Moodify for Spotify®

By Nazem Droubi, Martin Chu, Kevin Xiang, Jason Chan

Introduction & Goals

In this section, write about our initial ideas, how we progressed through the ideas, and write about the two aspects of our project 1) Song Clustering and 2) Song Similarity through playlists. Describe each of these two and overview how we do them

The Spotify Data Set

Our project is based on a Spotify data set of 4000 user generated playlists (containing in total over 2 million songs). This data is stored as a JSON object: each playlist is a dictionary of its features, with the 7th key (or feature) being a list of tracks within said playlist. Each song, in turn, is its own dictionary of its features. Our project uses this JSON, alongside the Spotify API, to construct two corpuses of data:

1. A mapping from playlist titles to track titles within that playlists. This would be used later to implement a song recommendation model based on the DL word vector algorithm.

2. A mapping from song titles to a predefined set of audio features that we found relevant to the purpose of our project (tempo - valence - danceability - energy - speechiness). We find these by making API calls to the spotify server through the unique track IDs of each track. This will be later used to implement a playlist generating model using KMeans clustering.

Song Clustering By Audio Features (KMeans)

To implement a playlist generating system, we decided to cluster tracks based on their audio features' similarity. That is, tracks were considered "similar" to each other if the differences between their audio features were small. When we first used KMeans clustering, we notices that our clustres spanned one of the axes a lot more than it did for the other. We were suspicious of this behavior and through some investigation, we realized that the audio features had dramatically different scales; the feature in particular had values up to 220 whereas most other features had a maximum value of 1. To solve this, we just normalized each feature by the maximum value that we found in our dataset for that feature. Subsequently, we decided to go about our KMeans clustering in 2 different ways:

We clustered songs based on all 5 features and used a dimensionality-reduction algorithm (PCA) to reduce our plots to 3 dimensions. We used the hypertools library, alongside sk-learn, to perform KMeans and plot our clusters. We ran the program for a wide set of clusters [1, 15] and used the accumulated errors to plot an elbow graph and infer that the optimal number of cluster for our purposes is 7 clusters.

We found all 10 pairs of features our 5-feature set and clustered based on each of these 10 pairs (5 choose 2) to get 10 different clustering graphs. We then used these graphs to compare clus-

Panel of images of varying cluster plots for different combinations of audio features and list clustering scores

TSNE Graph

Results from what we learned from our graph.....
.....
.....
.....
.....
.....

Song Recs by Playlist Similarity (DL)

Here we have text about how we implemented this.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

TSNE Graph

Caption for TSNE graph

Caption for song recgraph

Song Rec Graph

Conclusion + Future Ideas