UNIVERSITY OF VICTORIA
FACULTY OF SCIENCE, COMPUTER SCIENCE

Multimedia Systems December 11, 2023

Spotify's Recommender System

by

Marja Wahl

Abstract:

Spotify's personalized recommender system is one of the main reasons of its huge success. The music streaming giant uses a hybrid approach, consisting of collaborative filtering, content-based filtering, and natural language processing approaches. In my project, I applied collaborative and content-based filtering on a test set. The collaborative approach was not able to output meaningful recommendations, while the content-based one achieved an R-Precision of approximately 13%. This performance was limited by not having the same ressources available as Spotify. Nevertheless, my experiment demonstrates the algorithms behind Spotify's recommender system.

Keywords: recommender system, music streaming, Spotify

1 Introduction

Music streaming platforms have established themselves as a great way to enjoy music. Especially Spotify, as it is the world's largest streaming platform [3]. One of the reasons why Spotify is so successful is its recommendation system [1]. It suggests songs and can create whole playlists. This makes it possible for the subscribers to listen to music they enjoy, without having to look through the masses of music out there themselves. However, machine learning models are often a black box to many users; those who might be interested in how Spotify is able to successfully recommend personalized music. Yet, the complexity of Spotify's recommender system might reduce the understandability of the process for listeners and artists.

Spotify's algorithm uses matrix manipulation (MM) for collaborative filtering, convolutional neural networks (CNNs) for content based filtering, and natural language processing (NLP) for identifying cultural relevance of a new song [6, 2]. I implemented the MM and CNN approach, to illustrate and validate the design choices of the Spotify recommender system and create a comprehensive explanation for interested music enjoyers.

The approaches are evaluated on a subset of the Million Playlist Dataset on the task of automatic playlist continuation. This means, given a set of playlist features, like name or number of including different albums, generate a list of soundtracks that can continue the playlist to the users liking.

2 Analysis of Spotify's Recommender System

2.1 Collaborative Filtering

This technique uses information about other users in order to create a recommendation. For example, if two users listen to a lot of the same songs, Spotify will recommend songs that one of them listens to to the other, if the other user has not listened to that song yet. This is similar to Netflixs' "You may also like" or Amazons' "Customers have also bought" recommendations.

More specifically, Spotify uses one big matrix defined as rating matrix R with the size, $||U|| \times ||T||$, depicted in fig. 1. U is the number of users and T the number of tracks. Each entry resembles the amount of times a user has listened to a song. The problem is, the matrix alone can not give information about songs the user has never listened to before. This is why Spotify uses Weighted Matrix Factorization to approximate matrix R with two other matrices.

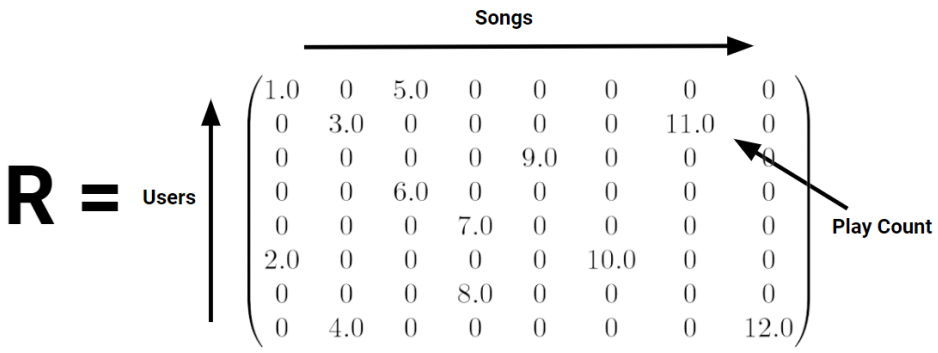

Figure 1: Matrix R [4]

Matrix factorization is an effective mathematical tool that is able to discover the latent features underlying the relation between playlists and songs. Latent features describe regularities that are hidden in the data; for example, when a user is a fan of a certain genre or artist. Factorizing the matrix means finding two matrices that multiplied result

in the original matrix. This mechanism finds underlying latent features and uses them to fill in the blanks of the matrix. That creates a ranking for songs that a user has not listened to yet. [7]

