Music fingeprinting

1st Ilias Alexandropoulos
mtn2302

2nd Vasiliki Rentoula
mtn2317

Abstract—This project presents the development of an audio fingerprinting system for song recognition, implemented as part of the Multi Model course in the ”MSc in AI” program at Unipi & NCSRD. Using the Dejavu library, the system pre-process YouTube URLs to determine whether the corresponding videos contain any songs from a predefined list and identifies the duration for which each song if it is present. This system improves the audio fingerprinting technology to show more accurate and efficient music recognition within video content.

Index Terms—Audio Fingerprinting , Song Recognition, Dejavu Library , Duration Detection

I. Introduction

The scope of this exercise is the creation of an audio fingerprinting system for song recognition, as part of the Multi Model course at ”Msc in AI of Unipi & NCSRD”. We achieved this by using a Dejavu library to create a system that takes a YouTube URL and returns whether the corresponding video contains a song from a predefined list and, if so, the duration for which the song is present.

II. Background

A. Audio Fingerprinting Technology

An audio fingerprint is a condensed digital summary, a digital fingerprint, deterministically generated from an audio signal, that can be used to identify an audio sample or quickly locate similar items in a music database.[1] Applications of audio fingerprinting include song identification, melodies, tunes, or advertisements; sound effect library management; and video file identification. Media identification using audio fingerprints can be used to monitor the use of specific musical works and performances on radio broadcast, records, CDs, streaming media, and peer-to-peer networks. This identification has been used in copyright compliance, licensing, and other monetization schemes. [1]

Audio fingerprinting works by creating a unique identifier for a piece of audio content based on its acoustic properties. Additionally, every audio file is “fingerprinted in which hash tokens are extracted. Both “database” and “sample” audio files are subjected to the same analysis. The fingerprints from the unknown sample are matched against a large set of fingerprints derived from the music database. [2]

Moreover, to achieve a good audio fingerprint system emphasize should me in the different parameters of the audio fingerprint system.[3] This parameter includes:

- Robustness: To achieve high robustness, the fingerprint should be based on perceptual features that are invariant with respect to signal degradations. Preferably, severely degraded audio still leads to very similar fingerprints. The false negative rate is generally used to express the robustness. A false negative occurs when the fingerprints of perceptually similar audio clips are too different to lead to a positive match.
- Reliability: The system should minimize the false positive rates.
- Granularity: Refers to the time needed for the song identification.
- Search speed and scalability: Refers to the search duration of the database to find the fingerprint. Search speed should be in the order of milliseconds.

B. Introduction to Dejavu Library

Dejavu is an open-source audio fingerprinting library that allows for identifying and matching audio content. It is a popular choice due to its ease of use and effectiveness. Comparison with other libraries reveals its strengths and limitations.

In our project, we focus on music fingerprinting, a technology used to identify songs from audio snippets. It transforms audio signals into unique identifiers (fingerprints) that can be matched against a database of known songs. The process starts with transforming the music into a digital signal. Before preprocessing, the audio is discretized by sampling. By default, Dejavu samples the signal with a frequency of 44.1 kHz, meaning that 44,100 samples of the signal are extracted per second. This sampling frequency is justified using the Nyquist-Shannon sampling theorem, which states that the signal should be sampled at double the maximum frequency we aim to capture. Since humans generally cannot hear frequencies above 20,000 Hz, the maximum frequency was established at 22,050 Hz, ensuring that no human-audible frequencies would be missed when sampling. Thus, Dejavu’s default sampling frequency is double the maximum frequency, or numerically, 2 * 22,050 = 44,100 Hz.

After converting the music files into a digital signal, Dejavu generates spectrograms using Fast Fourier Transform (FFT) over short windows, which visualize the amplitude of frequencies over time. Next, peak finding is performed. A peak is a time/frequency pair corresponding to an amplitude value that is the greatest in a local "neighborhood" around it. This involves identifying peaks in the spectrogram using image processing techniques, creating robust points of interest that are resistant to noise. Then, peaks are combined into fingerprints using hash functions, which uniquely identify parts of the song.

Finally, the fingerprints are stored in a database with fields for the hash, song ID, and offset. When recognizing a song, audio is captured, processed, and fingerprints are matched against the database. The song ID with the most aligned matches is identified as the song being played.

