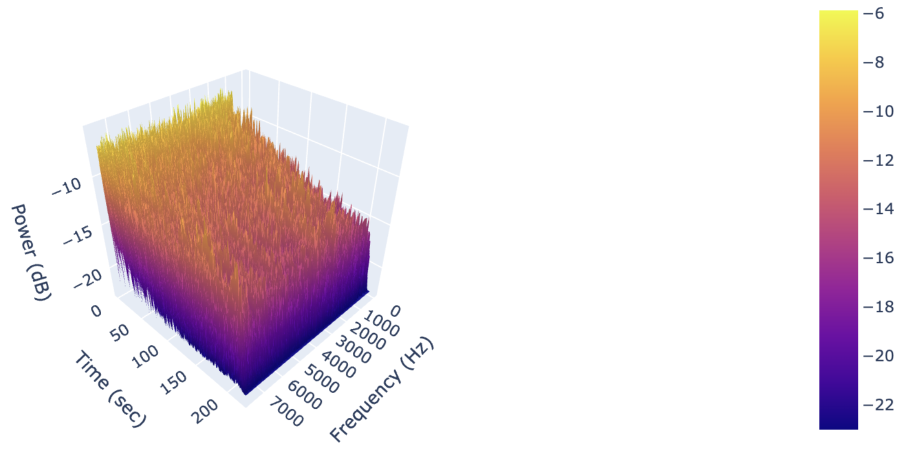


Activity	Start time	End time	Status	Activity Type
Research work	Apr 2	Apr 9	Done	Group work
Topic change	Apr 10	Apr 14	Done	Group work
Algorithm searching	Apr 17	Apr 24	Done	Group work
Team discussion about implementa...	Apr 26	Apr 26	Done	Meeting
Dividing coding tasks	Apr 27	Apr 27	Done	Meeting
Audio processing	Apr 28	Apr 29	Done	Personal work
Visualization	Apr 29	May 1	Done	Personal work
Hashing implementation	May 1	May 4	Open	Personal work
Adding audio recording	May 1	May 8	Open	Personal work
Implementing denoising	May 2	May 8	Open	Personal work
Web application development	May 8	May 15	Open	Personal work
Testing	May 15	May 18	Open	Personal work
Interim report 3	May 18		Open	Group work

As one can see from the pipeline above, we have done much researching on the topic and discussed its implementation. Currently, we are just analysing the song using FFT and its frequency domains and spectrograms.



Specgtogram 3d of first 70 seconds of the song



Spectrograms are very useful for analysing signals because they provide a visual representation of the frequency content of a signal over time. This allows us to see how the spectral content of a signal changes over time and can reveal necessary information about the underlying sound source.
The tasks like hashing, denoising and web application development with the blue flag “Open" are still ahead of us but will be accomplished successfully.

Sources:

1. *https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf*
2. *https://youtu.be/Arni-zkqMBA*

3. https://www.cameronmacleod.com/blog/how-does-shazam-work
4. https://www.toptal.com/algorithms/shaz∞am-it-music-processing-fingerprinting-and-recognition
5. https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/
6. https://www.royvanrijn.com/blog/2010/06/creating-shazam-in-java/
7. https://colab.research.google.com/drive/1JZG_l_smnx8rlwXLmy8gcTma2KTTcPl8?usp=sharing