Espotifai

Automatic Playlist Recommender

Lucas Emanuel Resck Domingues and Lucas Machado Moschen

School of Applied Mathematics
Getulio Vargas Foundation

August 7, 2020

GitHub Repository

1 Project statement

Question: Which song should we recommend based on a playlist and user or context information?

We can define playlist as a sequence of tracks (audio recordings). In this project we aim to study the playlist generation problem, that is, given a pool of tracks, a background knowledge about the user, and some metadata of songs and playlists, the goal is to create a sequence of tracks that satisfies some target as best as possible. The notebooks, scripts and work can be found in our repository.

2 The Datasets

In order to get user information, we generated a list of usernames in a networked way. We visited the Last.fm webpage of several artists and considered three random users in the top listenings from three different coutries: Brazil, USA and United Kingdom. Using only these usernames, through the command `generate_lastfm_users.py`, we get additional Last.fm usernames using the `user.getFriends` method from the API. With some loops, we can get the network (or part of it). It's possible to have some bias, unknown yet.

Both databases were built using the Application Programming Interface (API) from the website, in special Spotify API and Last.fm API. Both are great players in the music business and that was the reason for us to use.

2.1 Spotify Database

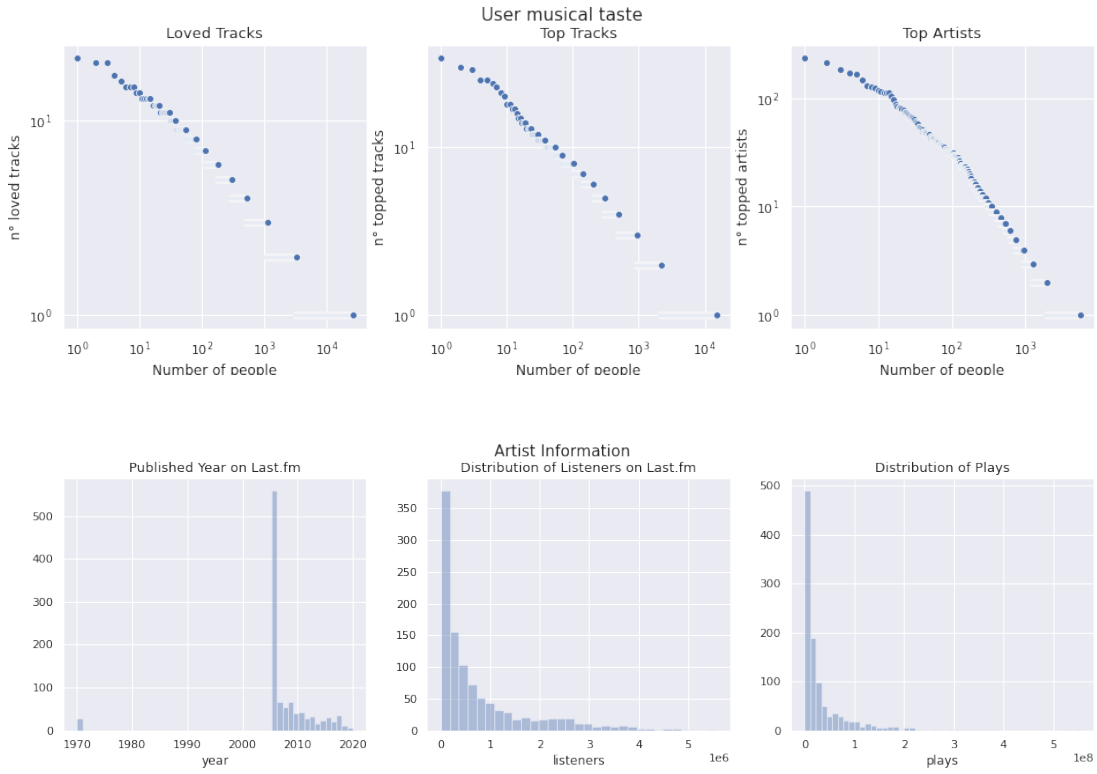
2.2 Last.fm Database

We used pyLast package for Python to help with the connection. It organizes all as an object with several functions. However, in order to retrieve all the information, it takes a long time to build all datasets: **Users info**, **Tracks info**, **Artists info** and **Tags info**, because we retrive a lot of information in different links from the same API. That's why in the first analysis we only considered a subset. We got the users randomly from the whole dataset of users. We saved the information in a `json` format. The variable types were fixed to help the analysis (convertion `string->integer`). The dates were converted to a standart type and after to datetime. Actually, as we used an API, the data wasn't bad. We created ids for tracks, tags and artists and save it in a `.csv` file. This structure allow us to save storage.

The considered datasets were:

1. **Users**: information about if is subscriber, when he/she has registered, playcounts, country, top artists and tracks and loved tracks. (no user had gender, age and number of playlists information and we deregarded.)
    - Users: 1000; users with no information: 1.
    - We are dealing with personal information generated by the user (like loved tracks) and by user's preferences (like top tracks)

2. **Tracks**: information about reaching, playcounts, duration, listeners, similar tracks and top tags.
    - tracks: 9902; tracks with no information: 311.
3. **Artists**: information about number of listeners, plays, when he/she was published, top tracks, top tags and similar ones.
    - artists: 1055; artists with no information: 0.
4. **Tags**: information about registration, taggings, reached people, top tracks and top artists.
    - tags: 1843; tags with no information: 4



3 Gained Insights

TO DO
Ideas:

1. Pareto's distribution in the data;
2. getSimilar is a good function from last.fm
3. Last.fm has information about general aspects of the user, but it's not good to create models fow playlists.

4 Baseline Model

TO DO

1. Qual o modelo mais simples que eu conseguiria pensar? Usando last seria esse: Toma uma playlist, pega a interseção dos artistas preferidos pelo usuario (top artists) com os artistas getSimilar dos tracks da playlist e seleciona as top tracks desse artista. Isso explora bem o last, o que acha Lucas? Tem outra ideia?