Exploring Content-based Recommendation for Music

Abhi Dubal
u0906015@utah.edu
University of Utah
Salt Lake City, Utah, USA

Kiranmayee Dobbali
u1265299@utah.edu
University of Utah
Salt Lake City, Utah, USA

Sergio Castanon Diaz
u1116980@utah.edu
University of Utah
Salt Lake City, Utah, USA

KEYWORDS

datasets, content-based recommendation, collaborative filtering, text processing, TF-IDF, K-Nearest Neighbor, word embeddings, and nDCG

1 INTRODUCTION

Most, if not all, recommendation systems from music streaming services utilize some sort of user profile to make relevant recommendations based off of what a user has listened to. These systems fall into the category of collaborative-filtering recommendation systems where their models try to give recommendations based off of what other users with similar listening habits listen to. For example, they may recommend songs based off of a certain genre if that genre is generally liked by other similar users, or they may recommend songs based off of a certain artist if that artist is generally liked by similar users.

A problem with the collaborative-filtering approach is that if a music service or system has few to no users, then there is no way to recommend songs based off of similar users. Another issue with collaborative-filtering is what is known as the cold start problem. For instance, if a new item is introduced into the set of all items that can be recommended, it is difficult to recommend such an item. A recommendation system that circumvents these issues is one that employs a content-based recommendation approach, which is analyzed throughout this paper.[1].

2 PROBLEM STATEMENT

Most recommendation systems for music often employ a user-based collaborative filtering approach where certain heuristics measuring user behavior and ratings are collected for making recommendations. This type of recommendation system is important for services that already have an established user base where the service already has data on user habits and user ratings. A problem with this approach is the cold start problem, where recommendation becomes difficult if a new item was introduced into the set of items that can be recommended. This brings an interesting question to explore: Are user profiles necessary for making quality recommendations? To successfully answer this question, it will be necessary to explore how features collected from music alone can contribute to music-based recommendation systems. There are plenty of features that can be gathered from music such as lyrics, genre, artist name, acousticness, energy, valence, tempo etc. In this paper, we analyze content-based recommendation approaches for recommending music by developing various models. In the subsequent sections, we explain the dataset we used for developing our recommendation systems, the models, the metrics used for evaluating our models, the results generated by the model, and detail our contributions of our overall project.

3 DATASET

Our dataset consisted of a compilation of song data from two main sources, **Data on Songs from Billboard 1999-2019** and Spotify's API. We collected the song name, artist, and genres which were included in the former dataset and using the **Spotify API**, we fetched audio features using REST principles. We list the audio features that we considered from the API below:

- Acousticness - Having a sound that is not electrically enhanced nor modified
- Danceability - Appropriate or conductive to dancing.
- Energy - Showing or involving great activity or vitality
- Instrumentalness - Performed on instruments, with no vocals
- Liveness - Performed publicly with little to no studio editing
- Speechiness - The presence of spoken words in music
- Valence - Musical positiveness (happiness, cheerful, euphoric)
- Tempo - The pace or speed of music

$\approx$ 100 of the unique songs from **Data on Songs from Billboard 1999-2019** were throwing exceptions because of unique characters that would be in the song names themselves (making it difficult to deliver speedy requests to the Spotify API), so we did not consider such songs. The .csv file consisting of our dataset is made available here: **final data**.

After cleaning the data and removing duplicates, we were left with 6873 songs. When choosing our dataset, we were not considering the fact that songs would be on the charts multiple weeks, but if we did, there would be quite a number of duplicates. Plus, some songs were not found on Spotify so that was also another factor that cut our dataset down. Collectively, we had basic song information and features from Spotify as well as lyric data for songs which we used to construct feature vectors in creating the different models we explored for our content-based recommendation systems.

4 MODELS

We considered four content-based recommendation models for the task of recommending songs. We list them below:

- TF-IDF based recommendation

[1]We make our code available at: **Song Recommender Repository**

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- K-means based recommendation
- Spacy based recommendation
- K-Nearest Neighbor (KNN) based recommendation

We chose these models primarily through personal research on recommendation systems not related to music that employ a content-based approach.

