(IJIEST) 2016, Vol. No. 2, Jan-Dec **e-ISSN: 2454-9584; p-ISSN: 2454-8111**

'AN IN-DEPTH ANALYSIS OF SPEECH PERCEPTION AND AUTOMATIC INDEXING, ALIGNING OF AUDIO RECOGNITION IN DEVELOPING A CORPUS FOR THE DETECTION AND CLASSIFICATION OF ACOUSTIC SCENES AND EVENTS USING PYTHON'

Diya Mawkin

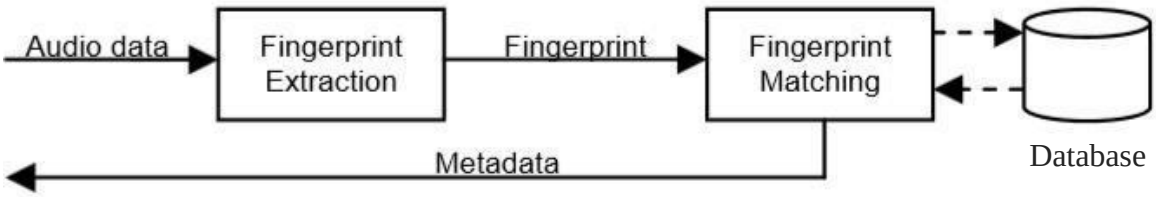
B.tech Computer science ,Jaypee University of Engineering and Technology, Guna

INTRODUCTION

Recognizing an audio or song from a library of songs with a sample, even 10 seconds, would be very useful for users. We are tackling this problem, by building a software and service which could recognize songs from samples recorded even in ambient noise areas. It involves a complex procedure of acoustic fingerprinting of audio.

It starts when we will take the input i.e. recorded sample from the user and generate spectrogram based on the audio sample, followed by a process of fingerprinting the sample. Beforehand, we have catalogued a library of fingerprinted songs.

Generated acoustic fingerprint is searched through a search algorithm and pattern matching using hashing. The metadata in the database which will match the fingerprint of the given sample will be presented as output to the user.



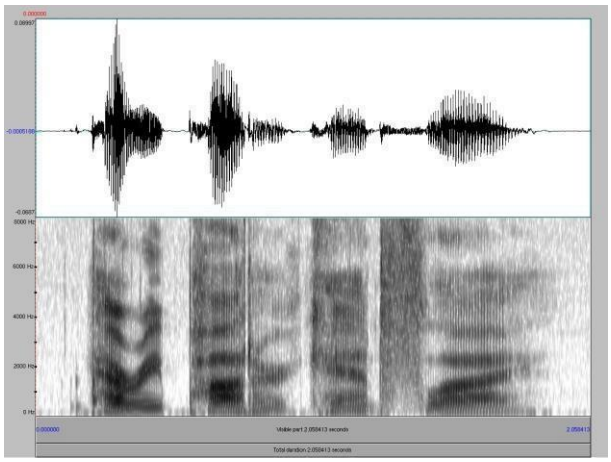
Proposed Algorithm

The input for our software would be a sample of about 10 - 20 seconds whose fingerprint we would generate taking into consideration all the possible error aspects such as external noises and transmission loss. The fingerprint will be stored in the table in a 32 bit format consisting of Hash value which is the frequency and time offset. Time offset is the time from starting of the track to the certain frequency and it helps us to determine the sample from any point on the actual audio fingerprint placed in our database.

We used the concept of **Anchor Point** which helps in speeding up the search process as it takes more than one fingerprint point and put it into single point such that by processing one point we process more than one point. This also helps in better and accurate results for the user.

Recognition of a song from a sample is hard problem to solve on its own, it must be noted that the users can actually get the Meta data about the song playing nearby to them or on some broadcast medium near them(If the song is fingerprinted and saved in the database). Plus, a lot of the time users would want to know the name of the song they just listened to and liked but don't want to go through the hassle of actually asking people around for the name of the song, but with this software they don't need to ask anyone about the song, they can just record the sample audio for around 10 – 20 seconds and get the meta data about the song presented to them.

Spectrogram: A spectrogram is a visual portrayal of the range of frequencies in a sound or other flag as they differ with time or some other variable. Spectrograms are in some cases called otherworldly cascades, voiceprints, or voice grams.



Spectrogram of a Sample AudioFile

Fingerprint: An acoustic Fingerprint mark is a dense advanced rundown, deterministically produced from a sound flag that can be utilized to recognize a sound example or rapidly find comparable things in a sound database.

Audio File:A audio document organize is a record arrange for putting away advanced sound information on a PC framework. The bit design of the sound information (barring metadata) is known as the sound coding group and can be uncompressed, or compacted to diminish the recorded measure, frequently utilizing lossy pressure.

