An open source platform to assist the creation of group playlists through artificial intelligence algorithms

Flaviano Dias Fontes[1]*, **Giordano Ribeiro Eulalio Cabral**[1] , **Geber Lisboa Ramalho**[1]

[1]Mustic – Centro de Informática / Universidade Federal de Pernambuco (UFPE)
Av. Jornalista Anibal Fernandes, s/n – 50.740-560 Recife, PE

Abstract. *Recommendation systems are a constantly expanding study area, with applications in various fields such as e-commerce, films, music to promote the user's suggestions. When we talk about music, we have more than 20 years of studies trying to solve the problem of a good generation of playlists that maximizes the satisfaction of a larger number of listeners. For automated automatic playlist generation methods focusing on a user group, we have the collaborative filter as a more assertive method to get the user's not likely, to improve the performance of group recommendation algorithms we store the preferences of users Especially I did not like it by placing the availability of using this data as an algorithm input parameter. The platform described in This paper is intended to facilitate testing between these recommendation systems, standardizing data entry, and facilitating requests. The use of GraphQL as a framework associated with Apollo as a library, greatly facilitates the integration of these APIs, as the separation of data sources makes it possible to associate Spotify data with Deezer or Apple Music data, these data are stored in the database of the connection, so that in future requests it will no longer be necessary to consult the Spotify API, thus facilitating the consumption of data from the artificial intelligence algorithms, as well as a possible sharing of songs between services, since all services have an ISRC code to identify the songs.*

1. Introduction

The music market has suffered several changes over the years and after a recent and long decline in revenues, the industry has finally managed to rejoice in revenue through streaming services, becoming the largest source of revenue, taking Spotify as principal service [1].

This way of consuming is based on the availability of songs in the order of millions of copies, for streaming services Some user-related variables are taken into account in the recommendation systems to define their musical taste, as their level of experience, their location, and their personality traits, however for user consumption has only released the songs he likes, a set of songs that the platform understands that he likes or just the playlists he created, but is not available what the user does not he likes [2]. It is easily found in the Application Programming Interfaces (API) of the Streaming Services, [3] the methods for querying playlists but with the limitation of not having the consultation of the songs that the user does not like.

*Supported by CNPq.

2. Music Recommender Systems Evolution

Recommendation systems are a set of software and techniques that suggest to the user items to be used for it. Suggestions refer to decision-making processes, such as what news reads, which product buy or music listen [4].

These systems solve the problem of information overload and help users choose from the various options in their daily lives. Recommendations systems capture previous users' preferences and generate an available list of items to meet the user [5].

Recommendation of groups for groups is a subject that has been studied, an example is the Smart Radio in 1996 that was a web application in the client-server format on which it allowed users to automatically filter and collect the songs, with User dislike capture, another important point to stand out is the automatic extraction of features making it possible to improve content-based recommendation. [6]

In 1998, MusicFX enhanced the capture of information to recommend musical genres through collaborative filters, on a scale described by I love (+2), I like (+1), I do not care (0), I do not like (-1) and I detest (-2), the algorithm adjusts the recommendations based on the evaluation of the genres of users present in the environment, described in the article as a gymnastics. [7] This approach to capturing directly does not like the user is a facilitator but through the main consultations, the APIs are not available to third parties what creates our need to capture this information about the user's dislike [8] [9] [10].

Flytrap in 2002 was a recommendation system that reproduces songs that users like, through music that a user plays on their computer it loads the data to the FLYTRAP database. Unlike MusicFX, the information sent are recovered through the playback of the track and these meta-data are, for example, the genre and the artist stored this information linked to the user who played. [11]

The JMusicGrouPrecommender merges recommendations from individual users, adds personal evaluations of users, and finally generates group preferences to generate recommendations. In this work, a fusion technique, a technique for generating group preferences, four aggregation techniques, and a hybrid technique including mixing and aggregation are proposed. Each technique is evaluated based on the satisfaction of each user who is within the group. Each song contains some attributes such as artist, album, language, and gender [12].

