Automated Playlist Generation

Kade Keith, *Student, Stanford University - Computer Science Department,*
Demetrios Fassois, *Student, Stanford University - Computer Science Department*

Abstract—Our project generates music playlists based on a set of seed songs, using diverse features ranging from lyrical sentiment to song popularity. We approach the problem as both a graph problem and as a classification problem, and evaluate our results based on real human-curated playlists.

Index Terms—Playlist Generation, Song Recommendation, Sentiment Analysis

1 Motivation

With the growth of musical streaming services, there are now more songs than ever at music listeners fingertips. Because of this growth, the art of constructing playlists has become increasingly challenging, and discovering new music the in the expanse of choices is a daunting task. For this reason, we seek to build an automatic playlist generator, that can take a few songs as a seed set, and generate a complete playlist for the listener.

1.1 Goal

Using popular, human-curated playlists as our training data and test data, our system should construct playlists of similar quality. A novel aspect of our project is that we incorporate lyrical analysis in our model, as we believe that lyrical content plays an important role when creating playlists.

2 Method

We combine data from a number of sources in our project. The primary source is the Million Song Dataset (MSD) [2], and the corresponding lyrics dataset, which provides lyrics for roughly a quarter of those songs in a bag-of-words format. In addition to that we use Spotify [4] as the source of our playlist data, as well as using their's and Last.fm's [1] song info to augment the data from the MSD.

In total, that gives us the following group of attributes:

Feature	Source
Year	MSD, Spotify
Tempo	MSD, Spotify
Timbre	MSD
Tags	Last.fm
Danceability	MSD, Spotify
Energy	MSD, Spotify
Loudness	Spotify
Speechiness	Spotify
Acousticness	Spotify
Instrumentalness	Spotify
Liveness	Spotify
Valence	Spotify

With these features, we approach the task two ways; first as a graph problem, and second as a classification problem.

2.1 Graph Problem

The first approach is to think of songs as nodes in a graph. With this you can apply k-nearest neighbors to find most similar songs given a seed or set of seeds. You can also think of a playlist as a path through this song graph. [5]

2.2 Classification

The second is to think of deciding whether or not a song belongs on a playlist as a classification problem. Positive training examples are a subset of songs on the playlist. Negative examples are a random selection of songs not on the playlist. Then, presented with a previously unseen song, the model classifies it as either belonging on the playlist or not.

3 Evaluation Method

TODO - HOW WE SCORE RESULTS. CLASSIFICATION IS EASY - DOES SYSTEM GET IT RIGHT OR NOT

Given a subset of a playlist as a seed, evaluation is based on the comparison of our results with the actual remainder of the playlist.

4 Preliminary experiments

The bulk of our work thus far has been in data collection and processing. We have set up the complete pipeline for our model using a small subset of the MSD. We first take the MSD and filter out the songs for which we do not have lyrics data. For those that we do, we perform preliminary analysis. As a baseline we are just using Naive Bayes (with the NLTK movie review corpus as training data [3]) to score each song as either positive or negative, which then gets included in the features. After that, we enhance our data with audio features from Spotify and Last.fm.

For gathering playlists, we rely on searching for a particular term, such as "summer" or "love", and saving the top playlists for that search. We have automated this process so that all we need to input is the term. Deciding which terms to search for provides an interesting sub-problem. In general, we want to avoid "trending" or "hits" playlists, because even though those are popular, they are not cohesive. We focus instead on themes that we believe users will make cohesive playlists about, such as emotions or seasons.

4.1 Graph Problem

The simplest graph approach is k nearest neighbors, which we have implemented. We represent each song as a point in 10-dimensional space according to our normalized features (all those listed above, excluding Timbre and Tags). Then we select the next songs for that playlist based on proximity to the seed.

4.2 Classification

As a classification baseline, we rely on SciKit Learn's linear regression implementation.

5 Results

5.1 Graph Problem

TODO - HOW'D IT DO?

5.2 Classification

TODO - HOW'D IT DO?

6 Next steps

TODO - WHAT'S NEXT. FIGURE OUT HOW TO COMPARE TIMBRES (KERNEL)?. OTHER ADVANCED THINGS WE READ IN PAPERS. BETTER LYRICAL ANALYSIS.

6.1 Graph Problem

TODO - EXPLORING PLAYLIST AS PATH IN GRAPH

6.2 Classification

TODO - NOT SURE

7 Contributions

TODO - WHO DID WHAT

References

[1] Last.fm. https://www.last.fm/.
[2] Million song dataset. https://labrosa.ee.columbia.edu/millionsong/.
[3] Nltk text corpora. http://www.nltk.org/book/ch02.html.
[4] Spotify. https://www.spotify.com/.
[5] Masoud Alghoniemy and Ahmed H. Tewfik. A network flow model for playlist generation. In *In Proc IEEE Intl Conf Multimedia and Expo*, 2001.