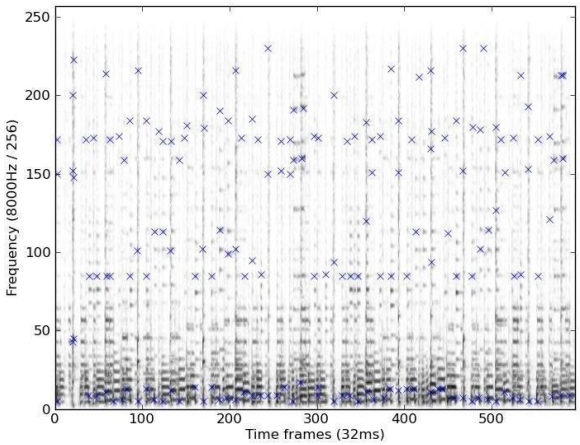
Noise: Noise is an assortment of sound. It implies any undesirable sound. Sounds, especially noisy ones that exasperate individuals or make it hard to hear needed sounds, are commotion

Hash Tables: In processing, a hash table (hash outline) an information structure used to actualize an acquainted exhibit, a structure that can delineate to values. A hash table uses a hash capacity to register a record into a variety of basins or spaces, from which the ideal esteem can be found.

Metadata: Metadata is data that depicts other data. Meta is a prefix that in most information development uses means "a concealed definition or depiction." Metadata diagrams fundamental

information about data, which can make finding and work with explicit instances of data less requesting.

Candidate Peak: A period recurrence point is an applicant crest on the off chance that it has a higher vitality content than every one of its neighbours in a locale fixated on the point.



Candidate Peak (A time-frequency graph)

Histogram: A histogram is a graphical portrayal of the circulation of numerical information. It is a gauge of the likelihood dispersion of a ceaseless variable (quantitative variable).

FFT:A snappy Fourier change (FFT) estimation enrols the discrete Fourier change (DFT) of a gathering, or its turn around. Fourier examination changes over a banner from its exceptional territory (every now and again time or space) to a depiction in the repeat space and a different way.

ABOUT PAPER

Our Paper's aim is to understand and implement an application which can recognize songs from short samples parts of songs. Shazam is just one possible audiofingerprinting implementation. This application utilizes basic counts that require a basic rationale and factual information to comprehend makes it workable for a million of its clients to perceive the melody they hear, regardless of where they are and what sort of a situation it is. The Shazam calculation can be utilized in numerous applications other than just music acknowledgment over a cell phone. Because of the capacity of the application to delve profoundly into commotion, we can recognize music taken cover behind an uproarious portrayal, for example, in a radio declaration. Then again, the calculation is likewise quick and can be utilized for copyright checking at a pursuit speed of more than multiple times continuous. The calculation is likewise reasonable for substance based signaling and ordering for library and documented employment.

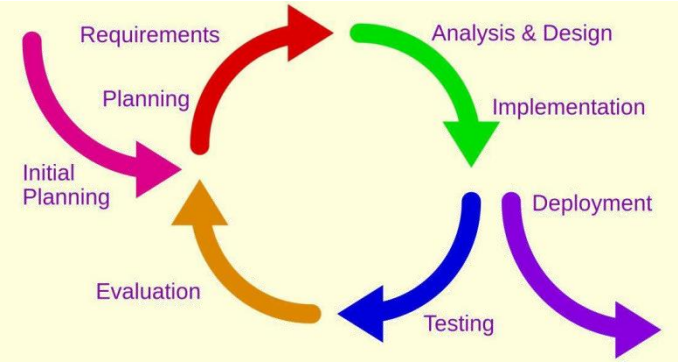
(IJIEST) 2016, Vol. No. 2, Jan-Dec **e-ISSN: 2454-9584; p-ISSN: 2454-8111**

Requirements

Paper requirements includes a microphone (to capture the sample audio), A platform to work upon, preferably Windows OS, Database to store the fingerprints etc.

SDLCModel

We will utilize the Iterative model. An iterative lifecycle show does not endeavor to begin with full detail of necessities. Rather, improvement starts by indicating and actualizing simply part of the product, which is then inspected so as to recognize further necessities. Underneath figure, demonstrates an essential procedure for iterative lifecycle display.



SDLC of Iterative Model

Modules

We divided our work on the Paper into various modules, completing each module, till we had the final application.

Module 01: Designing the basic structure of application.
Module 02: Implementation of Algorithm and connecting the application with a database.
Module 03: Develop a GUI based application.
Module 04: Testing and debugging

Programming Language

To design and develop such an application, we decided to develop our Paper in Python because of its huge array of scientific calculation library available for it.

When considering what language to prefer for developing our Paper upon, we came up with a list of languages that we already had some knowledge and previous use in and ranked them according to our familiarity with them, suitability for digital signal processing and availability of third party libraries and general support. From this the top 5 languageswere:
Python, Java, C++, Ruby and C

Our team's familiarity with Python and the huge amount of scientific calculations libraries available for audio processing for it, makes it a better choice in terms of programming our application as opposed to other languages.

C++ also has a lot of support and libraries available, and could be developed in for cross platform. However we am not experienced enough to commit to developing this Paper in it. Ruby does have support for audio processing. But it falls into the some pitfalls and Python with speed and pointless high level features.