4.1 Model 1: TF-IDF

In information retrieval, TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that reflects how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. First, the relevant features like song are extracted from our dataset. In this model, we created a TF-IDF matrix from the words in lyrics using **TfidfVectorizer from the Sklearn library.**
TF : Term frequency is generated using the formula:

$$tf(w,d) = \log(1 + f(w,d))$$

IDF : Inverse Document Frequency is generated using the formula below:

$$idf(w,D) = \log\left(\frac{N}{f(w,D)}\right)$$

Finally, TF-IDF is generated using:

$$tfidf(w,d,D) = tf(w,d) * idf(w,D)$$

Now, using this TF-IDF matrix, cosine similarity is generated between each of the songs to another. The higher the value of cosine similarity, higher the similarity between the songs. Cosine similarity can be calculated using the below formula:

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a}\cdot\vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors. For each song, we returned top 5 songs that are most similar to that song.

4.2 Model 2: K-Means Clustering

K-Means Clustering is a method of vector quantization that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. In this model, our goal was to use the Spotify features like danceability, energy, acousticness, valence, tempo, instrumentalness, liveness, speechiness to recommend similar songs along with the genres of the songs. The idea is to generate clusters/new genres using these features so that we could define each of the songs and identify how these songs are related to each cluster. Since each of the songs fell under different genres, we used one-hot encoding to captures this information. After normalizing the Spotify features using **minmax scaler** and finding out the best number of clusters, which we decided to be 6, that can be formed from the dataset using the elbow method, we grouped the songs into 6 clusters using the K-means algorithm. We used cosine similarity to determine the similarities between the songs that are in the same cluster and recorded the top 5 most similar songs for every song as the recommendations for that song.

4.3 Model 3: Spacy

Spacy is a popular, advanced natural language processing library designed for processing text for many purposes such as information extraction, deep learning, or gathering insights. We used Spacy's word embedding tool to create word embeddings for a song's lyrics and genres. The purpose of using word embeddings are to numerically represent words so they can be used for computations related to deep learning, similarity analysis, etc. In addition, Spacy's word embedding tool captures the semantics of a word. In order to embed the words of the lyrics and genres, we used **Spacy's large English model** consisting of $\approx 684k$ keys $\approx 684k$ unique vectors. Finally, we represented each number in the embeddings as their own separate features and packed all the audio features from Spotify to create the final feature vector. We normalized the vectors and used cosine similarity as shown in **Section 4.1** to compute the similarities for each song compared with every other song in our dataset. We recorded the top 5 most similar songs for every song as the recommendations for that song.

4.4 Model 4: K-Nearest Neighbors

As for our fourth model, we used the K-Nearest Neighbor (KNN) algorithm which means that we would compare feature vectors of every song and return the songs with the most similar features based on a similarity measure. Again, like some of the other algorithms, the similarity measure that we used was the cosine similarity metric. This model's purpose was to give a sort of baseline model where we only considered Spotify's song features which were:

- Artist
- Title
- Genre
- Danceability
- Energy
- Key
- Loudness
- Mode
- Speechiness
- Acousticness
- Instrumentalness
- Liveness
- Valence
- Tempo
- Duration
- Time Signature

These features were then preprocessed, mainly by encoding the artist, title and genres as labels so that there were not any words in the features. Then, the features were scaled using the sklearn **StandardScaler** method. After that, we used skearn's **cosine similarity** to compute the similarity between songs. For each song, we returned the five songs with the highest similarity score, which gave us the "neighbors" or most similar songs to that specific song.

5 EVALUATION

For our relevance labels, we assigned labels to the top 100 recommendations from Spotify's recommendation system for each of the songs in our dataset and labeled each of the songs based off of their position in the results from Spotify. Our method for this labeling was as follows:

- First quarter of results: Rank 4
- Second quarter of results: Rank 3
- Third quarter of results: Rank 2
- Fourth quarter of results: Rank 1

Then, any songs not in Spotify's recommendation were given a relevance label of 0.