The two matrices that are searched for are matrix P, of size $||U|| \times ||K||$, and C, of size $||T|| \times ||K||$, such that their product approximates R. K is the number of assumed latent features. The preference matrix P is similar to matrix R, but every entry is 1 if users have listened to the song, and 0 if they have not. The confidence matrix C describes how confident the model is about a certain preference, factoring in the listening count of the user. A higher listening count makes the model more confident that the user likes the song; thereby influencing the output of the recommender system of Spotify. [4]

To find those matrices, they are first initialized randomly, and then iteratively improved according to a criterion. This process is called gradient descent. The criterion is the difference of the product to the original matrix. The error is calculated with using eq. 1, also called the mean squared error (MSE). [7]

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \tag{1}$$

Then, the gradient at each value is calculated, which is essentially the differentiation of the MSE function for every point in the matrix. Finally, each point in the matrices is updated by adding the gradient. This process is repeated for a certain amount of iterations or until the error has reached a certain threshold.

To sum it up, Spotify basically uses your listening data and those of other users to find similarities and recommend songs. The problem with this approach is that no new songs, songs no user has ever listened to, will be recommended. This is known as

the cold start problem, and the reason why Spotify uses two additional techniques for recommendation.

2.2 Content-based Filtering

To overcome the cold start problem, Spotify also analyzes the song itself. Audio analysis can be performed by analyzing a spectrogram of the audio, depicted in fig. 2. This is like a picture of a song, it visually encodes the frequencies of the audio signal over time. The x-axis resembles the time dimension, while the y-axis represents the frequency of the signal in kilohertz (kHz), which determines the pitch. The intensity of a pixel resembles loudness in dezibel (dB). [5]

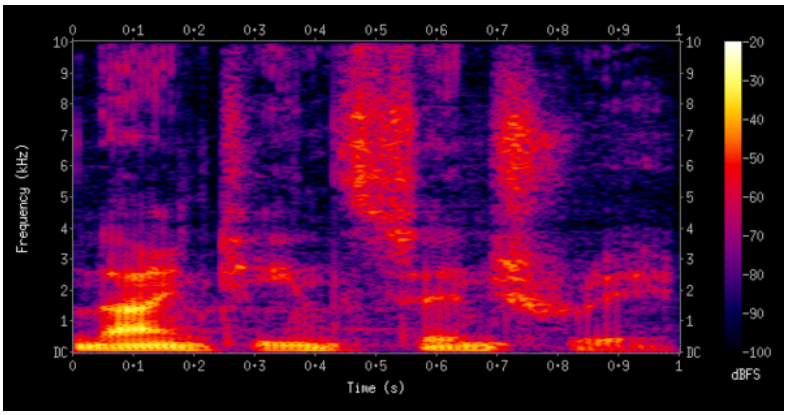

Figure 2: A spectrogram (source)

The first step when computing spectrograms is applying a Fourier transform (FT) to the soundwave, it maps the time domain of an audio signal to the frequency domain. The FT detangles possibly mixed sounds to the bare signal frequencies by utilizing Fourier's Theorem. In order to detect change in an audio signal over time, the fast FT (FFT) is applied on several specified time windows of the input, resembling the short time FT (STFT). The STFT outputs the spectrogram, the visual representation of the audio clip. This is obtained by stacking the results of the FFTs one after the other, one for every window. [5]

Spectrograms are converted to logmel spectrograms by transforming the frequencies of the spectrogram to the mel scale. This scale is non-linear, incorporating the fact that humans can recognize changes of frequency better at lower, than at higher frequencies. Finally, a logarithmic operation is applied in order to receive logmel spectrograms, which can be used as input for two-dimensional convolution. [5]

The spectrogram is fed into a CNN, which is a neural network architecture that is adapted to work with image data. It is mostly known for applications like face recognition or object detection. Since a spectrogram is essentially an image of the audio, we can use the CNN architecture to analyze it and and predict song features like loudness or beats per minute. More information about the different features that Spotify generates can be found in the documentation of Spotify's Web API. Since this analysis does not depend on any user data, it can be performed on completely new songs.

2.3 Natural Language Processing

Spotify also uses NLP to find popular new songs and artists. They scan online articles, blogs, and reviews, and perform NLP tasks like semantic analysis on them. By doing so, they generate "cultural vectors" for artists and songs. Those are then used to find similarities between songs a user listens to and freshly released music, similar to the collaborative filtering approach. [4]

3 Experiment

I used Python and Python Notebooks for implementing the project. The implementation can be found on my project website. I focused on collaborative and content-based filtering, and evaluated those methods on a test dataset. Both models are run locally.