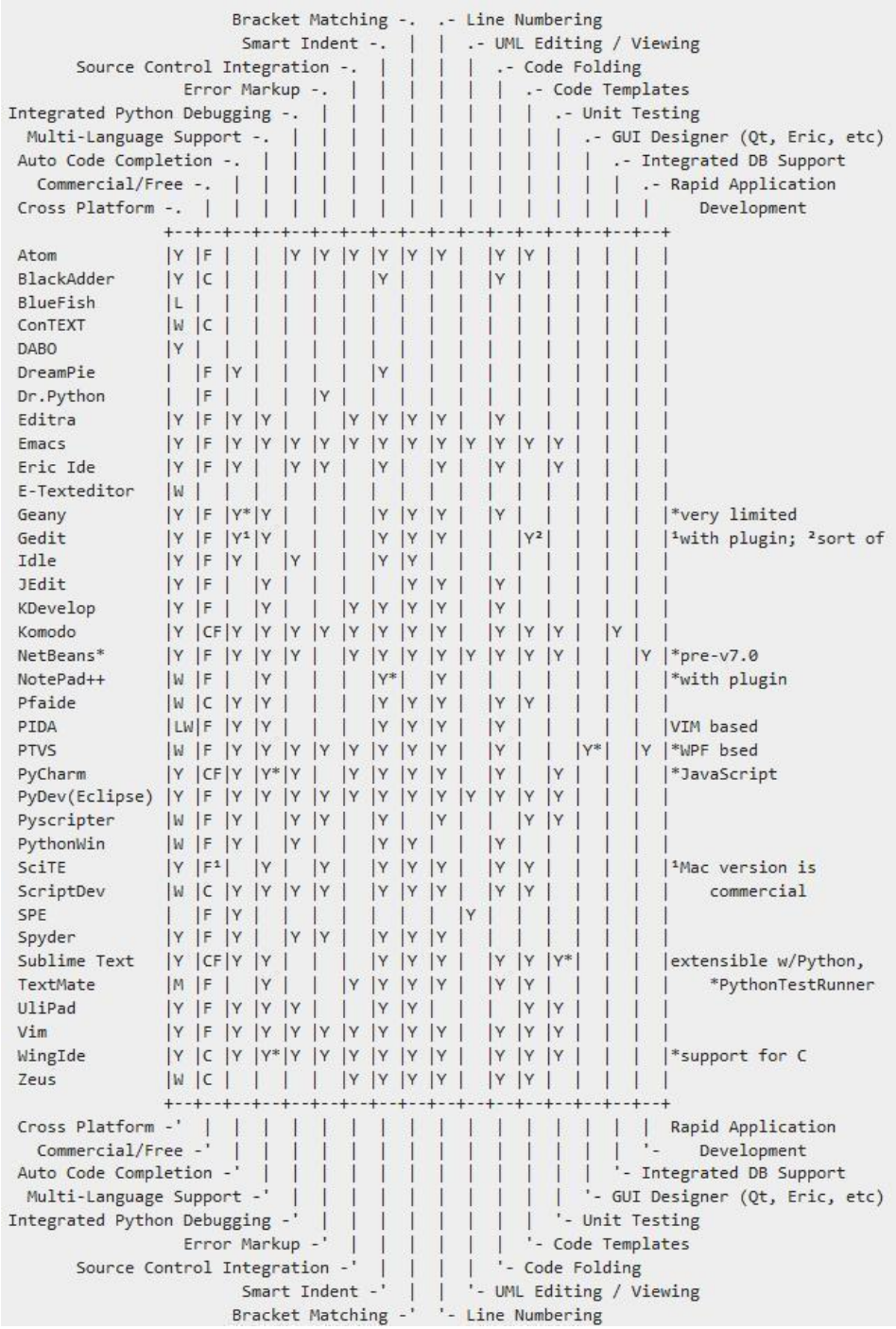
C seemed like it could be the better choice due to low level support which would help make for optimized audio data handling and processing. However, some of the higher level processing to be done, such as image rendering and Graphical User Interfaces, are not so simple to do.

While we already know C fairly well, with the time constraints of the Paper and the scope of development we wanted to undertake, it did not seem like a better choice over Python.

IDE

For choosing our Integrated Development Environment, we looked for various IDE's available for python and features they provide and finally choose PyCharm for its greater support for python and stability. Here is a comparison between PythonIDE's.

P hon IDE's Comparison



y

L – Linux C – Commercial CF–CommercialwithFree limited edition

F – Free W – Windows ? –Tobeconfirmed M – Mac

t

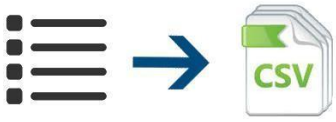
DATABASE

Now there are a lot of solutions available for storing such sort of data on databases, options that sprang to mind were MySQL, PostgreSQL or SQLite. MySQL and PostgreSQL both have similar features and are both suitable.

Oracle might be free for this use, but requires a fairly complex setup. While Oracle has the most features, it has nothing that this Paper requires that another database system can fulfil and it requires the overhead of running the oracle server. SQLite has enough features for use in this system.

