Music Search++

Jonathan Westerfield
jgwesterfield@tamu.edu

Matthew Broman
broman334@tamu.edu

April 25, 2019

Abstract

Spotify uses a music algorithm in order to group songs into music playlists. This allows a user to simply choose a song and Spotify will choose subsequent songs to play after the current one is finished. However, this algorithm is not very effective at choosing songs that are similar to the one being played. The algorithm looks at the genre and relies on other users groupings and playlists. Often times, the songs are not similar but are still in the same genre, which can lead to a very jarring transition from a quiet, slow song to a louder, quicker one. Examples of this can be *Come Sail Away* by The Styx transitioning to *The Grand Illusion* by the same band. These two songs are in the same genre, the same band, and even in the same album, however, the two songs are drastically different in style and volume dynamics.

1 Objective

Our objective is to modify the current Spotify search engine to improve playlist lineups. We will still take into account the genre, artist and album of a song but do some extra analysis to get more information about a song file. We will then create a playlist based off of the song metadata of a single song so that we can create a playlist of songs similar to the one we analyzed.

2 Implementation Breakdown

2.1 Plan

1. Crawl through Spotify's songs
2. Store the songs
3. Implement ability to search for songs individually or by artist
4. Implement ability to create playlist from song that was searched for

2.2 Implementation

1. We got Spotify's songs using the Spotify API
2. We stored all of the song information into individual json files
3. We then uploaded all of our json song files to a Solr Search Engine Instance
4. We use the Spotify API to search for songs

3 Tool Choices

3.1 Solr

We chose to use Solr for several reasons. First, it provides robust and efficient text searching. Solr returns search results based on text, making it very easy to implement search functionality based on the numerous parameters in the song data. Solr can perform a search based on any or all fields of a song. In addition, it is very quick and efficient to search based on certain parameters. Another surprising and convenient feature of Solr is the efficient storage of song data. We pulled json song data from Spotify for over numerous playlists. This was just information and not actually the song audio. However, when we uploaded it to Solr, its final size for our instance only came out to under 70 MB for over 91,000 songs. We created an Amazon AWS EC2 instance (a cloud virtual machine) and installed Solr on it so that it could be accessed quickly and easily over any network. All that is needed to interface with

it is the IP address of the server and the query for information. This interface was achieved programmatically using Python connection libraries. An example URL string to query Solr is:

http://52.14.160.147:8983/solr/SpotifyDB/select?q=artists:"Taylor Swift"

3.2 Spotify API

We used the Spotify API for numerous reasons. The most obvious one is that since we are trying to improve the Spotify search algorithm, we might as well use the Spotify API to pull songs so that we can make a fair comparison between our implementation and theirs. The Spotify API provided an easy way to crawl through multiple playlists and pull all of the songs from a single source. In addition, the Spotify API allowed us to pull information about the songs like tempo and energy that was already calculated by Spotify. What specific song data we were able to get from Spotify is in the Song Data section below. In addition, we were able to use the Spotify recommendation feature to pull similar songs. This allowed us to custom tailor our searches for more specific queries.

4 Song Data

We stored very detailed song information in order to create more accurate playlists. Every song that went into our database was stored with its name, artist, album, danceability, liveness, energy, instrumentalness, speechiness, valence, tempo, key, and other information shown below. The song data was uploaded into the Solr Instance in the following format:


```
{
    "album": "Waking Up",
    "artists": "MJ Cole",
    "duration_ms": 224633,
    "episode": false,
    "external_urls": {
        "spotify": "https://open.spotify.com/track/5boGUtIrJjSd2uAueKCHKz"
    },
    "id": "5boGUtIrJjSd2uAueKCHKz",
    "name": "Waking Up",
    "popularity": 36,
    "type": "track",
    "playlists": [
        "37i9dQZF1DX2czWA9hqErK"
    ],
    "key": 11,
    "mode": 0,
    "acousticness": 0.95,
```

```
        "danceability": 0.482,
        "energy": 0.211,
        "instrumentalness": 0.0000478,
        "liveness": 0.0929,
        "speechiness": 0.0365,
        "valence": 0.273,
        "tempo": 127.826
    }
```


5 Music Search++ CLI

We wrote a Python CLI program to facilitate the interface between the user, Solr and Spotify. This CLI lets the user search our Solr database by song name, by artist, or by its specific Spotify ID. In addition, the CLI allows the user to create a playlist based on a specific song. The user must input their username, the song ID, the playlist name and the playlist description. The CLI then calls the appropriate functions to get songs similar to the one we specified and create a playlist with those songs. The instructions for the CLI are shown below:

```
$ ./search.py -h
usage: search.py [-h] [-s SONG | -a ARTIST | -i ID] [-p PLAYLIST & -u USERNAME & -i ID & -d DESCRIPTION]

Utility to search for spotify by song, artist or song ID and to create
playlists based off of song ID's

optional arguments:
  -h, --help            show this help message and exit
  -s SONG, --song SONG  Search for a song by name
  -a ARTIST, --artist ARTIST
                        Search for songs from an Artist
  -i ID, --id ID        Search for song based on ID or create playlist based
                        off of song ID
  -p PLAYLIST, --playlist PLAYLIST
                        Name of the playlist to be created. MUST be used with
                        -i/--id
  -d DESCRIPTION, --description DESCRIPTION
                        Playlist Description. Must be used with -p,-i and -u
  -u USERNAME, --username USERNAME
                        Spotify Username. Must be used with -p, -i and -d
```