3.1 Data

The used data is a subset of the Million Playlist Dataset by Spotify. This dataset originally contains 1 million playlists consisting of over 2 million unique tracks, while the subset I used consists of 869 playlists and 30,046 tracks. The raw data subset can be found here.

For the experiments, I used the processed dataset, which can be found here. Processed means that features were added to each track, those can be accessed via Spotify's Web API. In Python, the library spotipy can be used for this. In addition to meta information features, like track name, duration, artist, album name, the processed dataset includes content-specific features, like danceability, energy, acousticness, or liveness.

The data was then split into train and test dataset according to the playlists. Randomly, 90% of the playlists were added to the train, and 10% were added to the test dataset. The train dataset was used to build the models, while the test dataset was used to evaluate them.

3.2 Collaborative Filtering

Collaborative Filtering means that interests of other users are taking into account for making a playlist prediction. Originally, Spotify uses a matrix storing information about each user and their track counts. Since this data is not publicly available, I used a matrix P created from the train data, where each row is a playlist, instead of a user, and each column is a song, to try to mimic the original data used by Spotify. The idea was that each playlist could be seen as a user, because each user might have created one of the playlists. The final dimensions of the matrix were $P = 779 \times 27,902$, as the train data consists of 779 playlists and $27,902$ unique songs.

After constructing this matrix P, matrix factorization is applied, like described in sec. 2.1. In my project, I let it run for 100 iterations and set hyperparameter $K = 2$.

Given a new playlist, it is encoded like a row of the matrix P. For each song that the playlist contains, the corresponding index is set to 0. Then, this vector is compared to every other playlist row in the matrix P using cosine similarity. After finding the most similar playlist vector, the entries corresponding to songs in this vector are ordered descending by their difference to 1, as 1 symbolizes an entry being in the playlist. The songs with the lowest difference and not in the input playlist are output as recommendations.

3.3 Content-based Filtering

For the implementation of the content-based filtering I followed a notebook available here. The first step was to prepare the data. Therefore, one hot encoding and normalization was applied. To one hot encode means that category data, like the genre, is transformed to binary data. This is realized by adding attributes like *genre = hip hop*. They are then set to 1 if they are true, and 0 otherwise. Normalization means scaling values to the range of 0 to 1, in order to remove bias when comparing the vectors.

After preprocessing, each track held 2,094 attributes. The next step was to turn the playlist, that recommendations are wanted for, into a vector of its songs, by summing the corresponding song vectors. The process is depicted in fig. 3. This summarized playlist vector is than compared to each vector in the dataset using cosine similarity. The songs with the highest similarity are output as recommendations.

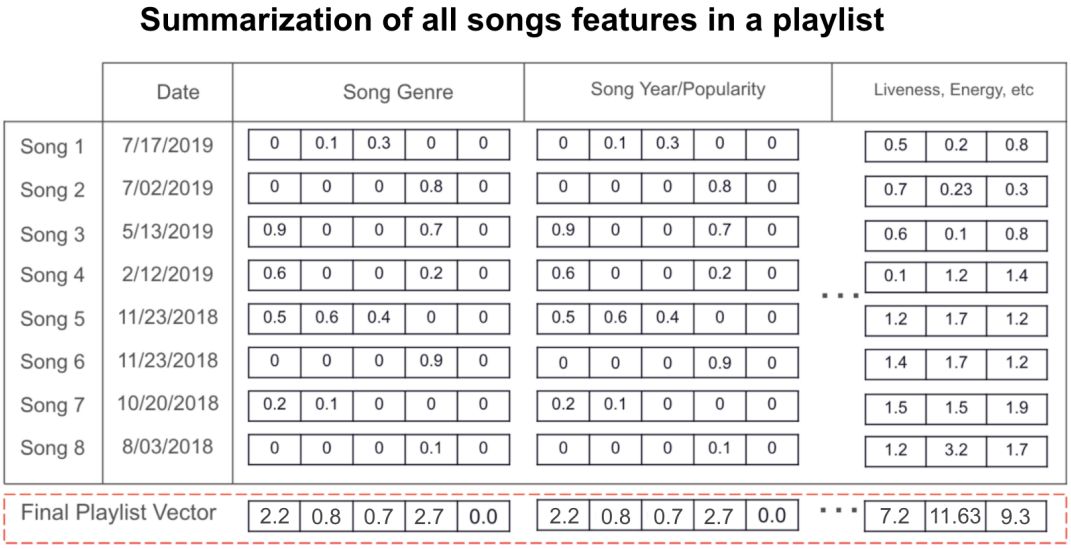

Figure 3: Creation of a playlist vector for content-based filtering (source)

