

Spotify recommendation system



Ruben Bromée
Viktor Carlsson
2022/01/20

For the course:
TNM108 - Machine Learning for Social Media

Examiner:
Pierangelo Dellacqua

1 Background

Spotify is a music service. As most services do, Spotify creates recommendations for its users, in this case songs that they might want to listen to. In creating these recommendations, the service uses a few tricks, one of them being *Collaborative filtering* [1]. In this case, the service uses the listening patterns of other users with similar taste in artists to find out what to recommend. The alternative to this is *Content-based filtering* [1], which uses the content itself to recommend new content for the user. We wanted to attempt developing a program that used this alternate method to create a recommendation instead, and therefore recommend songs that are audially similar to what the user already listens to.

2 Research question

Can a recommendation system be created that takes a number of input songs and recommends audially similar songs to a musical average of the input songs?

3 Data collection

The data that was used to find recommended songs was the *Spotify 1.2M+ Songs* [2] dataset. The dataset contains the following track features for each track:

1. ID
2. Name
3. Album
4. Album ID
5. Artists
6. Artists ID:s
7. Track number
8. Disc number
9. Explicit
10. Danceability
11. Energy
12. Key
13. Loudness
14. Mode
15. Speechiness
16. Acousticness
17. Instrumentalness
18. Liveness
19. Valence
20. Tempo
21. Duration ms
22. Time signature
23. Year
24. Release date

The dataset was constructed from data gathered using the Spotify API [3]. A subset of these are a part of a song's audio features, and this will be used in our method.

4 Data processing

To be able to easily compare the tracks in the dataset with Spotifys own audio features for a track, features 10 - 20 from the previous list were selected since those are the ones retrieved when getting a tracks audio features using Spotifys API. This was done by reading the track feature data into a *pandas* [4] dataframe and then isolating the relevant audio features.

The data was scaled and standardized using *StandardScaler* [5]. *Principal component analysis (PCA)* [6] was applied on the audio features to reduce the number attributes used in calculations. The number of audio features were reduced to $\min(\text{number of input songs}, \text{number of audio features})$. The five audio features weighted the highest by *PCA* were:

1. Tempo
2. Speechiness
3. Liveness
4. Key
5. Mode

5 Modeling

5.1 Method

When the dataset was processed the audio features of the given input songs were retrieved using *Spotipy* [7]. These audio features are the same as the ones retrieved from the dataset. They were also reduced to the same number of principal components as the audio features from the dataset and scaled in the same way.

Three different distance measures were used to compare the audio features of the input songs with those from the dataset. Cosine similarity, euclidean distance and manhattan distance. Cosine similarity is the cosine value of the angle between two vectors in feature space. Since the vectors of audio features most often are in a high dimension the cosine similarity is calculated using a dot product between two feature vectors \mathbf{x}_1 and \mathbf{x}_2 as seen in 1. The calculation for euclidean distance and manhattan for two feature vectors can be seen in 2 and 3 respectively.

$$\cos(\theta) = \frac{\mathbf{x}_1 \bullet \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|} \quad (1)$$

$$d_e(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| \quad (2)$$

$$d_m(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_1 \quad (3)$$

When the distance measure between each input track and each track in the dataset was calculated it was stored in an $n \times m$ array where n was the number of input tracks and m was the number of tracks in the dataset. The column sum of this array created a vector of distance measures between all input tracks and the tracks in the dataset. These were then sorted according to the distance measure to get the most similar tracks. For cosine similarity the tracks from the dataset with the highest similarity was wanted and for the euclidean and manhattan distance the tracks with the lowest distance were the most similar.

After a number of songs were recommended after their audio features, they were sorted according to their genre similarity to the input tracks. This sorting was done using *Text frequency, inverse document frequency (TF-IDF)* [8] where each track was considered a document and the genres were considered the contents of that document. The genres for each recommended song and the genres for the input tracks were retrieved using *Spotipy*. The name of the genres for each recommended track and the input tracks were vectorized using *Tfidfvectorizer* [9]. The recommended tracks were sorted by their summed TF-IDF score of their genre when compared to the input tracks.

Similarity with Spotifys recommendations were also measured where the track ID of $n \times l$ of the top recommended songs were compared with $n \times l$ tracks that were retrieved using Spotifys own recommendation system on the input tracks. n is the number of input tracks and l is the number of recommendations gathered for each input track from Spotifys recommendation system.

5.2 Result

The first batch of input songs that were used can be seen in figure 1. The top five recommended songs using each distance measure can be seen in figures 2, 3 and 4.



Figure 1: First batch of input songs.