Figure 1: Music Search++ CLI

5.1 Song Searching

We were able to create playlists using our CLI. This allowed us to achieve fast and accurate song searching either by name of the song or by artist. It returns a result if there is a text

match in the results. This means if only a portion of a song name is typed in (maybe the user doens't remember the entire song name) they can still get a list of potential matches. The results of which are shown below:

```
$ ./search.py -s Taylor
Searching for song: Taylor

Song: Speakerbox (feat. Ohana Bam & Lafa Taylor) - F8 Remix
        Artist:  Bassnectar
        Album:  The Fate of the Furious: The Album
        URL: https://open.spotify.com/track/2S2GQxhLpQO9hnCUsW2Pv7
        ID:  2S2GQxhLpQO9hnCUsW2Pv7

Done!
```

Figure 2: Search For Songs Including the Word: "Taylor"

```
$ ./search.py -s Delicate
Searching for song: Delicate

Song: Delicate
        Artist:  Taylor Swift
        Album:  reputation
        URL: https://open.spotify.com/track/6NFyWDv5CjfwuzoCkw47Xf
        ID:  6NFyWDv5CjfwuzoCkw47Xf

Done!
```

Figure 3: Search For the Song: "Delicate"

```
$ ./search.py -a Taylor
Searching for songs by artist:  Taylor

Song: Delicate
        Artist:  Taylor Swift
        Album:  reputation
        URL: https://open.spotify.com/track/6NFyWDv5CjfwuzoCkw47Xf
        ID:  6NFyWDv5CjfwuzoCkw47Xf

Song: Back To December
        Artist:  Taylor Swift
        Album:  Speak Now (Deluxe Edition)
        URL: https://open.spotify.com/track/7lxADouiWFkwR7ZV2GKUcH
        ID:  7lxADouiWFkwR7ZV2GKUcH

Song: Counting the Days
        Artist:  Will Taylor James
        Album:  Counting the Days
        URL: https://open.spotify.com/track/1umJyGPzSaIzgh0Na4lE1E
        ID:  1umJyGPzSaIzgh0Na4lE1E

Done!
```

Figure 4: Search For Songs by Artists with "Taylor"

5.2 Playlist Creation

Using our CLI we were able to construct a playlist around a specific song. We search for song ID because it removes any ambiguity or possibility of returning different songs with the same name. We then call the Spotify API, specify the specific metrics with which we want to search for songs, and receive 20 songs back (this number can be increased or decreased). The CLI then creates a playlist using the CLI and populates it with the returned similar songs.

```
$ ./search.py -u matthew.broman -i 5ru8NxVD4U5zH7wJtt1cE5 -p "Taylor Swift
Country" -d "Playlist based off of Taylor Swift's Tear Drops on My Guitar"
Creating a playlist based off of song ID: 5ru8NxVD4U5zH7wJtt1cE5
Authenticating user:  matthew.broman
Authenticating user:  matthew.broman
Returned Playlist:  5vlIhm9JiTx6AS1PFI7NYX
Searching for song with ID: 5ru8NxVD4U5zH7wJtt1cE5

Song: Teardrops On My Guitar
        Artist:  Taylor Swift
        Album:  Fearless
        URL: https://open.spotify.com/track/5ru8NxVD4U5zH7wJtt1cE5
        ID:  5ru8NxVD4U5zH7wJtt1cE5

Done!
```

Figure 5: Creating a Playlist Based on the Song, *Teardrops On My Guitar*, by Taylor Swift

