Milestone Report - SpotRec: A Spotify Recommendation System

Shuo Han, Edmond Lu, Alvin Qu, Jungwon Shin, Wei Wang

1. Abstract

Spotify users and music lovers in general need a better user experience of receiving recommended songs. We are building a new recommendation system to provide a more customizable (controlled) and sound-based recommendation result. Our system takes a new and unique approach by implementing a combination of collaborative filtering and content-based filtering algorithms on two different datasets. We will then validate the outcome of our system by evaluating with accuracy metrics and comparing to Spotify's existing system's outputs.

2. Background

Most of the existing open-source solutions are flawed because they typically only use one type of approach. Collaborative filtering only approach causes uneven distribution of data (popular items get more data and less popular data get "drowned out") and the content-based filtering only approach tends to ignore user patterns and social trends. Spotify's current recommendation system is fully automated using historical data, which has no consideration for user input, no way to reset user data and start fresh, and no flexibility to "tweak" algorithm if users are unsatisfied with the recs. We combine both approaches to recommend newly released (still unpopular) music based on existing user patterns as well as musically-similar content. Our solution gives the users the option to input their preferences for how the song is sonically in the form of audio feature sliders on the UI.

3. Datasets

To build the recommendation system, we use two separate datasets:

- Spotify Million Playlists Dataset (MPD)
- 160K Track Dataset (Spotify API)

3.1 Datasets Overview

3.1.1 MPD

Spotify Million Playlists Dataset contains one million playlists created between 2010 and 2017 by Spotify users in the U.S. The dataset is 33GB, and contains 1,000 JSON files, each consisting of 1,000 playlists.

Each playlist contains:

- Playlist meta-data: id, name, description, # of albums, # of artists, # of tracks, # of followers, duration, *etc*.
- Track: url, name, artist, album, duration, the position in the playlist, *etc*.

Playlists	Tracks	Albums	Artists
1000000	2262292	571629	287742

3.1.2 The Spotify Dataset

Collected using the Spotify API, this dataset consists of 169909 songs with release dates between 1921 and 2020.

Each song contains:

- Song Description: Song Artist(s), Track ID, Song Name, Genre, Released Date, Released Year, and Song Popularity
- Audio Features: Acousticness, Danceability, Energy, Duration (ms), Instrumentalness, Valence, Tempo, Liveness, Loudness, Speechiness, Key Mode, Song Key, and Explicit Content.

4. Milestones/Analysis

4.1 Data Cleaning and EDA

4.1.1 EDA for MPD

Firstly, we used pandas to check for any duplicate playlists or null values in the tracks and found that there are none. Besides the tracks, we care about some metadata of the playlist: number of albums, number of artists, number of tracks, and number of followers. These fields help identify the diversity and the popularity of the playlists.

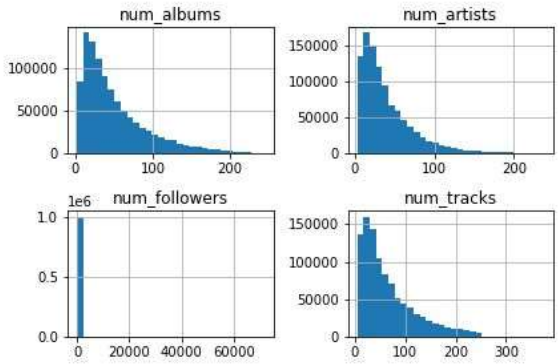

Fig 1: Distributions of Features

From Fig 1, we can see that all these fields have a high peak and a long tail except for the number of followers, suggesting that the dataset is noisy and contains outliers. The following bar plots help us in detecting the quantiles and outliers.

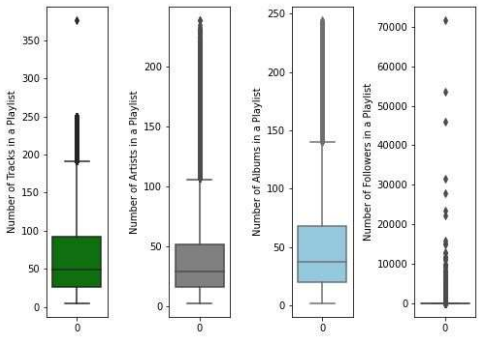

Fig 2: Barplots of Features

Although there are some outliers as shown in the figure above, the "nearest-neighbor" approach for generating recommendations using our system means that the outputs are unaffected by their existence.