Between MySQL, PostgreSQL and SQLite, SQLite stood out as the best choice due it its small size, efficiency, available libraries for java and documentation. As this system doesn't require a server/client setup, a capacity for high volumes of connections and queries, a large dataset or high concurrency then SQLite is fine. MySQL and PostgreSQL would also suit the job, but require a lot more processing power and are made for purposes more complex than our uses. But that database choice is for latter Paper development.

Since for demonstration purposes, we used comma-separated values (CSV) file that stores tabular data (numbers and text) in plain text and a simple library textual file to order and index the songs stored in our CSVfile.



A Representation of Data to CSV

UML

A common method of modelling software during the design stage is to design UML diagrams. For our system there is a relatively simple use case diagram as the software isfairly compartmentalized and doesn't link to anything external besides the user and system's backend.
Below figure, outlines an overview of the systems functionality.

UML also has a type of diagram to help developers design how processes interact with each other, a

sequence diagram. We designed two of these to help us refine the sequence of operations that occur when a user fingerprints a file, searches for it in the database or stores it in the database.

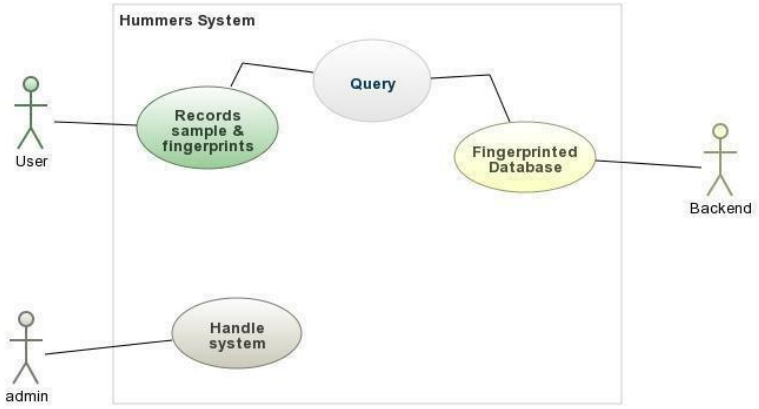
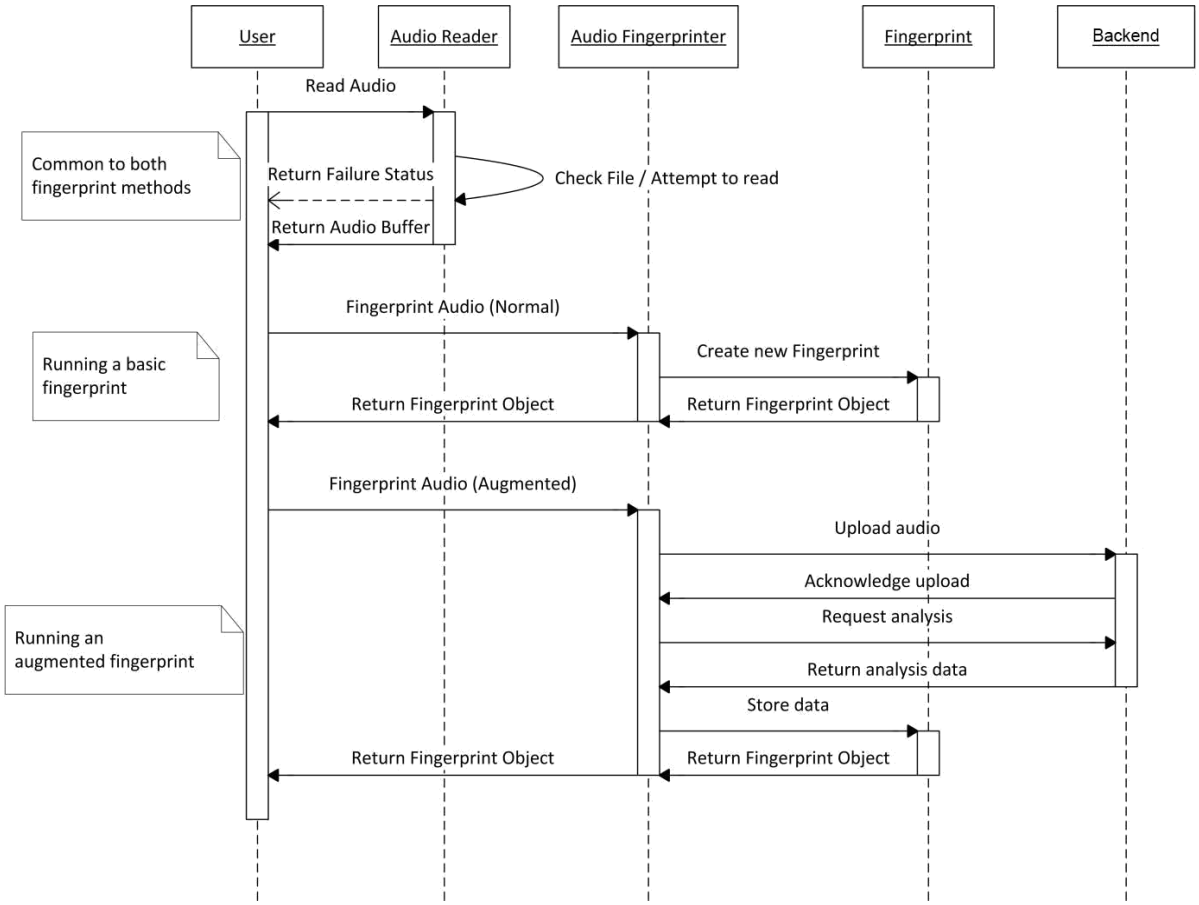


Figure on the next page demonstrates the sequence diagram outlining the fingerprinting process for both normal and augmented fingerprints.



