Spotify Music Recommender System
Machine learning, CS, MIU [1]

Basma Ashour Saloni Vora Shahane Eksuzyan

August 2, 2021

[1]Instructor: Anthony Sander

Contents

Abstract

This is a short acount of a Music Recommendation System made by Basma Ashour, Saloni Vora and Shahane Eksuzyan. Here we report steps we took to build three recommendation systems - Demographic(a.k.a trending now), Content Based (a.k.a because you listened) and Collaborative (a.k.a because you liked). We also report the data we used, initial steps taken to reduce the amount of it(from billions of data entries to something that we could work with the limited resources of our computers). In exploratory data analysis part, we show which features of the tracks make them more popular and plot some graphs to show our findings.

1 Introduction

The music recommendation system was built using the data of Spotify[1] and Yahoo! Music User Ratings of Musical Artists, version 1.0[2]. The Spotify datasets included 7 csv files: artists.csv, data-by-artist-o.csv, data-by-genres-o.csv, data-by-year-o.csv, data-o.csv, dict-artists.json and tracks.csv. In our project we used 3 files only. The first artists.csv contains 1.104.349 data entries. Each row contains one artist name, artist id, followers, genres and popularity. The tracks.csv contains 586.672 entries, each consisting of 19 columns: id, name, popularity, duration-ms (duration of song in milliseconds), explicit, artists, id-artist, release-date, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time-signiture. The data-by-genres-o.csv file contains genres, and all the features of tracks, but correlated to genre. The access to yahoo dataset was granted to Basma Ashour, as part of the Yahoo! Research Alliance Webscope program, to be used for approved non-commercial research purposes. Yahoo! Music data represents a sample of (anonymized) Yahoo! users' ratings of musical artists, gathered over a thirty-day period sometime prior to March 2004. The dataset consists of two files: 1. ydata-ymusic-user-artist-ratings-v1-0.txt 2. Ydata-ymusic-artist-names-v1-0.txt "ydata-ymusic-user-artist-ratings-v1-0.txt" contains user ratings of music artists. It contains 11,557,943 ratings of 98,211 artists by 1,948,882 anonymous users. The format of each line of the file is: anonymous-user-id (TAB) artist-id (TAB) rating. The ratings are integers ranging from 0 to 100, except 255 (a special case that means "never play again"). "ydata-ymusic-artist-names-v1-0.txt" contains the artist-id and name of each musical artist.

All of the data was used to give 3 types of recommendations to the user, 1. Demographic (the most popular songs right now), 2. Content-based recommendation (because the user listened to a song) and 3. Collaborative recommendation (based on user preferences). The data for the 3 types of recommendation was cached or serialized (in the 3rd case) so no new calculations are performed, while the user is using a User Interface. The User Interface shows icons of songs recommended and allows you to play it if clicked. We used docker to ship the system, we generated the dockers files to be compatible with Flask.

2 Exploratory Data Analysis

To analyse the data, we first combined 2 files from yahoo into one csv and reduced the data entries, taking only 1 percent of it. Then we changed the artists file, to have entries only about those artists which were present in the user ratings file. We created a heatmap of tracks' features, to see which of them are correlated (figure 1). We can see that danceability, loudness, energy, valence and popularity has the strongest correlations between them. Other features like instrumentalness, acusticness or duration were not strongly correlated to other features. We can conclude that the more energetic, loud and danceable the track is, the more popular it will be.

We decided to further explore what mixture of above-mentioned strongly correlated are in the 10 most popular genres (figure 2).

[1]Spotify dataset 1921-2020, 160k+ tracks. (n.d.). Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

[2]Webscope. (n.d.). Webscope — Yahoo Labs. https://webscope.sandbox.yahoo.com/catalog.php?datatype=randdid=1

