Spotify Recommender
Team 20
CSCI E-109a

Erik Subatis, Ankit Bhargava, Mark McDonald

Project Statement

At its core, the project seeks to answer the question:

"How do I generate a desirable playlist for a listener?"

More specifically, the project seeks to answer the above question *given* a context consisting of one or many songs provided by the user. The result is a playlist of 20 songs that are similar to the song(s) provided by the user. The recommendations are primarily based on preferences from other users measured by other users' playlists.

Communication and Team Structure

The team is communicating via Slack.

Code is shared via GitHub: https://github.com/subatis/CS109a_finalproject_group20

The database is shared via GoogleDrive:
https://drive.google.com/drive/folders/14OBw3t3gKwPgxX3tx_ogHHpXpx6iWXRC

EDA

Data Preparation

The project data has been stored in a database for uniformity, speed and simplicity of access. In addition to the playlists provided, tables were added to include additional data for tracks and artists.

A set of API wrappers were created to streamline access to Spotify data and the Team Database. These API's simplify Spotify authentication and avoid the need to navigate arguments that are not relevant to the project.

Various records needed to be deleted or changed after reviewing their validity. Some artist URL's had changed in Spotify and some tracks were no longer available. These changes were not extensive and did not have a major impact on the usefulness of the data.

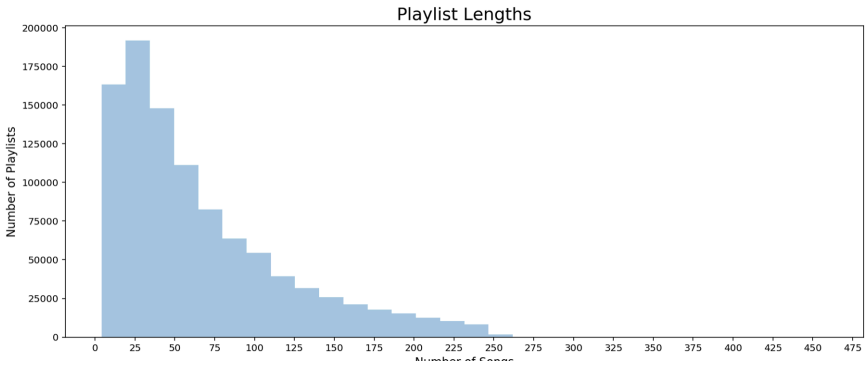
Playlists

The Playlists table is extensive with 999k playlists and 66M tracks

```
Number of unique playlists in db: 999,001
Number of tracks in playlists (not unique): 66,344,450
```

There are some outliers with very long lengths, but the average playlist is 50 songs long and the most common length is 20.

```
Mean number of songs per playlist  : 66.41
Median number of songs per playlist: 49.0
Minimum playlist length: 4
Maximum playlist length: 459
Mode playlist length: 20 songs ( 15028 playlists )
```

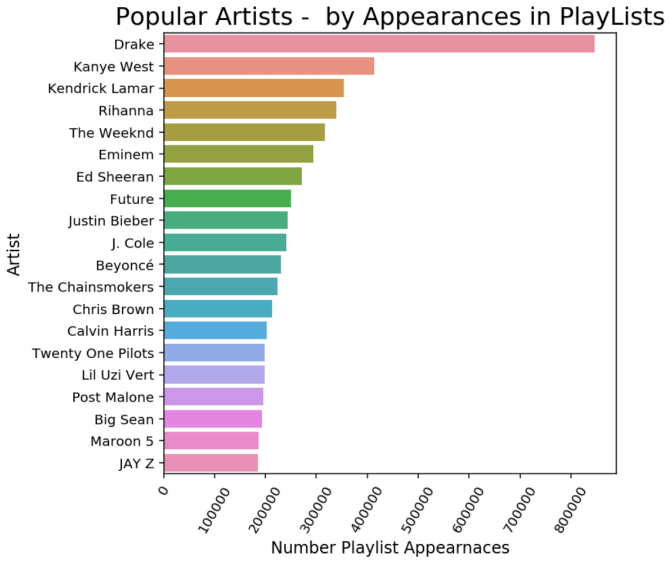


Artists

In the playlists, 296k unique artists exist:

```
Number of unique artists in db:   296,002
```

By examining the number of appearances in playlists, we are able to determine the popularity of Artists based on our dataset. Spotify also supplies a field called ‘artist popularity’; however, we found that a majority of Artists in our dataset had a popularity of 0, so we will not rely on the Spotify popularity data.



Spotify supplies genres by artist. After extracting this data, we determined that over 60% of artists had no genre assigned by Spotify, so we will not rely on this data.

Tracks

2.2M unique tracks can be found in the playlists.

```
Number of unique tracks in db:    2,261,597
```

After querying data from Spotify, various additional useful fields are available for each track. Values are assigned to a significant portion of the population making these features useful for building recommendation lists. Distributions of these features are available in the accompanying notebook.

Baseline Models

We were able to generate 2 baseline models. One is based on Word2Vec embeddings and the other is based on KNN. Both focus on using collaborative filtering to generate recommendations.

The specifics on how these are created are in the accompanying Jupyter Notebook.

Both approaches return reasonable playlists based on subjective observation.

Word2Vec Embeddings

Using the Gensim Word2Vec library, an embedding is created for the playlists where each playlists represents a sentence and each song in the playlist is a word in the sentence. The embedding process builds vectors for the vocabulary of songs which allows similar songs to be identified by their vector magnitudes.

KNN

In this approach a sparse matrix is used to generate vectors for songs. Then, KNN is used to find the songs that have the closest vector distance to a user's 'seed' song or songs.

Next Steps

This document presents our methodology for data wrangling, some exploratory analysis and 2 initial models for recommendation systems.

Further exploration, refinement and testing of models is the primary next step. Alongside that, it is important to determine how to assess "accuracy"/relevance and present recommendation results. Finally, we will implement and present our model results via Github pages for our final submission.