Sequence diagram of fingerprinting process

(IJIEST) 2016, Vol. No. 2, Jan-Dec **e-ISSN: 2454-9584; p-ISSN: 2454-8111**

Third Party Libraries

To analyze the audio samples and compute the other factors needed for our augmented fingerprint we had to utilize other third party libraries for python to generate spectrograms and other wave related computation. Hence, it was best we utilize third party libraries for some of the low level computation. Some of the libraries that were used in development of the system are follows.

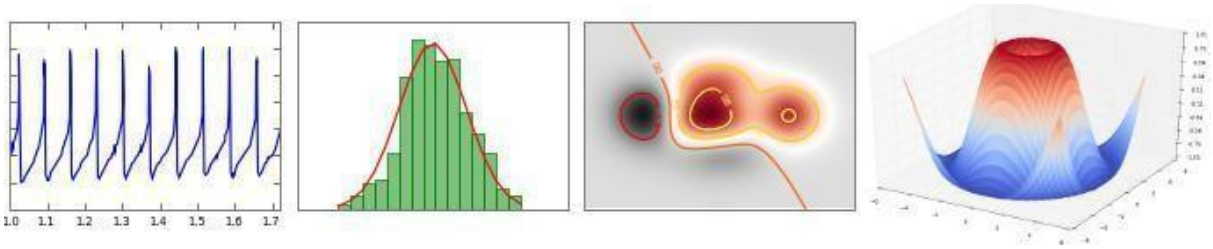
PyAudio: PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux , Microsoft Windows.

NumPy: NumPy is the principal bundle for logical registering with Python. It contains in addition to other things:
an amazing N-dimensional cluster object
modern (telecom) capacities
tools for incorporating C/C++ and Fortran code
valuable straight variable based math, Fourier change, and irregular number capacities

Other than its conspicuous logical utilizations, NumPy can likewise be utilized as an effective multi-dimensional compartment of nonexclusive information. Self-assertive information types can be characterized. This permits NumPy to flawlessly and rapidly incorporate with a wide assortment of databases.

SciPy: SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:
NumPy, SciPy library, Matplotlib, IPython and pandas.

Matplotlib: matplotlib is a python 2D plotting library which produces distribution quality figures in an assortment of printed version groups and intelligent conditions crosswise over stages. Matplotlib can be utilized in python contents, the python and ipython shell, web application servers, and six graphical UI toolbox.



Demonstration of matplotlib library function

IMPLEMENTATION

To develop the software we divided the implementation work into several key tasks, basing our decisions on the development principles laid out by the Iterative programming development model which calls for small, incremental releases that start by building the core of the system and expanding it to meet requirements as it is developed.

Our main goal is to implement a system which can detect songs from short samples of songs. The implementation involves some complex components, working together as an application.

AcousticFingerprinting

Music is carefully encoded as only a not insignificant rundown of numbers. In an uncompressed .wav record, there are a ton of these numbers - 44100 every second for every channel. This implies a 3-minute long melody has right around 16 million examples.

3 min * 60 sec * 44100 examples for every sec * 2 channels = 15,876,000 examples

A channel is a different succession of tests that a speaker can play. For instance, having two ear buds - this is a 'stereo', or two channel, setup. A solitary channel is called 'mono'. Today, present day encompass sound frameworks can bolster a lot more channels. In any case, except if the sound is recorded or blended with a similar number of channels, the additional speakers are excess and a few speakers will simply play an indistinguishable stream of tests from different speakers.

Sampling

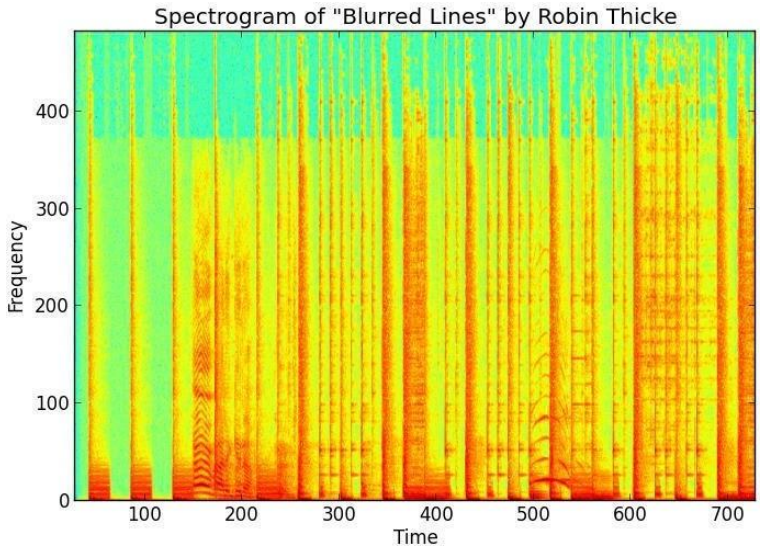
Why 44100 examples for every second. The secretive decision of 44100 examples for every second appears to be very self-assertive, however, it identifies with the Nyquist-Shannon Sampling Theorem. This is a long, scientific approach to state that there is a hypothetical limit on the most extreme recurrence we can catch precisely when recording. This greatest recurrence depends on how quick we test the flag.