Collaborative filtering often suffers from the "cold start" problem. Therefore, it is always good to know popular tracks/artists/albums when users provide little information about their playlists. Here we generated the top 10 frequent tracks/artists/albums showed up in the million playlists:

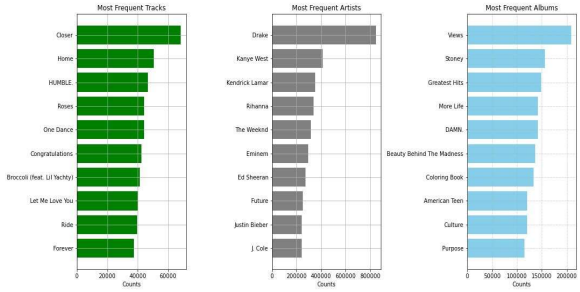

Fig 3: Frequent Items

4.1.2 EDA for Spotify Dataset

Our initial analysis on the 160k tracks and their features shows that there are no missing or null values in the dataset.

In the cases where users wish to see a global recommendation before inputting a song or playlist, we look at the "popularity" feature to recommend songs that are most popular. Here are the top five most popular songs and their popularity scores:

Song	Popularity
Blinding Lights	100
ROCKSTAR (feat. Roddy Ricch)	99
death bed (coffee for your head)	97
THE SCOTTS	96
Toosie Slide	95

We use different similarity metrics that are affected by the feature values' range. So to prepare this data, we normalize many of the features using Sklearn's MinMaxScalar.

For content-based filtering, we mostly care about the audio features and some of the song's information, such as popularity and year released. Three of these features (explicit, mode, and key) are discrete features (binary or integer), while the other 12 are continuous floats.

Histograms of these 15 normalized features are shown below in Figure 4.

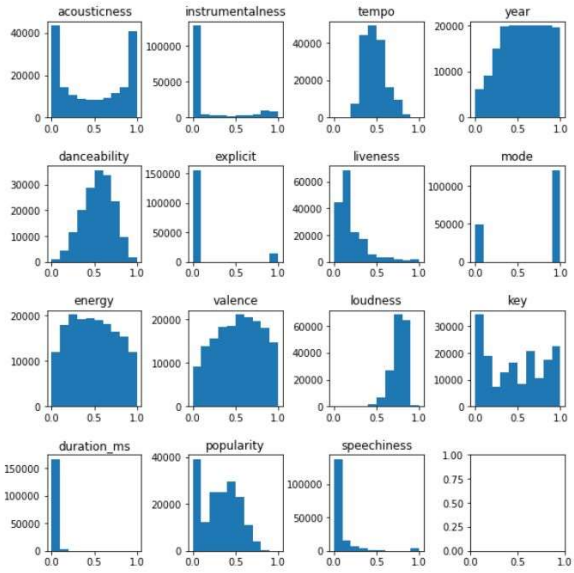

Fig 4: Histogram of Normalized Audio Features

From Fig 4, we see that several fields have skewed distributions, such as duration_ms, because there exist outliers with extreme values while the rest have a much lower average. To account for this "bias", we will explore adjusting the weights of the feature columns that contain these outliers to be lower, or even discarding the columns that have a noticeably negative effect on the output accuracy due to skewed distributions, or simply removing the outlier points from the data. We will then evaluate and analyze the resulting performance to decide which solution to employ.

4.2 Recommendation Algorithms

4.2.1 Collaborative Filtering

Firstly, we processed MPD to a utility matrix to perform user-user collaborative filtering. In this case, the user is the playlist, and the item is the track. The Jaccard similarity is calculated to measure the distance between playlists. However, due to the number of playlists and the number of unique tracks, we used MinHash and Locality Sensitive Hashing to approximate nearest neighbors of playlists. After finding the top-k nearest neighbors, we computed the real Jaccard Similarity to exclude false positives. To increase the recall rate, we would find $2k$ nearest neighbors when we need to obtain k true nearest neighbors. The following plot shows the predicted Jaccard similarity and real value after finding 10 neighbor playlists of a query input on about $\frac{1}{10}$ of the entire dataset.