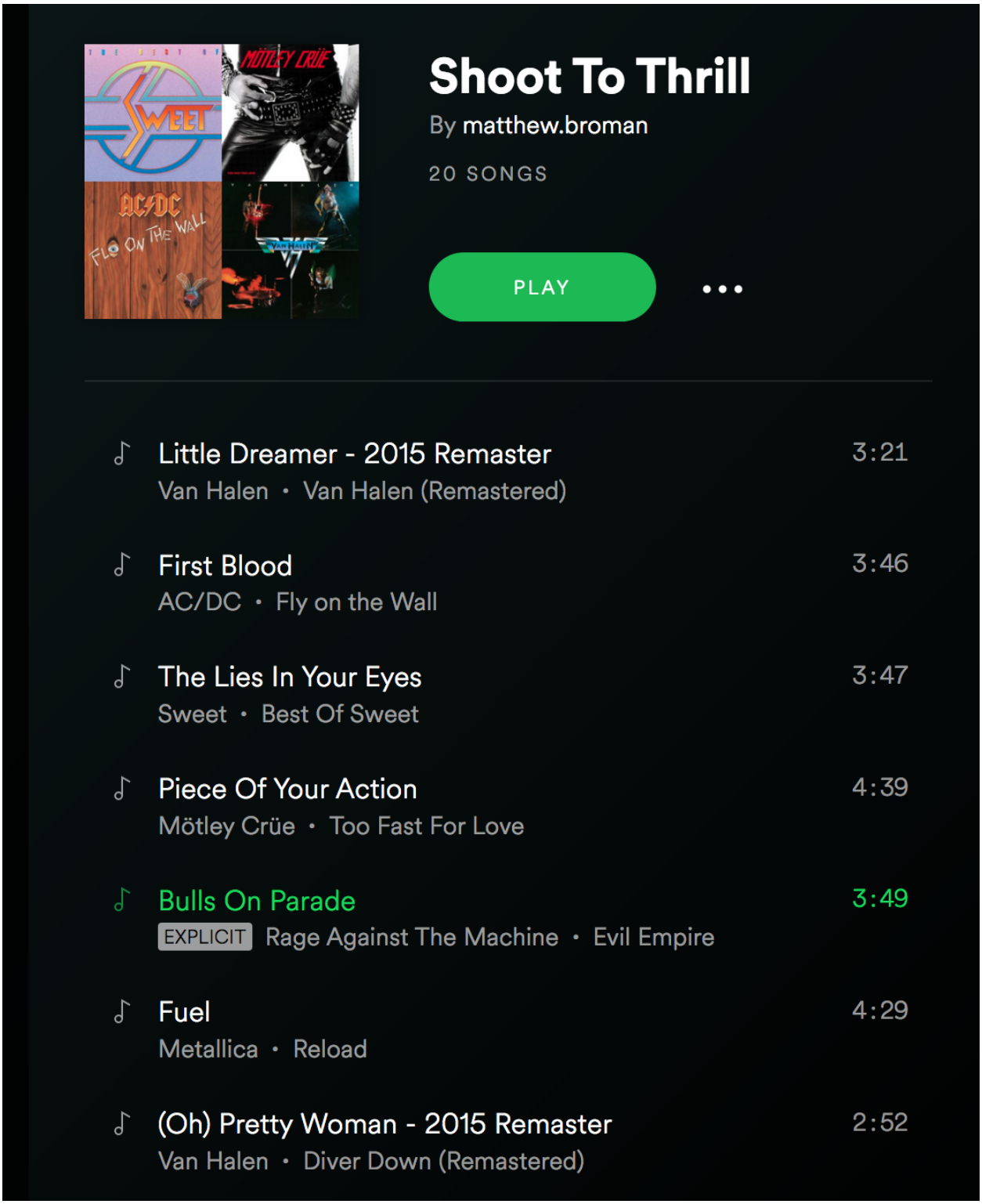

Figure 6: Playlist Based on the Song, Shoot To Thrill, by AC/DC

6 Playlist Analysis

We were able to create a playlist based off a single song. In the example above, the playlist was based off of the song, *Shoot To Thrill*, by AC/DC. The playlist created had fairly consistent song choice. We did not realize until the end, but we actually created a new feature for Spotify. Spotify does not have the option to create a playlist based off a single song. Its closest equivalent is going to the "Song Radio" which is a similar feature to ours, but does not allow the user to save those songs without further user action. In addition, Spotify usually tries to generate a playlist off of multiple songs, such as recommending songs for adding to a playlist while viewing a specific playlist. In addition, when Spotify recommends songs to be added to a playlist, it usually does so using an amalgamation of song data generated from all of the songs in the playlist or from the user's listening history. This usually dilutes the song information with information from other songs and can skew the target mood and feel of the returned songs of that search. Also, our returned songs are usually better matches than Spotify's implemenation because we specify ONLY the song metadata as target values to reach. Spotify tries to create a range of song metadata (range of values for liveness, energy, etc.) in order to broaden its search. In addition, we don't include album, artist, or popularity data in the search so this can expand the search to more obscure, yet high quality songs that fit our criteria. Due to the characteristics of the song, results are usually in the same genre but from different artists. Somtimes, the song will come from the same artist or even in the same album. Every playlist we created had songs with very similar characteristics.

7 My Contributions

Essentially, my contribution to the project was the Solr setup and querying tools. I handled the setup and configuration of the Solr Instance as well as writing the Music Search++ CLI. I set up an Amazon EC2 instance (a cloud virtual machine) and loaded Solr 7.7 onto it. I set up the VM to allow network requests and started the Solr engine. Setup was very simple as most of the features that we needed came out of the box. Solr was able to use its dynamic field generation feature so that we were able to upload our json song documents without having to specify the specific fields in the Solr schema beforehand. For the CLI, I used the default Python package, Argsparse, as the command line parser since it was easy to use and flexible enough to have the CLI fulfill the various functions it needed to have. From there, I was in charge of using the Spotify API in order to get similar songs based off a song's ID in addition to creating a playlist and populating the playlist with the similar songs.