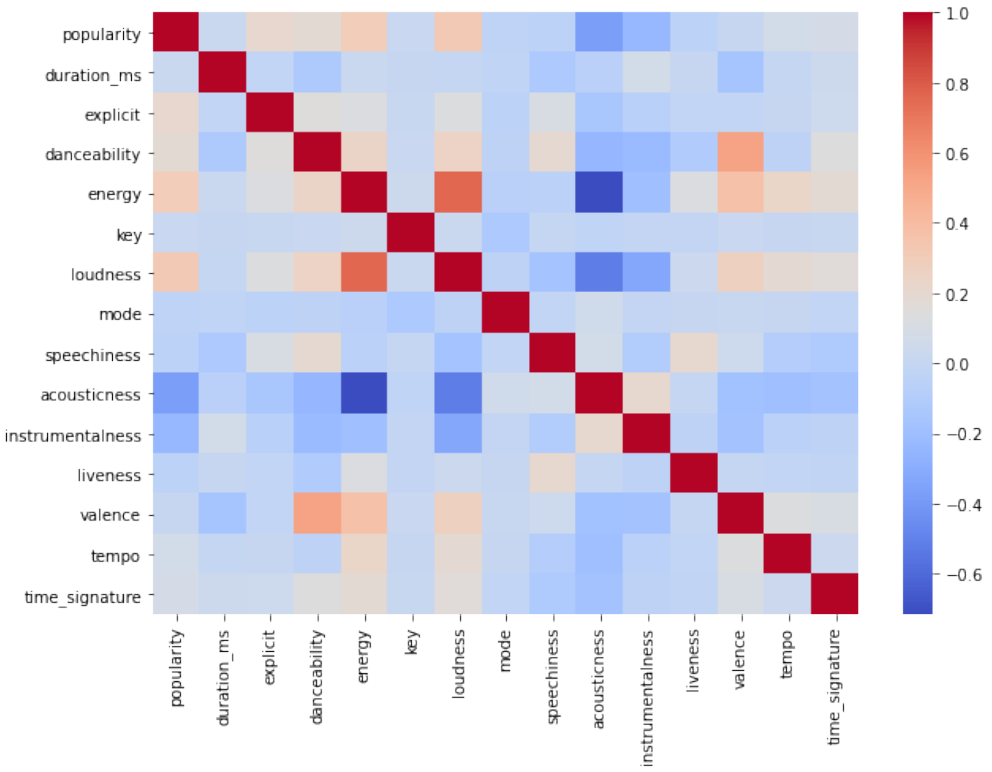


Figure 1: The heatmap of features of tracks dataset.

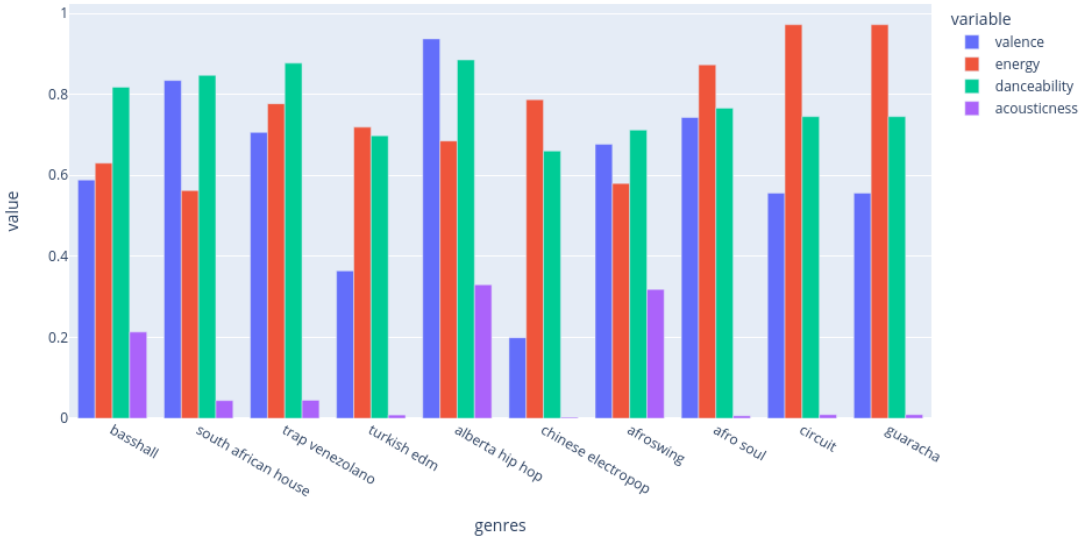


Figure 2: Presence of valence, energy, danceability and acousticness features in 10 most popular genres.

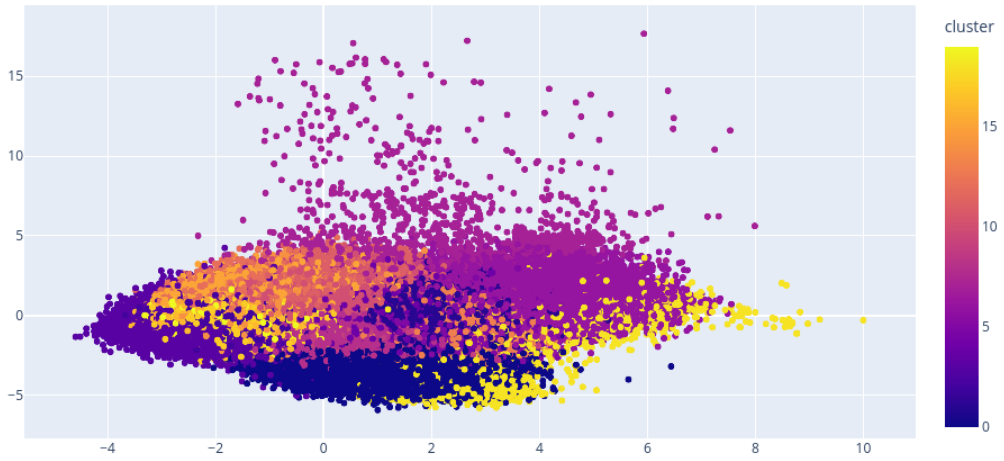
3 Models

3.1 Demographic or Popularity-Based Recommendation

In the demographic model we used only the spotify datasets. We used two files: the artists.csv file and the tracks.csv file. First, we sorted the artists file based on the most popular artists, and then we took the first 9 artists, which are obviously the most popular. Next we sorted tracks file based on the tracks popularity. We search for the most popular tracks of the 9 most popular artists received from previous file. The final output is 9 tracks which belong to the most popular artists.