On account of chronicle sound, the acknowledged principle is that we're OK passing up frequencies over 22050 Hz since people can't hear frequencies over 20,000 Hz. In this manner by Nyquist, we need to test twice that:

Tests per sec required = Highest-Frequency * 2 = 22050 * 2 = 44100

Spectrograms

Since these examples are a flag of sorts, we can more than once utilize a FFT over little windows of time in the melody's examples to make a spectrogram of the tune. Here's a spectrogram of an initial couple of moments of "Blurred Lines" by Robin Thicke.

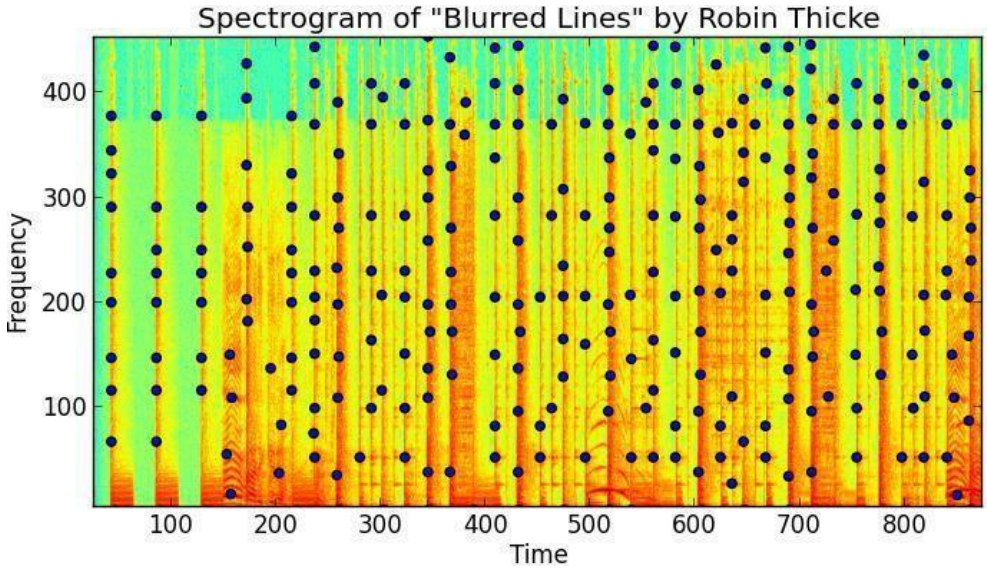


Spectrogram of the song "Blurred Lines" by Robin Thicke

As you can see, it's just a 2D array with amplitude as a function of time and frequency. The FFT shows us the strength (amplitude) of the signal at that particular frequency, giving us acolumn. If we do this enough times with our sliding window of FFT, we put them together and get a 2D array spectrogram.

PeakFinding

Since we have a spectrogram of our sound flag, we can begin by discovering "tops" inadequacy. We characterize a top as a (period, recurrence) match comparing to a plentifulness esteem which is the best in a nearby 'neighborhood' around it. Other (time, recurrence) matches around it brings down inadequacy, and in this manner less inclined to endure the noise.