For evaluation, we used the nDCG equation to explore which recommendation model gave us the best result. So, we took the DCG@5 like below where we let the song relevance label for each of the top n documents be one of the r values:

$$DCG@5 = r_1 + \frac{r_2}{log_2(3)} + \frac{r_3}{log_2(4)} + \frac{r_4}{log_2(5)} + \frac{r_5}{log_2(6)}$$

And then we also computed the ideal DCG (iDCG@5) for the same five songs and then divided the DCG@5 by the iDCG@5.

$$nDCG@5 = \frac{DCG@5}{iDCG@5}$$

All of the nDCG values for every song were averaged for each of the models and that is what we present in our results.

6 RESULTS

Model Name	Average nDCG@5
TF-IDF	0.043962
K-Means	0.051963
Spacy Word Embeddings	0.123207
K-Nearest Neighbor	0.071269

Table 1: Table showing the nDCG result of each model

We see that from our results, TF-IDF obtained an nDCG@5 of .044, K-Means Clustering had an nDCG@5 of .051, Spacy had an nDCG@5 of .12, and finally K-Nearest Neighbor had an nDCG@5 of .07.

7 RESULTS ANALYSIS

7.1 Objective Analysis

From the results table above, we can see that the Spacy Word Embeddings model outperformed the other 3 models by a little over 0.05. Then, next we have the K-Nearest Neighbor model which outperformed the other two models by about 0.02. Finally, the other two models performed fairly equal. With these results, we can conclude that using words embeddings in a model outperforms the other models that we considered to give song recommendations. This might be because the word embeddings created by Spacy effectively capture semantics and semantics are an important feature for recommending songs.

7.2 Subjective Analysis

From our nDCG scores, we saw that they were on the lower end, but this can be explained by the fact that we used Spotify's recommendation system to generate labels for songs which creates a bias towards newer songs and includes songs that may not have been on Billboard. Although this was the case, by simply looking at the top 5 recommendations for every song, we judged that they were fairly relevant. For instance, selecting a Drake song such as Money in the Grave (a rap song) would lead to recommending more rap songs by artists such as Kendrick Lamar, Iggy Azaela, etc. We also saw that songs for a particular artist would also lead to more recommendations for songs by the same artist as was the case for Khalid's song, My Bad. Finally, we also saw that songs recommended by a model would also align with other models. This instance can be seen from Nav's song, Price on My Head, which would give a recommendation for Lil Skies song, Nowadays, by the Spacy and K-Nearest Neighbor models.

8 CONTRIBUTIONS

We constructed a web-based user-interface that allows a user to choose up to five songs from our dataset and receive five song recommendations for each song chosen. Users are able to see the recommendations given by all four of the recommendation systems at once. We summarize our contribution as:

- Built a content-based recommendation system deviating from standard collaborative filtering approaches.
- Considered various models for performing recommendations and conclude a best performing model.
- Evaluated each system with nDCG using Spotify's recommendation as the gold standard
- Built a user-friendly interface.

In **Figure 1**, we first show the table in the UI where a user is allowed to select up to 5 songs from 6873 songs that were considered in our dataset. The user must select a song and is restricted to 5 songs in order to show an output from the UI.

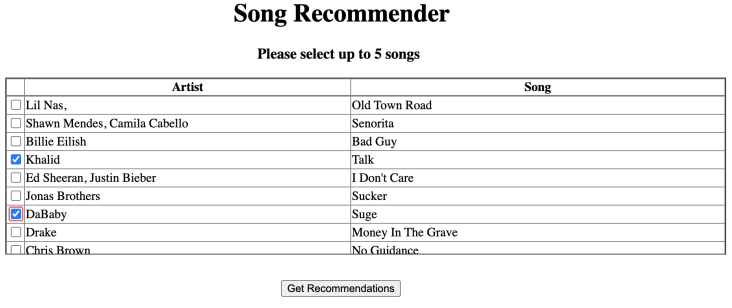

Figure 1: SongRecommender UI Table

Once the user clicks on the "Get Recommendations" button, the outputs are shown in Figures **2**, **3**, **4**, and **5**.