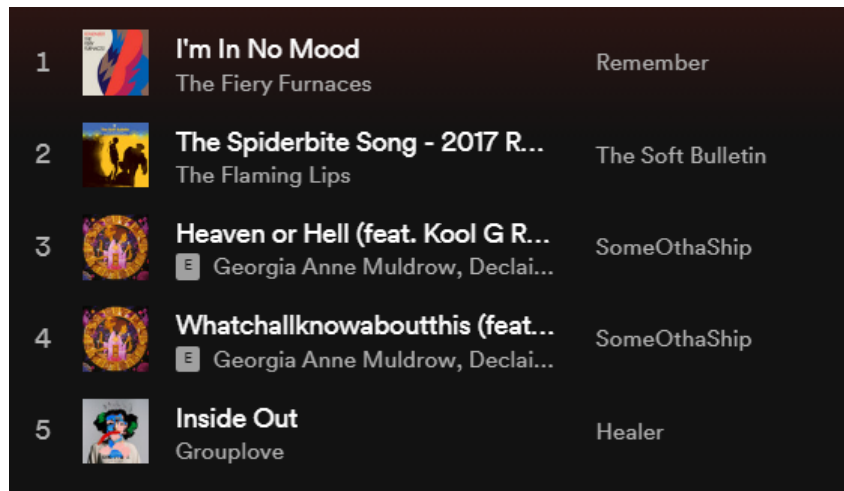


Figure 2: Recommended songs for first batch of input songs. Cosine similarity.

1		The Spiderbite Song - 2017 Remaster The Flaming Lips	The Soft Bulletin
2		Tokyo Julien Baker	Tokyo
3		Castle Down a Dirt Road The Magnetic Fields	Quickies
4		Two E The Antlers	Hospice
5		Falling in Loves too Mean Hether	Hether Who?

Figure 3: Recommended songs for first batch of input songs. Euclidean distance.






1		Close My Eyes Will Butler	Generations
2		Estrangers No Vacation	Phasing
3		Powerful Man Alex G	Rocket
4		A Gypsy Life Helvetia	Headless Machine Of The Heart
5		3 Boys Helvetia	This Devastating Map

Figure 4: Recommended songs for first batch of input songs. Manhattan distance.

The second batch of input songs that were used can be seen in figure 5. The top five recommended songs using each distance measure can be seen in figures 2, 3 and 4.






1		Maniac Carpenter Brut, Yann Ligner	Maniac
2		Graveyard Shift Dance With the Dead	Near Dark
3		The Top - Extended KEN BLAST	Ken Blast And Friends - REMASTERED
4		Division Ruine Carpenter Brut	TRILOGY
5		Nuclear Lethargy Lane 8	Automatic

Figure 5: Second batch of input songs.






1		Come On Baby Todd Terry	Todd Terry Presents Ready for a New Day
2		I've Got You Martha Wash	Something Good
3		Purple Like the Summer Rain MØ	Forever Neverland
4		Lightspan Soundwave The Shamen	En-Tact
5		Tension Dagny	Strangers / Lovers

Figure 6: Recommended songs for second batch of input songs. Cosine similarity.

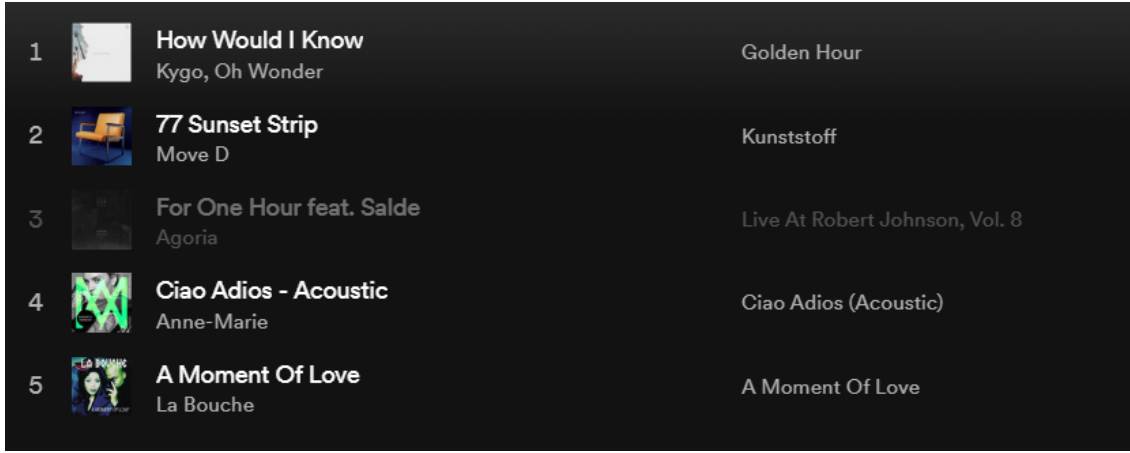


Figure 7: Recommended songs for second batch of input songs. Euclidean distance.

