# Spotify Mood Search

## Written Report

Dawson Abner

Jason Gilman

Elin Maredia

Elnaz Marediya

Ilhan Raja

CSCE 470

11-18-2020

## Introduction

Spotify, the popular music streaming service, offers the ability for users to search for their favorite artists, albums and songs. Although, these searches are very basic in nature, only offering search capabilities that match the user's query terms to the titles of the artists, albums and songs in Spotify's database. Our project sought to add functionality to searching Spoify's musical database, in the form of a mood based search.

## Overview

Our project, Spotify Mood Search, added functionality to the Spotify service that allows a user to narrow down their search based on their current mood or taste in music. We utilized the Spotify API, which offered endpoints to access specific musical data like beats per minute (bpm), key, energy, danceability, and positivity. By ranking the results of a Spotify search based on these data points, we were able to deliver a more comprehensive search feature that allowed users to find the exact type of music they are interested in. Currently, Spotify does not have any functionality to directly search for music based on your mood or taste. By adding this functionality to a popular service like Spotify, our project has the potential to to reach a vast audience of music listeners. Additionally, this audience would greatly benefit from the greater control of music searching.

## Dataset

During the development of our project, we created two datasets. The first was used for the development of our project and was made up of 500 songs, while the second was for the final release of our application and was made up of 5000 songs. By utilizing the Spotify API, we were able to obtain songs for our datasets that were rated with the following metrics:

| | |
|---|---|
| Acousticness | % of how acoustic a song sounds |
| Danceability | % of how suitable a song is for dancing |
| Energy | % of intensity and activeness |
| Instrumentalness | % prediction of no vocals in a song |

| | |
|---|---|
| Liveness | % presence of an audience in a song |
| Speechiness | % presence of spoken words in a song |
| Valence | % rating of positivity in a song |

Spotify Metrics That Each Song is Rated With

Beyond the metric ratings, each song has an entry for the song title, the album it belongs to, and the artist that produced it. The following are the average metrics for the two datasets used:

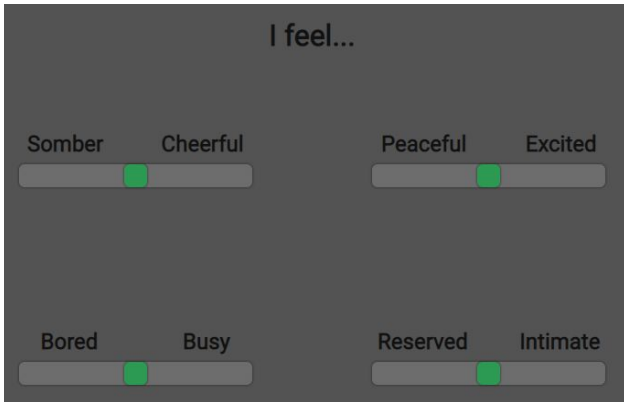
| Development Dataset | Production Dataset |
|---|---|
| # Songs = 500 | # Songs = 5000 |
| Avg Acousticness = .257022 | Avg Acousticness = .211714 |
| Avg Danceability = .617296 | Avg Danceability = .59368 |
| Avg Energy = .622326 | Avg Energy = .651787 |
| Avg Instrumentalness = . 048271 | Avg Instrumentalness = .045818 |
| Avg Liveness = .262644 | Avg Liveness = .249266 |
| Avg Speechiness = .100322 | Avg Speechiness = .10391 |
| Avg Valence = .441958 | Avg Valence = .457824 |

Average Metrics for our Datasets

## Technical Approach

Our algorithm's main function is to calculate the similarity between the user's mood and the songs, artists, and albums available on the Spotify service. Our main approach broke this idea into two necessary phases. The first phase is to translate a user's mood into the Spotify-defined metrics that our dataset is based on. The second phase is to calculate the similarity between the user's mood and the content in our database. Once the similarity is calculated for each item in our dataset, we can sort and return the best results based on the user's mood. We decided that a vector space model with an addition of centroid vectors would best suit our project.

To help the user specify their current mood, we provided percentage sliders on our UI.




Showing the Percentage Sliders on UI

We tried to come up with the terms labeled on the sliders such that it would provide the user enough information to define their mood correctly, but we also wanted to present a degree of ambiguity so that interesting results are presented. Based on the user's selections, we weighted the Spotify-defined metrics to accurately translate the user's mood to quantifiable data. Through the weighting process, we were able to create a vector defining the user's mood. This vector contained the same Spotify-defined metrics as each song in our dataset. This vector also serves as the input to our ranking function.

After the user mood vector is created, the ranking algorithm can begin. The first step of our ranking algorithm is to calculate centroid vectors for each album and artist in our dataset. These centroid vectors will be of the same format of the user mood vector to ensure the most accurate similarity can be calculated between vectors. To create the centroid vectors, all of the songs in our dataset were iterated over. If songs belonged to the same album or artist, the normalized average of each Spotify metric was recorded. It was important to normalize the centroid vectors because each album and artist could potentially have a different number of songs. Once the centroid vectors were created, the cosine similarity could be calculated between the vectors. Since our application returns the most similar songs, artists, and albums based on the users mood, the cosine similarity would need to be calculated between the user mood vector and each song, artist centroid, and album centroid. Once the similarities were calculated, we sorted the songs, albums, and artists in descending order. For visualization on the UI, we returned the top 100 of each item.