Each figure shows the recommendations from two songs, Khalid's Talk & DaBaby's Suge, totaling to 10 recommendations per model (5 recommendations per song). Our objective with making the UI

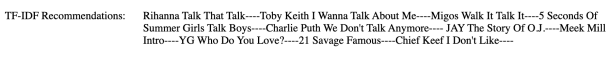

Figure 2: TF-IDF Recommendations for Khalid - Talk & DaBaby - Suge

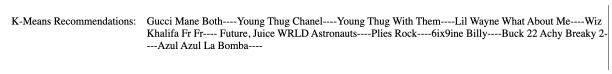

Figure 3: K-Means Recommendations for Khalid - Talk & DaBaby - Suge

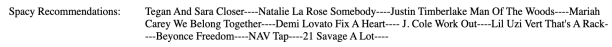

Figure 4: Spacy Recommendations for Khalid - Talk & DaBaby - Suge

Figure 5: K-Nearest Neighbors Recommendations for Khalid - Talk & DaBaby - Suge

was to allow the user to see the different songs that would be recommended by the different models we explored in this paper. The user can easily perform a subjective analysis and see if the models are recommending reasonable songs based on what songs they selected from the table. In addition, we precomputed the recommendations for every song and keep the recommendation data locally to offer fast recommendations. The UI can be accessed here: **Song Recommender UI**.

9 TAKEAWAYS

There are many points to takeaway from this project. First, Spotify tends to skew their recommendations to modern songs which could be the reason for why the nDCG results were fairly low for each model. We also believe collaborative-filtering approaches tend to be better for recommending songs to a user because such approaches consider user data. In the of personal song recommendations, having user data is vital because the quality of a recommendation is independent from user to user. In terms of our project, we believe using a larger dataset could be more promising. If we considered possibly tens of thousands of songs, the likelihood of recommending songs that align with Spotify's recommendations could be greater which would in turn increase our nDCG values. Although our nDCG values were low, we did see reasonable recommendations being made from a subjective analysis. For instance, we saw that selecting a song by country singer Blake Shelton would lead to recommendations for songs with artists Darius Rucker and Alan Jackson, who are also considerably popular in the country genre.

10 CONCLUSION

In this paper, we built a content-based recommendation system for the songs based on the features like song lyrics, genre, title, artist, and Spotify features (danceability, energy, acousticness, valence, tempo, instrumentalness, liveness, speechiness). Our dataset consists of around 6873 songs and we built 4 models using TF-IDF, K-Means Clustering, Spacy and K-Nearest Neighbors. We generated similar songs for each of the model based on Cosine-Similarities and found that Spacy model performed the best with an nDCG score of 0.123207. To compare the performance and view which songs were recommended for each of the model, we generated a UI where user could select up to 5 songs from the database and could see the recommendations on the webpage.

We were able to answer our question of whether user profiles are necessary for relevant recommendations and we found that while our recommendation system results did not quite line up with Spotify's recommendations, our model's recommendations were still valid and relevant results. So, no, user profiles are not needed to give relevant recommendations, but having a bigger repertoire of songs to choose from definitely gives Spotify an advantage and we believe that having a bigger dataset would result in recommendations more in line with the recommendations from Spotify.

As for the models, we found that the models that included Spotify's features performed better, but we also saw a jump in relevance once lyric word embeddings were added to the model. So, to answer one of our other exploration questions as to whether song lyrics were valid features to consider, yes, they are beneficial to add as a feature. This would make sense as the lyrical content of a song can influence a person's choice to listen to that song. So, by comparing song lyrics, we are able to recommend songs that have similar lyrics or cover similar topics in their lyrics.

Of course, further research with a larger dataset is needed in order to draw further conclusions with the relevance of results that are given from these models, but from what we have seen, these models are capable of producing relevant recommendations.