3.4 Results

The test set consists of 90 playlists with 5,448 tracks. For the evaluation 20% of the songs in one playlist are removed, but remembered for computing the R-Precision. This measure describes the number of relevant retrieved tracks, divided by the number of relevant tracks. This is the percentage of recommended songs, which additionally were part of the original playlist, but removed for testing. The procedure is depicted in fig. 4.

The collaborative filtering approach achieved an R-Precision score of 0.00049, while the content-based filtering resulted in a score of 0.13254.

3.5 Discussion

When implementing collaborative filtering, the original data that Spotify uses is not available. Therefore, the information of the play counts is missing, as well as the differentiation of playlists created by one and the same user, in the data that is used in the project. This is the reason why the R-Precision of the collaborative filtering model

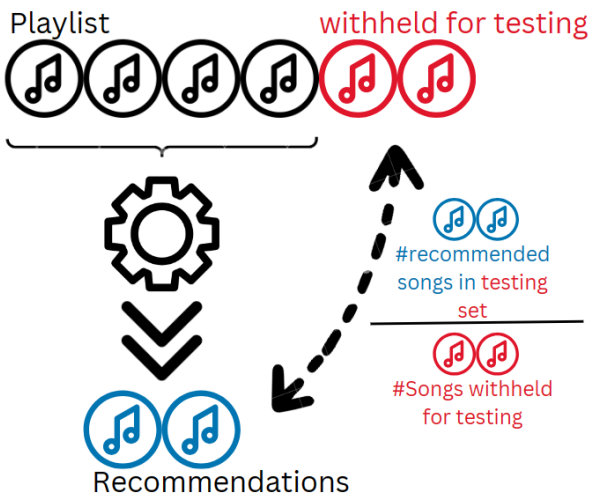

Figure 4: Evaluation procedure

is very low.

Another limitation of the collaborative filtering approach was that the training data and test data did not contain the same set of songs. Since the matrix constructed during training has one column for every song in the training dataset, it was not possible to represent new songs in the test data. This led to those songs being ignored for the recommendation, which possibly decreased its accuracy even further.

4 Conclusions

The project showed what kind of data Spotify uses to provide their state-of-the art music recommendation system. Without having access to this knowledge, recommendations are not as good, as can be seen from my experiment. As I have explained the different mechanisms behind the recommender system, it would be also interesting to investigate how their results are combined in future work. Additionally, other music recommender features of Spotify, like their weekly playlist generation or their new AI DJ, would be worth investigating as well. Another interesting idea would be to compare the recommender systems of other leading streaming services, like Ap-

ple Music or Youtube Music, to the one of Spotify, according to their recommendation performance.

References

[1] B. Baran, G. D. Junior, A. Danylenko, O. S. Folorunso, G. Forsum, M. Lefarov, L. Maystre, and Y. Zhao. Accelerating creator audience building through centralized exploration. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 70–73, 2023.

[2] S. Dieleman. Recommending music on spotify with deep learning. https://sander.ai/2014/08/05/spotify-cnns.html, 2014.

[3] M. Iqbal. Spotify revenue and usage statistics (2023). https://www.businessofapps.com/data/spotify-statistics/#:~:text=Spotify%20is%20the%20world's%20biggest%20music%20streaming%20platform%20by%20number%20of%20subscribers, 2023.

[4] A. Mohan. How does spotify's recommendation system work? https://www.univ.ai/blog/how-does-spotifys-recommendation-system-work#:~:text=Recommendations%20for%20each%20user%20are,algorithm%20on%20every%20song%20vector., 2023.

[5] L. Roberts. Understanding the mel spectrogram. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53, 2022.

[6] A. Saravanou, F. Tomasi, R. Mehrotra, and M. Lalmas. Multi-task learning of graph-based inductive representations of music content. In *ISMIR*, pages 602–609, 2021.

[7] A. A. Yueng. Matrix factorization: A simple tutorial and implemen-

tation in python. https://albertauyeung.github.io/2017/04/23/python-matrix-factorization.html/, 2017.