C. Related work

In this section, we focus on the research works performed in the audio fingerprinting area. In [2] introduces the firstly edition of the Shazam algorithm, a method for audio fingerprinting that identifies songs from short audio samples. It uses a hashing technique where the key is the frequency to create unique fingerprints for fast and accurate music recognition, that leads to fewer hash collisions improving the performance of the hash table. In [5], Locally Linear Embedding extracts audio fingerprints from the audio recording used to obtain the smaller fingerprint by mapping the energy vector to a lower dimension. This method shows excellent retrieval performance in both multiple and single group reduction. In addition, the fingerprint reduction ratio also reached a maximum of 27%, which helps reduce the fingerprint data significantly. The techniques used in paper [6] employ a space-saving algorithm. After the primary fingerprints are sub-sampled and extracted, only a part of the original data is stored thus saving space. This approach helped significantly decrease the memory required to store the fingerprints and increased reliability and robustness. However excisting systems struggle with accuracy in challenging conditions, limiting broad applicability. This research proposes an AI and ML integrated audio fingerprinting algorithm to enhance accuracy.

III. Methodology

A. System Overview

The system architecture consists of several components, including audio extraction, fingerprint generation, and matching in order to find the duplicate song.

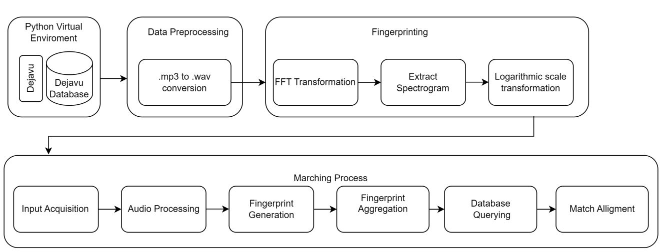

Figure 1. Components Description

B. Setting Up the Environment

To set up the environment, several tools and libraries are required, including Python and Dejavu. First after setting up the python virtual environment, a crucial step is to setup the MySQL database and create a database called dejavu, to store the songs among with the corresponding fingerprints. Also, ffmpeg is needed for audio preprocessing and youtube-dl command to download audio from youtube videos.

C. Data Preprocessing

We downloaded the FMA small dataset for testing the tool. After the download is complete a conversion from .mp3 to .wav files is needed because wav files use lossless compression or can be uncompressed, meaning they retain all the original audio data and provide better audio quality.

D. Fingerprinting

First the fingerprinting can be start either from giving a wav file or a directory of wav files. Then, the audio samples are converted into the frequency domain using the Fast Fourier Transform (FFT) (Figure 2). This is done by segmenting the audio signal into small overlapping windows and applying the FFT to each window. The size of these windows and the degree of overlaping is given by the parameters wsize and wratio, respectively. The result of this process is a spectrogram, which shows how the frequency content of the audio signal varies over time.

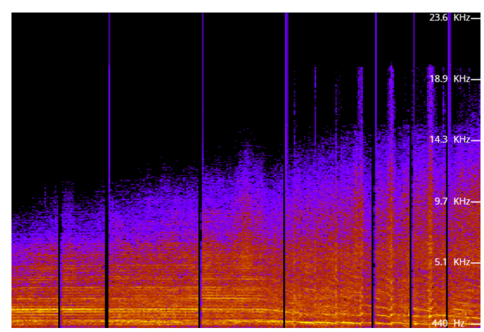

Figure 2. Spectogram example

Next, the spectrogram is transformed using a logarithmic scale (Figure 3). This step is necessary because the output of the FFT is in a linear scale, but the human ear perceives sound in a logarithmic manner. By converting the spectrogram to a logarithmic scale, the data becomes more suitable for further processing and analysis.

Then, it identifies peaks in the spectrogram. These peaks represent the most important frequencies at different points in time. It uses the 'amp_min' parameter to set a threshold to filter out less significant peaks, ensuring that only the most relevant features are considered. This can be modified according to each task, for example if you want to reduce the number of fingerprints you need to raise this value. For our task we used $amp_{min} = 10$.

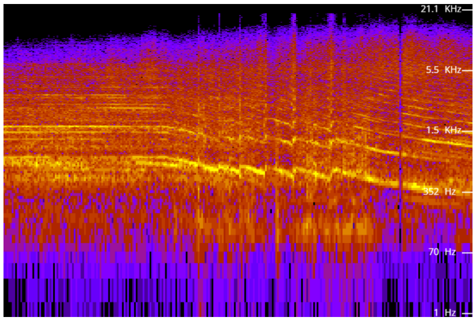

Figure 3. Spectogram after logarithmic scale

Finally, the identified peaks are used to generate hashes. Each peak is paired with its neighboring peaks within a certain range, the range that have been used is 5. These pairs of peaks form unique combinations that are then converted into hashes. Each hash includes information about the specific frequencies and the time offset between the peaks, allowing for robust identification of the audio sample even in the presence of noise or distortions.

E. Fingerprinting and Matching Process

Firstly, to recognize an audio or a song from the pre-fingerprinted audios you can use the path directly to the wav file or a youtube link. After that, it extracts the audio channels and the sampling rate from the file, dividing the audio data into separate channels. Next, each audio channel undergoes fingerprint generation. The generate fingerprints process applies the fingerprinting algorithm that we described in section III-D to convert the audio samples into a set of unique hashes.