3.2 Content-Based Recommendation

Content based music recommendation system was built using suggested[3] algorithm. First K-Means unsupervised algorithm was run on the tracks dataset to divide them into 20 different clusters. After PCA the data looked as follows:



To make a prediction for a new track all the numeric features of it (danceability, popularity, duration . . .) are vectorized and the mean matrix of all the vectors is computed. After that the cosine similarity, that is to say dot product, is calculated between the vector and mean vectors of each cluster. 10 closest data points to this vector in the dataset are found based on the smallest value of cosine similarity. These 10 points are recommended to the user.

3.3 Collaborative Recommendation

3.3.1 A matrix factorization SVD

For Collaborative recommendation SVD algorithm was used, which is built-in surprise python library. The data was split to train set and test set 80 and 20 percent respectively. User, item and rating were fed into

[3]Mavuduru, A. (2021, February 10). How to build an amazing music recommendation system. Medium. https://towardsdatascience.com/how-to-build-an-amazing-music-recommendation-system-4cce2719a572

the SVD algorithm, where rating varied from 0 to 100. An RMSE score of 62.1644 was received during training. Then for 130 users predictions were made of all artists present in the filtered artists dataset. The results were sorted by predicted rating, and top 10 were taken. For each of 10 artists the most popular track was taken from the track.csv file. These tracks are cached in csv and are going to be recommended to the given user.

3.3.2 Neural Collaborative Filtering

Neural Collaborative Filtering[4] is an implicit feedback model to build a recommendation system.

We have built this model to perform recommendations based on user-id and artist-id, based on teh user taste, we are trying to identify if the user will like songs from a particular artist or not.

To build this model, we have used libraries like pytorch and tqdm, and to serialize this model we have used the pickle library.

We have followed some steps to keep the memory usage and CPU usage minimal so that we can handle the task by using the limited resources we have. Hence we sampled some random records(approx. 1 percent of the complete dataset) to perform the operations.

Post the completion of records selection, we proceeded to split the train and test data so as to prepare an efficient model and evaluate the hit ratio of the same.

To avoid data leakage caused by random selection of records, we have created the rankings based on the rating given by the user to a particular artist. (Ratings lie between 0-100)

As mentioned the rating are between 0-100 which makes it an explicit type of feedback and hence we first converted this to implicit type feedback which uses 1 to represent the user interacted with the artist, by this we are predicting if the user will interact with this artist(song by this artist) or not and not trying to predict what rating he will give the song.

Now all the samples used by us belong to class 1, because the user did interact with the artist and hence we have to create some negative class records randomly.

After this step we are trying to update the dataset so that it can be consumed by the pytorch library. Now we have tried to do the User embeddings which is the main part of NCF.

By using the ratings dataset, we can identify similar users and artists, creating user and item embeddings learned from existing ratings.

The embedded user and item vectors are concatenated before passing through a series of fully connected layers, which maps the concatenated embeddings into a prediction vector as output. Finally, we apply a Sigmoid function to obtain the most probable class. Hence to achieve this, we defined teh NCF model. We have trained teh NCF model with 4 epochs using the CPU and used this reload-dataloaders-every-epoch=True parameter which randomly creates a set of negative samples for each epoch and makes sure that our model is not biased.

To evaluate the recommendation system we have used the tqdm library and calculated the hit ratio for our code. And for the current samples we have attained the hit ratio of 0.92 percent.

References - https://www.kaggle.com/jamesloy/deep-learning-based-recommender-systems,
https://medium.com/@victorkohler/collaborative-filtering-using-deep-neural-networks-in-tensorflow-96e5d41a39a1,
https://www.kaggle.com/prmohanty/python-how-to-save-and-load-ml-models

4 Restful APIs

We used Flask Framework to build our three restful apis. For the demographic api we have a cached file with the 9 most popular tracks and after that we call the Spotify Api to fetch these tracks data and then show

[4] YouTube. https://www.youtube.com/watch?v=O4lk9Lw7lS0

the results. For the Content based api we had cached the 9 most popular tracks and after that we used them to fetch with every one of them the most similar tracks after that we called the Spotify Api to fetch these tracks data and then show the results. For the Collaborative api we used the users dataset to fetch based on the users high rating tracks similar to those, after that we called the Spotify Api to fetch these tracks data and then show the results.

5 User Interface

We used the Flask template to show the results for the 3 apis.

6 Docker

We used docker to ship the system, we generated the dockers files to be compatible with Flask.