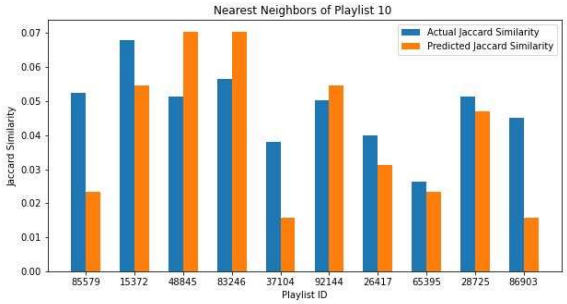

Fig 5: Jaccard Similarity of "Neighbors"

Because of the sparsity of the utility matrix, the Jaccard similarity between the two playlists is relatively low, which indicates that playlist-playlist collaborative filtering may not be a good choice. We are currently working on pivoting to a track-track method for collaborative filtering.

4.2.2 Content-based Filtering

In order to find songs that are similar to the user's input song, we are currently utilizing four different similarity metrics: Cosine Similarity, Euclidean Distance, Pearson Correlation, and Inner Product. For each similarity, we find the top-*k* similar items to the input song, using the 15 features shown above in Figure 4. For now, the system is still separated among the 4 metrics, with the metric to be used specified in the algorithm as an input field. The metrics will later be combined to output a coalesced ranked list.

5. Initial results

5.1 Collaborative Filtering

Currently, users can input a list of song names or playlist ID existing in the MP dataset, and the collaborative filtering would return a top-*k* songs output to the user based on the output of the LSH. Here, *k* is a hypermeter that the user defines.

As shown in the example below, the current outputs are somewhat inaccurate due to the sparsity of the utility matrix, but we will explore track-track as mentioned previously. Here's a sample output of the current state:

```
Playlist 10 have songs:
 ['Fix You', 'Tiny Dancer', 'Chasing Cars', 'Say Something', 'Iris', '100
Years', 'Drops of Jupiter', 'Hallelujah', 'Slow Dancing in a Burning Roo
m', 'Breathe Me', 'Rather Be (feat. Jess Glynne)', 'For the First Time',
'This Dance', 'Find A Way (feat. Emmanuel Jal)', 'Sign of the Times', 'St
op And Stare', "I'll Be", 'Calling All Angels', 'Too Much To Ask', 'This
Town', 'With Or Without You', 'She Will Be Loved - Radio Mix', "You're Be
autiful", 'Fields Of Gold', 'Paper Houses', 'Praying', 'Perfect', 'Shoo
p', 'Push It', 'Whatta Man', 'Jump', 'Bust A Move', "U Can't Touch This",
"It's Tricky", 'Insane in the Brain', 'Brass Monkey', 'Wannabe - Radio Ed
it', 'The Way You Make Me Feel - 2012 Remaster', "Don't Stop 'Til You Get
Enough - Single Version", "Wanna Be Startin' Somethin'", 'Man in the Mirr
or - 2012 Remaster', 'Beat It - Single Version', 'Barbie Girl', 'My Prero
gative', 'Macarena', 'Blue (Da Ba Dee) - Video Edit', 'What Is Love', 'Ne
ver Gonna Give You Up', 'Wake Me up Before You Go-Go', 'Take On Me', "Sta
yin' Alive", 'Dancing Queen', 'Mamma Mia', 'Fernando', 'Oh, Pretty Woma
n', "These Boots Are Made For Walkin'", 'Feeling Good', 'The Tide Is Hig
h', "Don't Stop Believin'", 'I Melt With You', "Friday I'm In Love", 'The
Safety Dance', 'You Spin Me Round (Like a Record)', 'Down Under', 'Afric
a', 'My Sharona', 'Ballroom Blitz', 'Love Shack', 'Whip It', 'Tainted Lov
e', 'Heartbreaker', 'Your Love']


We can recommend the follwing songs:
 ['pick up the phone', 'iSpy (feat. Lil Yachty)', 'Congratulations', 'Bou
nce Back', 'HUMBLE.']
```