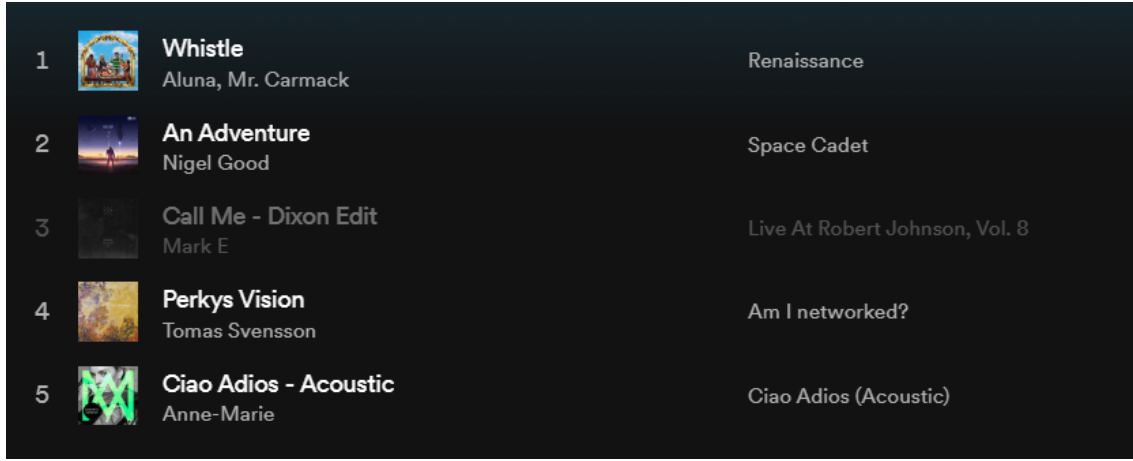


Figure 8: Recommended songs for second batch of input songs. Manhattan distance.

The three different distance measures performed differently. The summed *TF-IDF* scores for the top five songs recommended to each batch of input tracks can be seen in

	Summed TF-IDF for batch 1	Summed TF-IDF for batch 2
Cosine similarity	10.5019208929355403	2.30540618460681187
Euclidean distance	9.2673089374101056	2.46097307357086925
Manhattan distance	10.0756287425600434	2.44448840377475956

Table 1: Summed TF-IDF values for the top five recommended tracks depending on batch of input songs and distance measure.

0% of the recommended tracks were the same as Spotifys recommended tracks.

6 Evaluation

The system that was created was very slow in its execution. This is most likely due to the large number of API calls that are being made when retrieving the genres for the recommended songs. This performance is decreased further by the fact that there is a greater chance of getting a better result if more songs are recommended before they are sorted using *TF-IDF* according to genre.

The musical similarity of the recommended songs differ. It was of our opinion that for the first batch of input tracks the euclidean distance and manhattan distance gave the most consistently musically similar songs to the input tracks.

If the input tracks have wildly different audio features this will affect the systems performance since outliers will affect distance measures such as euclidean and manhattan distances.

None of the tested recommended tracks was the same as Spotifys recommended tracks. The idea of this system was to recommend songs purely based on their musical characteristics and not on popularity so it makes sense that it does not recommend the same songs as Spotifys recommendation system. Other things that could have caused this is the limitations of the dataset used. The dataset is quite limited compared to the total number of songs on Spotify.

The summed TF-IDF scores in table 1 indicate that the cosine similarity distance measure performed best when it came to recommending songs that had relevant genres to the first batch of input songs. This disagrees with our opinion that euclidean distance and manhattan distance recommended the most musically similar songs. The TF-IDF scores for the second batch of input songs are in general lower and here the euclidean and manhattan distance measure produces slightly more genre-relevant results.

Many tracks have no genres listed when getting the genre using the Spotifys API. This makes the sorting using *TF-IDF* difficult since many tracks will have no correlation with the input tracks according to genre.

To answer the research question, a system that recommends audially similar songs to the input songs were produced since songs where the audio features were similar to the input songs were recommended, making the system use content based filtering. Songs that are recommended solely in this aspect do not take popularity in to account. Which makes the system more likely to recommend songs that are not so popular. The songs recommended are the songs most similar to the musical average of the input songs since it recommends songs based on the sum of the different distance measures for all input songs.

References

- [1] Recommending music on Spotify with deep learning
<https://benanne.github.io/2014/08/05/spotify-cnns.html>
Retrieved 2022-01-22.
- [2] Spotify 1.2M+ Songs
<https://www.kaggle.com/rodolfofigueroa/spotify-12m-songs>
Retrieved 2022-01-18.
- [3] Spotify API <https://developer.spotify.com/documentation/web-api/> Retrieved 2022-01-20.
- [4] Pandas
<https://pandas.pydata.org/>
Retrieved 2022-01-18.
- [5] StandardScaler
<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
Retrieved 2022-01-18.
- [6] Principal component analysis
<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
Retrieved 2022-01-18.
- [7] Spotipy
<https://spotipy.readthedocs.io/en/2.19.0/>
Retrieved 2022-01-18.
- [8] TF-IDF
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
Retrieved 2022-01-22.
- [9] Tfidfvectorizer
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
Retrieved 2022-01-18.