Once the ranked lists were returned to the UI, we simply visualized the rankings so that the user could scroll through their top 100 songs, albums, and artists. If the user clicks on one of the items, they are able to listen to the song, album, or top 5 songs for the artist.

## Evaluation and Results

Since our algorithm produces ranked results, we decided to use the NDCG score to accurately assess the performance of the results. An evaluation dataset was created with 1000 songs, and about 50 artists of varying genres. To compute the ideal DCG score and the ranking function's DCG score, we came up with a relevance label scheme.

| Range | Relevance score |
|---|---|
| .97 < % similarity <= 1.0 | 8 |
| .94 < % similarity <= .97 | 7 |
| .90 < % similarity <= .94 | 6 |
| .85 < % similarity <= .90 | 5 |
| .79 < % similarity <= .85 | 4 |
| .72 < % similarity <= .79 | 3 |
| .64 < % similarity <= .72 | 2 |
| .55 < % similarity <= .64 | 1 |
| 0.0 < % similarity <= .55 | 0 |

Relevance Labels

The ideal DCG was calculated with a selected amount of each relevance score, adding up to 1000 documents. After the ranking function computed the similarity scores for the evaluation dataset, we were able to assign the relevance scores to each similarity score. This allowed us to compute the true DCG score for the evaluation dataset. Finally dividing the two allowed us to compute the following NDCG scores:

| | |
|---|---|
| Song Ranking NDCG Score | .950945 |
| Album Ranking NDCG Score | .882538 |
| Artist Ranking NDCG Score | .867102 |

NDCG Score Results

It can be seen that the album and artist NDCG scores are lower than that of the song NDCG score. This is because each song has its own isolated vector, it is not a normalized centroid vector. Both album and artist vectors are normalized centroid

vectors, so when the similarity between the user mood vectors occurs, it is harder for the algorithm to find the most relevant results. When the normalized centroid vectors are created, there is a possibility of high variance in how an artist's, or album's songs sound, which would skew some of the results.

## Error Analysis

To perform the error analysis on our dataset, we took the input value for each data type and compared it to its respective output value. The following snippets show the top 5 results of two inputs:

| ranking | overall similarity | acousticness % deviation from input | danceability % deviation from input | energy % deviation from input | instrumentalness % deviation from input | liveness % deviation from input | speechiness % deviation from input | valence % deviation from input |
|---|---|---|---|---|---|---|---|---|
| input | – | 0.189 | 0.837 | 0.837 | 0.159 | 0.215 | 0.841 | 0.837 |
| 1 | 0.987965 | 0.021164021 | 0.189964158 | 0.328554361 | 1 | 0.51627907 | 0.395957194 | 0.323775388 |
| 2 | 0.987965 | 0.021164021 | 0.189964158 | 0.328554361 | 1 | 0.51627907 | 0.395957194 | 0.323775388 |
| 3 | 0.986398 | 0.501587302 | 0.223418965 | 0.23655934 | 1 | 0.172093023 | 0.405430226 | 0.327359618 |
| 4 | 0.985659 | 0.716402116 | 0.340501792 | 0.126642772 | 1 | 0.2 | 0.268727705 | 0.244922342 |
| 5 | 0.985659 | 0.716402116 | 0.340501792 | 0.126642772 | 1 | 0.2 | 0.268727705 | 0.244922342 |

Highly Specified Input Vector Error Analysis

These are the results from a highly specified input vector, meaning that the input values are taken from a realistic standpoint. The dark green values are the outputs that are the most similar to the input value, whereas the red values are the most deviant from the original input. These are fairly good results for the top 5 options for this particular input. The error percentage is very low for this case.

| ranking | overall similarity | acousticness % deviation from input | danceability % deviation from input | energy % deviation from input | instrumentalness % deviation from input | liveness % deviation from input | speechiness % deviation from input | valence % deviation from input |
|---|---|---|---|---|---|---|---|---|
| input | – | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| 1 | 0.921809 | 0.094 | 0.154 | 0.156 | 0.18 | 0.472 | 0.8186 | 0.266 |
| 2 | 0.919367 | 0.498 | 0.444 | 0.856 | 0.908 | 0.91 | 0.18 | 0.308 |
| 3 | 0.918338 | 0.652 | 0.158 | 0.544 | 0.484 | 0.096 | 0.9162 | 0.896 |
| 4 | 0.916683 | 0.12 | 0.236 | 0.386 | 0.518 | 0.798 | 0.334 | 0.122 |
| 5 | 0.910566 | 0.122 | 0.159 | 0.122 | 1 | 0.35 | 0.134 | 0.072 |

Unspecified Input Vector Error Analysis

These are the results from an unspecified input vector, meaning that these are the default set of values that are used when the input is not specified or changes. All the values are set to 0.5. This results in highly devient values. The top 5 results are actually pretty different from the input compared to the one above. The error percentage is a bit higher for this case than the one above, but it still yields relevant results.

By performing the error analysis, we can conclude that these results are quite error free and useful. As found above, this algorithm is more useful when the input is highly specified or given by a user, rather than the default values given.

## Conclusion

Spotify mood search is a simple application that allows its users to input their current mood or feelings and allows them to view and listen to music based on their input. Its purpose is to allow users to explore new music that matches their feeling rather than listening to something they would always want to hear. By using the dataset from the Spotify API, we are able to apply a VSM algorithm to return songs or music that correspond to the mood reflected in the user's input. By careful evaluation of the application and by calculating the NDCG scores and error percentages, we are able to prove the accuracy of our application. Users of this application can easily find music that accurately reflects their mood.