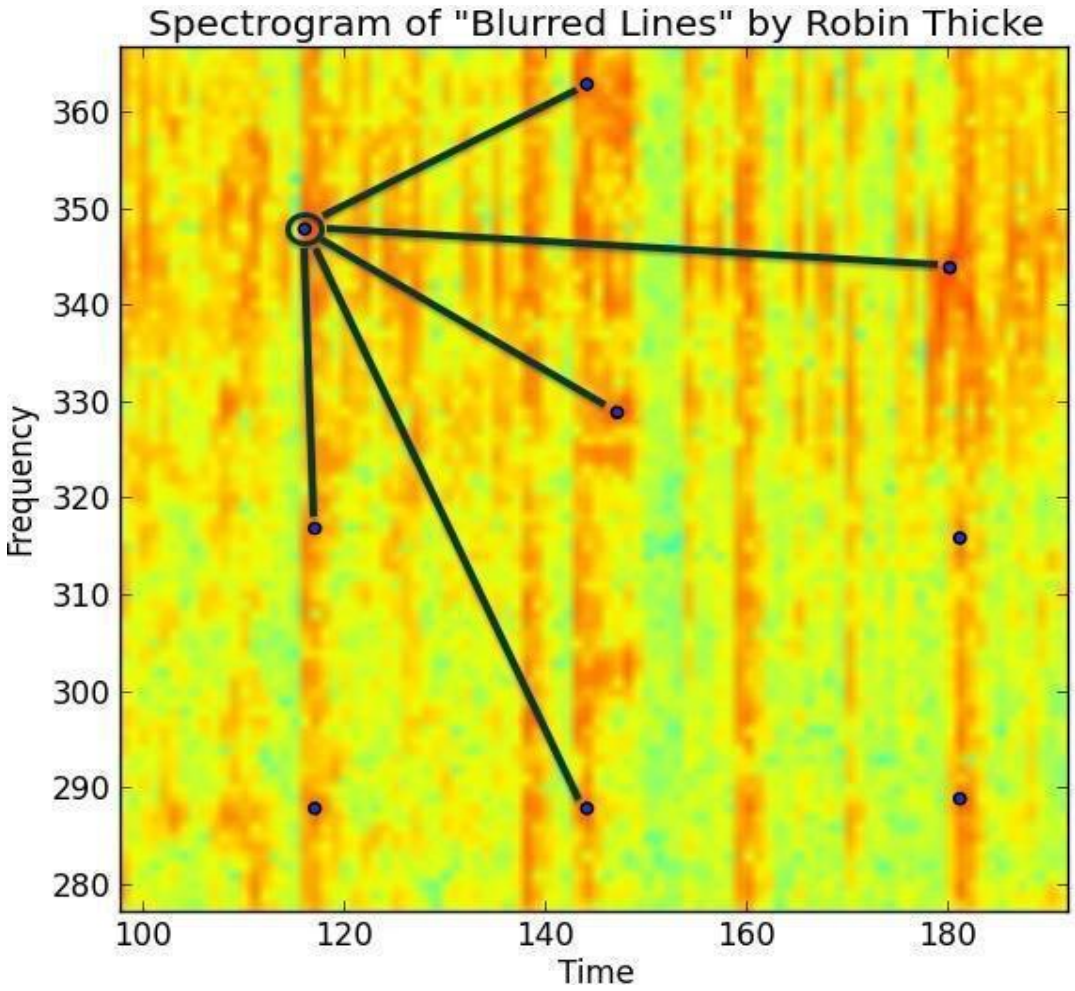
Spectrogram with Peaks outlined for Song by Robert Thicke

When we have removed these clamor safe pinnacles, we have discovered focal points in a tune that recognizes it. We are viable 'squashing' the spectrogram down once we've discovered the pinnacles. The amplitudes have filled their need, and are never again required.

FingerprintingHashing

A hash work takes number info and returns another whole number as yield. A decent hash capacity won't just restore a similar yield number each time the info is the equivalent, yet adds that not many distinctive information sources will have a similar yield.

By taking a gander at our spectrogram pinnacles and consolidating crest frequencies alongside their time distinction between them, we can make a hash, speaking to a one of a kind unique finger impression for this melody.



Hash (frequencies of pinnacles, the time contrast between pinnacles) = unique mark hash esteem

A Zoomed-in annotated spectrogram

CLUI Application

We started with developing a CLUI (Command-Line User Interface) based application to debug the software and test the various features we want to implement in our final version and also it would act as a backend for final application.

CLUI Application

We designed the interface with basic commands that a user could perform with the application that we are building. As soon as the user runs the application, the library is loaded, and the user is presented with options. The options are listed below.

Match a song from file - Selecting this option will prompt user for a file path, which is the sample of audio, and it will match for it.
Match a song from microphone - Selecting this option will prompt user to record a new sample from the microphone, it will then match the recordedsample.
If you press *'r'* after each match, you can see a list of related songs
If you press *'a'*at the main menu, this would bring up the admin panel, which is preferably should be used by Administrator, for managing the library and testing the code.
Cache current library – Choosing this option function would cache the current library forposterity.
Import directory of songs to library - As described
Remove song from library - Asdescribed
Test match on a file or directory - This allows admin to run a battery of tests on a directory ofsongs.
Force library save - This forces the library tosave.

FingerprintDatabase

For basic implementation, we are using a comma-separated values (CSV) file that stores tabular data.

The hashes are stored in a CSV hash file.

The application uses a CSV file in which it stores the fingerprinted database and a cache folder in which it save a cached CSV file of the library, to make the processing of fingerprints fast. Within our Paper it also have a textual file '*library_list.txt'* which indexes all the songs in the library which arefingerprinted.

We designed a class *library* to retain all information related to the music database, including all of the song fingerprints as well as the hash table. *Library* class when initialed, it loads all the data from CSV file into the *library*object, which contains following elements:

songs - an array of loaded songs filenames
hash - a dict(Python Dictionary) of songhashes.