5.2 Content-based filtering

Here is a sample of output recommendations using our content-based filtering:

```
Input Song: Bohemian Rhapsody
Input Artists: ['Queen']
Similarity Metric: Cosine Similarity

Input: exact artist match found.

Recommendations:
1. Bohemian Rhapsody - 2011 Mix
   by ['Queen']
   Similarity score: 0.9981
2. Ven Espíritu Ven
   by ['Marco Barrientos']
   Similarity score: 0.9941
3. Rooster - Live at the Majestic Theatre, Brooklyn, NY
   by ['Alice In Chains']
   Similarity score: 0.9939
4. Aléjate de Mí
   by ['Camila']
   Similarity score: 0.9903
5. Movement
   by ['Hozier']
   Similarity score: 0.9892
```

6. Next steps/Discussion

6.1 Current Issues & Plans

6.1.1 Speed for MPD

Due to the size of MPD, running LSH each time will take more than 20 minutes. It will be better to store the hashed values and hash functions beforehand, so that when there is a new query, we can utilize the exact hash functions and permutations to hash the new query, which is much faster.

6.1.2 Lack of User Personalization

Content-based filtering is not currently personalizable by the users. We plan to make customized recommendations based on the user inputs. It assumes that all audio features are both weighted and "preferred" equally. As a result, we will explore a way to place custom input weights or multipliers on certain features that the user wishes to

adjust. (Example: for a song, a user can adjust the "acousticness" and "speechiness" sliders to be higher in order to accentuate said features.)

6.1.3 Joining Two Dataset

To combine both the collaborative and content-based filtering methods, we will join the datasets by matching the tracks in both datasets using shared columns. Although the MPD does contain track URLs, which are unique to each song on Spotify, the Spotify dataset only contains unique track IDs from Spotify's API. We will attempt to map from track URL to track ID. As a backup plan, we may just use the columns for track names and artist names to match the tracks and join the two datasets.

6.1.4 Cold Start Problem

Currently, neither the collaborative nor the content-based filtering methods address the "cold start problem": when a newly released song, that is not in the dataset, is used as an input. Our current implementation simply recommends trending popular songs, tracks, albums to the new user if it cannot recognize the input song. However, we may want to use matrix factorization algorithms such as SVD to alleviate the issue. Another solution we will explore is to pull the input's features from Spotify's Web API.

6.2 Future Work

6.2.1 Algorithmic Improvements

For the collaborative filtering method, we will explore the new approach that uses similarity both between playlist-playlist as well as track-track, as mentioned previously. We will also explore the effects of SVD to alleviate the cold-start problem. As for the content-based filtering component, we plan to add more similarity metrics and optimize them efficiently. After we finish improving both methods, we will merge all the outputs in order to provide the final personalized song recommendations.

6.2.2 Evaluation

Quantitatively, we will test the performance of our outputs against those of Spotify's current implementation, using metrics such as R-Precision, Jaccard / Cosine Similarity, and Discounted Cumulative Gain. We will also look into using other metrics such as coverage and personalization to provide more insight into our recommendations.

Qualitatively, we will evaluate our system's outputs by surveying people and asking them to evaluate our recommendations. Two main metrics that we will use for this are providing a numerical score (scale of 1-10) for our output as well as calculating how many people choose our recommendations over Spotify's Song Mix Radio.

6.2.3 User Interface

If time permits, we hope to create a website with a user-facing UI. Users will be able to input songs and adjust sliders that control the weights of each audio feature.

6.3 Final Deliverable Timeline

	Milestone	Due
1.0	Algorithmic Improvements	11/24
2.0	Combining All Metrics	11/24
3.0	Testing/Evaluation	11/28
3.1	Quantitative	11/26
3.2	Qualitative	11/28
4.0	Final Submission	12/3
4.1	Write-Up	11/31
4.2	Results Visualization/ UI	12/3