Milestone 3

Boesinger Léopaul, Egli Marc, Khalfi Yassine
CS-421
EPFL

May 4, 2021

1 Our approach

Our goal is to create a music playlist that fits the taste of a group, given previous listening habits of the group. In order to achieve this, we want to apply both machine learning techniques (such as SVD recommender systems), and heuristics (using the genre, and musical features such as danceability, energy and tempo).

Our hybrid approach will allow us to have a collaborative recommendation system that will rely on the tastes of the community to generate recommendations from the historical data of the different users. We will also use a content-based system to suggest music similar to what users listen to most often.

1.1 Datasets

In our study, we model the group, as a sample of users from the "Last.fm Dataset - 1K users", which tracks the full listening history of 1'000 users.From this dataset, we only retrieve music that has a certain number of listens, which allows us to reduce the computations that are very long, and also to avoid recommending unpopular music. The dataset will also be used to train a recommender system, in order to compute a rating for each user, for the songs.

Then, we also use Spotify's API, in order to compute information about the tracks listened in the Last.fm dataset, in order to get information about the tracks themselves. Some of these are : the Danceability, Valence, Energy or Tempo.

1.2 Experimental plan

1.2.1 Computing (user, track) rankings

Given our dataset, we want to compute for each user and each track, a rating, that would model how much a user would like the track. However, since we have no rating in the first place, we have to try to compute our own rating given a user's track history.
Our current choice is to compare the number of times the track is listened by the user, to the user's average number of listens per track, meaning that if a song is played much more than the user's average, it has to be liked.
This enforces a significant assumption : the user listens more to music that he likes, independently of the time, place, or mood. It is a huge simplification, but is our best chance at having significant results.

1.2.2 Discovering new tracks

In order to include domain-knowledge about tracks and to not recommend only musics that were already listened, we use Spotify's API to obtain musical features about tracks. This enables us to compute a distance matrix across tracks, and to find the closest tracks to the ones enjoyed by each user, that we can then add to the playlist. However, we have to keep in mind that we will not have any scoring for these (user, tracks) pairs.

This is the content based part of our algorithm that will allow us to propose to users music that is similar to what they are listening to, we assume that users will like it because it is similar to their past listening.

1.2.3 Pleasing a crowd

Satisfying a group of users is never easy as everybody has different taste of music. Thus our goal is to satisfy the most users. However we cannot simply consider the taste of the majority which would leave a lot of users with an uncommon tasted dissatisfied. Therefore we will try to balance the user satisfaction by introducing a fairness metric that will ensure a minimum amount of satisfaction. [1]
We will introduce a voting system that will vote for an artist if he was listen before or clustered among something user listened before. This voting system will allow us to detect artists widely appreciated by all users and therefore add some of their music to the playlist.

2 Evaluation protocol

To evaluate the performance of our protocol, we first tried to see what was possible with our resources and dataset. Unfortunately, given the resources we have, it will be difficult to experiment to see if the playlists made really correspond to the playlist that people would make or listen to at a party. The data we have is only the play history per user and it is according to this data that we will evaluate the performance of our models, so our evaluation will be done at the individual user level.

For the SVD, we can compute metrics such as RMSE to see if our model is a good approximation of our data. Rank-aware metrics should be preferred, but in our case, as we do not have information about what is useful or relevant to a user, it is not possible to use these metrics unless we use composite features which may add bias to our model. Once the playlist is created, we will use metrics to see if the music in the playlist appeals to our users, the aim of the playlist is to be balanced so that the tastes of all users are captured, to do this we think about using fairness metric.

3 Results

3.1 Rankings of (user,track) pairs

For the SVD of the users, we have ran a GridSearch with 3-fold cross validation over the size of the SVD embedding (n_factors parameters), the learning rate, and the number of epochs. The resulting numbers are the MAE, given that ratings are from 1-10.

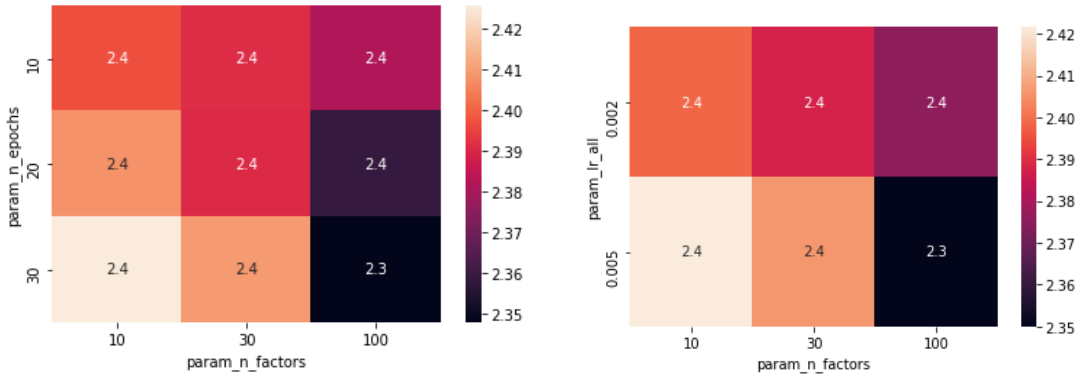

Figure 1: Heatmaps obtained from gridsearch

We are not sure why the MAE is so high (2.3 for the best model), out of a scale of 10. However, in another experiment, we tried to predict for some sampled users, songs that they would like, and got results that seemed to make sense :

	track-name	artist-name
112722	Billie Jean	Michael Jackson
213700	Here Comes The Sun	The Beatles
213748	Something	The Beatles
213857	Little Wing	Jimi Hendrix
213872	Wanna Be Startin' Somethin'	Michael Jackson

Figure 2: Most listened tracks from user

	track-name	artist-name
74309	Blue Orchid	The White Stripes
213700	Here Comes The Sun	The Beatles
213749	Oh! Darling	The Beatles
213836	I Want You (She'S So Heavy)	The Beatles
214038	She Came In Through The Bathroom Window	The Beatles

Figure 3: Predicted best tracks from SVD

3.2 Track Discovery

We were able to compute a distance matrix across tracks, so we are able to add those new recommendations directly to a playlist. We have also plotted a 2D (normally interactive) embedding of tracks :

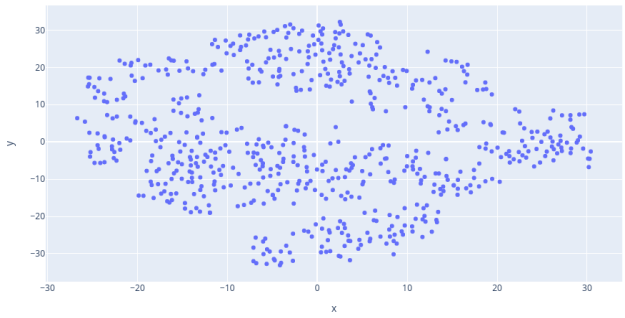

Figure 4: 2D Embedding of Tracks

References

[1] L. Boratto C. Trattner A. Said and A. Felfernig. "Evaluating Group Recommender Systems". In: *Group Recommender Systems*. URL: `https://www.christophtrattner.info/pubs/group_recsys_trattner.pdf`.