PaperStructure

We have some major classes in our Paper, which handles a specific part or component of the application. The application works as a framework of these classes, each class have specific functions it needs to perform. Major classes are defined below:

Fingerprint Class – An object to store all data related to audio fingerprints. See APPENDIXE.
Hummers Class – CLUI application handler. See APPENDIXB.
Library Class – A class designed to retain all information related to the music database, including all of the song fingerprints as well as the hash table. See APPENDIXC.
Query Class – A class designed to handle all functionality related to filteringthe music database and finding a song match based on a sample. See APPENDIXD.
Main Class – Web Based application handler, works on Flask and Jinja2framework.

APPLICATION DEVELOPMENT

SoftwareArchitecture

After implementing a basic CLUI application, we started developing a web based application, which would be built on top of our previous implementation of CLUI based song recognizer. As our CLUI application was based on python, we decided it would be if we could make a web based application on python.

We developed a web based application based on following stack of technologies:

Python 2.7, Flask 0.10.1, Bootstrap 3, RecorderJS, NumPy, SciPy, Matplotlib, Git, HTML5, CSS,JS, JQuery

We studied about various frameworks available for python to develop our web based application, and finally decided to use Flask and jinja2 for our architecture because they were lightweight and easier to develop upon. Some notes on the technologies:

Flask: A lightweight Python web framework based on Werkzeug and Jinja.

Twitter Bootstrap: Bootstrap , a sleek, intuitive, and powerful mobile first front- end framework for faster and easier webdevelopment.

Git: Git is a generally utilized source code the board framework for programming advancement. It is a disseminated amendment control framework with an accentuation on speed, information honesty, and support for appropriated, non-direct work processes.

RecorderJS: A plugin for recording/exporting the output of Web Audio API nodes.

JQuery :jQuery is a cross-platform JavaScript library designed to simplifythe client-side scripting ofHTML.

Jinja2 : Jinja2 is a modern and designer-friendly templating language forPython.

HTML5: HTML5 is a markup language used for structuring and presenting content on the World WideWeb.

CSS: Cascading Style Sheets (CSS) is a template dialect utilized for portraying the introduction of an archive written in a markup dialect.

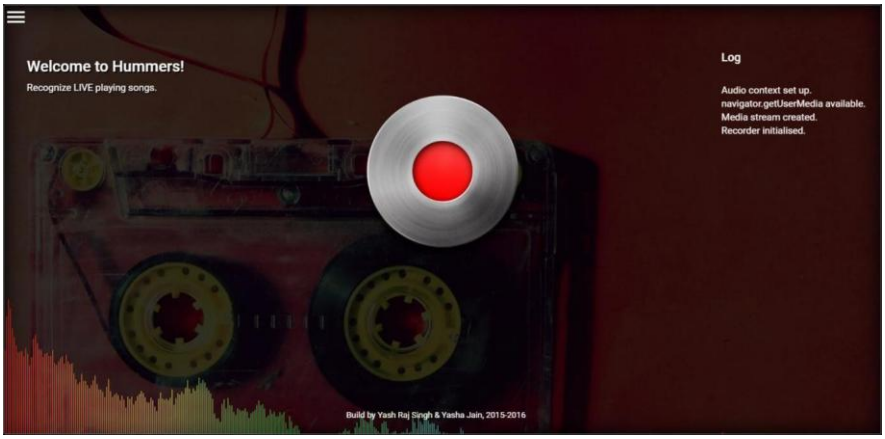
Web Application HandlerCode

We will see more about the python code for the final web based application, it works on basis of the various components we discussed earlier. It works on the Flask and jinja2 Framework for python to render templates.

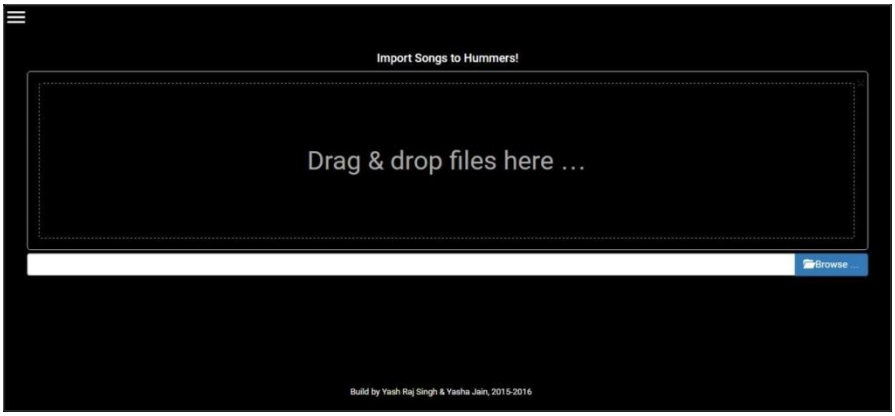
Screenshots

The application went through a lot of design changes as the usability changed and we improved upon the UX of the software.

Screenshots of application build v2



Application Build v2 Dashboard Screenshot



Application Build v2 importer Screenshot

Application Build v2 library Screenshot


Administrator Options
Cache Current Library
Remove All Songs/Fingerprints
Force Save Library & hash to CSV
Build by Yash Raj Singh & Yasha Jain, 2015-2016