Once the fingerprints are generated from all channels, they are combined into a single set to eliminate duplicates, ensuring that each unique fingerprint is considered only once.

The combined set of fingerprints is queried against the database of pre-fingerprinted audio tracks and searches for matching fingerprints in the database and records the query time, resulting in a list of potential matches along with their corresponding hashes. After identifying potential matches, the align matches process is next. This process aligns the potential matches with the original audio sample, ensuring that the temporal relationships between the fingerprints are consistent with those in the database.

Finally, the results are aggregated. The total time taken for the recognition process, including the time spent on fingerprinting, querying, and aligning, as well as the matched results, are collected into a dictionary for reporting purposes.

F. Handling Edge Cases

To improve the accuracy of audio fingerprinting and mathcing fingerprints, specific strategies are employed to address common issues such as noise, overlapping audio, and partial matches.

As mentioned in the previous section for noise reduction, logarithmic transformation is applied after computing the spectogram in the fingerprint process. This achieves, to mitigate the impact of noise by focusing on how humans perceive sound by adjusting the linear scale of the FFT output to a logarithmic scale. Also an other implementaiton of noise reduction is in the identification of peeks of the spectrogram process, where the amp_{min} parameter sets a minimum amplitude threshold. This threshold filters out less significant peaks, allowing the algorithm to concentrate on the most prominent features of the audio signal. By ignoring weaker signals, the process becomes more resistant to background noise.

G. Find duplicates

The process begins, to fingerprint all audio files within the directory. Fingerprinting involves analyzing each .wav file to extract unique acoustic features, transforming these features into a set of digital identifiers (fingerprints) as explain in Section III-D. The system then iterates through each file in the directory, focusing on those with a .wav extension. For each file, it constructs the full file path and proceeds to recognize the file by comparing its fingerprints against a database of pre-existing fingerprints. This comparison process aims to find matches between the fingerprints of the current file and those already stored in the database. During the recognition process, the system evaluates the confidence levels of the matches. It specifically looks for matches where both the input audio and the fingerprinted audio have confidence levels of 0.90 or higher. This high threshold ensures that only reliable matches are considered potential duplicates, minimizing false positives. For each high-confidence match, the system decodes the song name and logs the duplicate information in the dictionary. If a song name is encountered for the first time, it initializes an entry for that song in the dictionary. The file name, input confidence, and fingerprint confidence are then recorded under this entry, creating a comprehensive log of all identified duplicates. After processing all files, the system compiles the results. If duplicates are found, it prints a detailed list of these duplicates, including the song names and corresponding files with their confidence levels. If no duplicates are detected, it informs the user that no duplicates were found.

IV. Results and Analysis

A. Testing and Evaluation

Metrics such as accuracy, precision, recall, and F1-score are used to evaluate the system's performance.

V. Conclusion and Future Work

One promising direction for future work is the integration of Deep Learning (DL) techniques. Instead of relying only on the Dejavu library for audio/music fingerprinting, employing DL models, such as those based on contrastive learning, can enhance the system's ability to extract relevant information from short audio fragments. Contrastive learning, in particular, focuses on learning features by comparing similar and dissimilar pairs of data, which can lead to more robust and discriminative audio representations. Additionally, the use of more sophisticated noise reduction and signal enhancement techniques can further improve the accuracy of the fingerprinting process. For instance, implementing advanced denoising algorithms and leveraging the power of neural networks to clean audio signals can result in clearer and more distinct fingerprints, thus improving the system's performance in noisy environments.

References

[1] 1 https://en.wikipedia.org/wiki/Acoustic_fingerprint
[2] https://www.ee.columbia.edu/ dpwe/papers/Wang03-shazam.pdf?utm_source=buffer
[3] https://ismir2002.ismir.net/proceedings/02-FP04-2.pdf
[4] https://core.ac.uk/download/pdf/189234543.pdf
[5] Jia, Maoshen, Tianhao Li, and Jing Wang. "Audio Fingerprint Extraction Based on Locally Linear Embedding for Audio Retrieval System." Electronics 9, no. 9 (2020): 1483.
[6] Yang, Guang, Xiaoou Chen, and Deshun Yang. "Efficient music identification by utilizing space-saving audio fingerprinting system." In 2014 IEEE International Conference on Multimedia and Expo (ICME), pp. 1-6. IEEE, 2014.
[7] Zhang, Xueshuai, Ge Zhan, Wenchao Wang, Pengyuan Zhang, and Yonghong Yan. "Robust audio retrieval method based on anti-noise fingerprinting and segmental matching." Electronics Letters 56, no. 5 (2019): 245-247.
[8]