MusicSurf

Surajnath Sidh

U101114FCS146

NIIT University

Idea

Why MusicSurf ?

MusicSurf is built to solve the problem of searching in your music collection, not just by title, not just by Artist name because those things can be done using Existing Music players, It does way more then that

It let's you do a full text search on attributes(ID3 Tags) for your music collection, in natural language. It is able to find your music if you only remember that single phrase from lyrics or if you want to listen songs that are "single" or maybe you want to listen all the "Radio edit" or maybe you remember that song you are looking for had 'Draft Punk' as Feat. Artist, It can find that too. After that you got your song, it let's you export search results as a playlist so you can play all those songs in your favorite music player

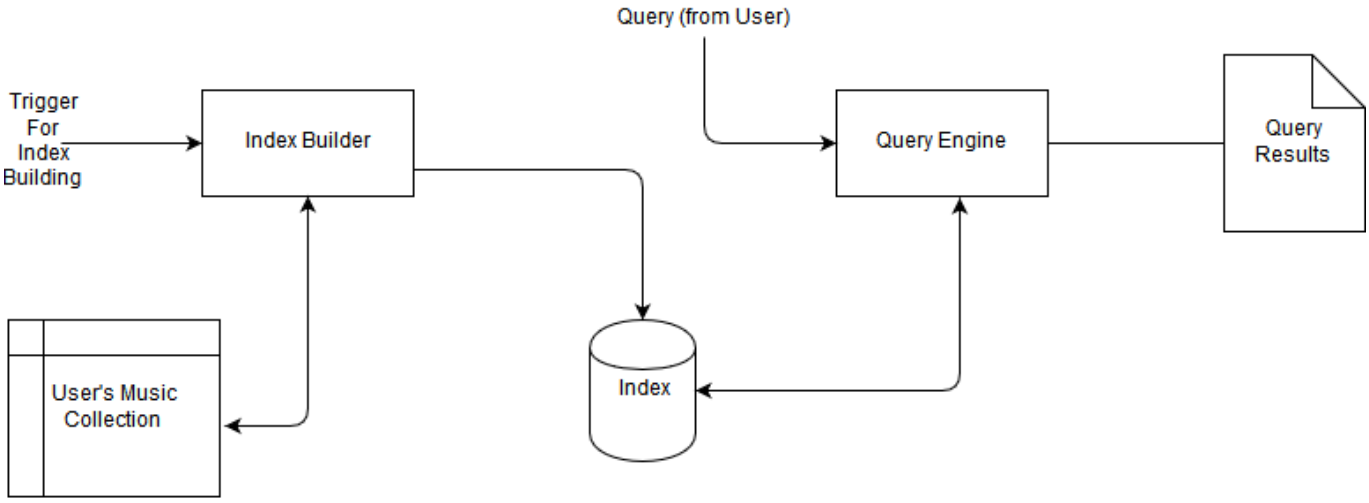
Features

- Natural Language Querying
- Filters
- Export search results as playlist

Design and Architecture

MusicSurf is built using Flask and Elasticsearch, They both were chosen for very specific reasons.

MusicSurf Architecture



MusicSurf : Basic Flow of Control

Flask

Written in Python, Highly Modular, Let's you Build APIs with minimum effort, Can be integrated with any other python modules without any hassle

Elasticsearch

Provides Full text search and have a nice API to index and query data, it is distributed, Highly scalable and best about it is that it is Open Source (Apache 2.0)

Elasticsearch features that we used in our Project

- RESTful API hence easy to integrate with any web service in our case being the Flask
- Query Boosting to implement filters
- Built in Analyzer here analyser is component that does
  - First, tokenizing a block of text into individual terms suitable for use in an inverted index,
  - Then normalizing these terms into a standard form to improve their “searchability,” or recall
- Built in Scoring Function to evaluate and see behavior of system
- Full Text search(This is possible by using inverted index as indexing mechanism)

Contribution

During Architecture (Early Stage)

During out first meeting we had few ideas in our mind first one being the **Building Search server on our own** that takes care of indexing and querying (as it can be seen in architecture Diagram) And plug it with Flask to build the Web Frontend

During the later stage we realized that build and evaluating a search server will a lot of work which can not accomplished by us in time span we have in our hands, the we started looking for OpenSource projects that can help us out in this task, at the end we settled with Elasticsearch. Using an off-the-self Production grade system means we have time to understand a real system

that is used in real products, rather then reinventing the wheel and building our own toy system

It also gives us the time and opportunity to understand a production grade system

Design and implementation process

My Primary contribution was extracting and providing the document object for music collection. I Wrote the id3 module that takes the root folder and crawls the folder recursively, building an JSON list of document objects which then used to index the music.

My document list is indexed by using elasticsearch module written by my team member Rajdeep.

After Indexing is done our system is ready for querying, Which is done using the Web Interface Built by Raghav, Web Interface and Query engine is built using Flask and UI it Built using Google MDL(Material Design Library)

We choose Python as our Programming language to build the system because it is Higher level, Team is familiar with language and availability of Libraries.

ID3 Module

ID3 module utilities the mutagen library to get ID3 data from music files, along with glob to recursively crawl the music directory and JSON library to build the final document list for indexing purposes

ID3 have multiple tags that stores metadata, we used only few of them for our indexing need These are the official ID3 tags used to build this document object

ID3 standard tags Mapping to document object

```
- title --> [TIT2]
- artist --> [TPE1] + [TPE2] + [TSOP]
- album --> [TALB]
- year --> [TDRC]
- lyrics --> [USLT::eng]
- path --> [file-path]
```

Document object for a music file

```
{
    "title": "A Head Full Of Dreams",
    "artist": "Coldplay Coldplay Coldplay",
    "album": "A Head Full Of Dreams",
    "year": "2015",
    "lyrics": "",
    "path": "C:\\Users\\Electron\\Music\\test_music\\Coldplay\\A Head Full Of
Dreams\\Coldplay - A Head Full Of Dreams.mp3"
}
```

This music files does not have lyrics in it hence the lyrics block is empty

ID3 Module is responsible for building a JSON list of these document objects for all the music files.

Elasticsearch internals and configurations

Indexing

Elasticsearch indexes documents in the form of inverted index, which is designed to allow very fast full-text searches. More on Inverted index & how Elasticsearch index documents

Analysis process

When we index a document, its full-text fields are analyzed into terms that are used to create the inverted index. However, when we search on a full-text field, we need to pass the query string through the same analysis process, to ensure that we are searching for terms in the same form as those that exist in the index.

Analysis process is responsible for tokenizing a block of text into individual terms suitable for use in an inverted index, Then normalizing these terms into a standard form to improve their "searchability," or recall.

We configured Elasticsearch to use `Standard analyzer`, It splits the text on word boundaries, as defined by the Unicode Consortium, and removes most punctuation

Before analysis `"Set the shape to semi-transparent by calling set_trans(5)"` **After Analysis** `set, the, shape, to, semi, transparent, by, calling, set_trans, 5`

Querying

When Elasticsearch gets an query it goes thorough same analysis process as documents went during indexing process, to ensure that we are searching for terms in the same form as those that exist in the index.

Contribution Graph

Contributions can be verified on GitHub and can be seen in Contribution graph

GitHub Repository of project is located at https://github.com/electron0zero/MusicSurf

My Contribution

Project Progress