This article it was describing are proposals, a fusion technique, a technique for generating group preferences, four aggregation techniques, and a hybrid technique including mixing and aggregation. Each technique is evaluated based on the personal satisfaction of all team members. Each song contains some attributes, such as artist, album, language, and gender. These attributes are used to infer indefinite membership preferences. Among all the algorithms proposed in this article, it is reported that the hybrid technique has high rates of individual satisfaction among the users of a group. Compared to Automatic Consensus Selection, this template offers users the option to have at first a set of items [13] [5].

The problem when data is spaced is addressed in group recommendation systems in different ways. It is important to note that traditional group recommendation techniques do not produce better recommendations when the user-item relationship is sparse, and your preferences are unknown. [5]

The possibility of calculating the similarity of the item-item and computed similarities are used to improve existing memory-based recommendations techniques. The degree of similarity is a decimal attribute that is between 0.0 and 1.0, which is added in this template to all pars of item-item. Initially, the support technique of the Support Vector Machine (SVM) employed in this model is compared to other regression models, and has been discovered that SVM surpasses other models. The proposed model surpasses the latent factor model, which is general considered the best method to use among memory-based models. [14]

3. Streaming Available Information

Using as a matrix of characteristics of the Flytrap and JMusicRecommender features we have as a reference for information required to create music recommendation systems, the artist, the album, and the language. If the approach is a collaborative filter the preferences of users of a particular item must be stored in a database. [11] [12] For collection of this information the REST access points of Spotify, Apple Music, and Deezer as an Apollo data source, a GraphQL implementation that provides a data chart layer that connects applications to the cloud. [15] [8] [3]

3.1. Spotify

Queries in this API occurs through REST requisitions by following the URI standard. [16] The consumption of the Spotify API has the need for a free developer account and you can see user playlists list more played songs among other methods. By consulting certain music and recovering a lot of information such as name, platform popularity, artist, duration in MS, markets that music is available, album cover, and the International Standard Recording Code (ISRC). To get the genre only with the artist, but for being a list for the artist the selection of music can be deficient, as it is possible for an artist to evolve musically during his career and not get stuck in a style necessarily. [3].

3.2. Apple Music

Access to the MusicKit API, you need to have an Apple developer account to be able to make the requisitions, the account is required so that you can generate the access keys to the application. [17] As the API is still in beta, some features are not as ripe as other APIs, but it is possible, for example, Search music by IRC, by name approach, in addition to user playlists but user information If a new requisition of access to data is required. The information that API returns on music are artist name, album name, cover art and colors, composer name, list of genres that music has, duration in milliseconds, launch date, and an ID for easy localization. [8]

3.3. Deezer

The consumption of information through the Deezer API is due to your API, to access this API requires a developer account that is free. The API offers the search method by music name, the return of the track search method contains information from the album, the artist, gain, bpm, preview with 30 seconds, ISRC, duration, title plus other information related to platform as rank. [9]

4. Proposal

The conception of a playlist for just a minimally accurate user of the information than he likes and from this information recommend what other users who also like that item consume. However, when we want to recommend for groups we come across the situation of what a user likes the other hates and riding a playlist that will be consumed by all users, for streaming we still have the availability barrier of particular music on the platform that the user consumes. To minimize this problem is proposed an open-source platform for group playlists generation through community-developed algorithms to supply from the lack of information between platforms, such as audio characteristics that are available only in Spotify [3], Or even the search for similarity available at Deezer [9] and Apple Music [10].

Only the Spotify API returns the user's most listened songs, the Deezer API has a list of recommendations with what the user is interested in, know as Radio, the Apple Music API has several restrictions, but because being in beta is expected This behavior [3].

All 3 APIs [8] [9] [3] return to the user the ISRC information, with the Apple Music API search methods by ISRC or by the name of the band returning what makes it easy to build our database. Common information between all APIs, is the artist name, album name, track name, ISRC, and duration. In order to be able to use any machine learning algorithm, more information is required in the vector of features, this additional information is only available through the Spotify API being them danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence and tempo. [18]

5. Architecture

It is an open-source client-server platform based on GraphQL, a query-based query language developed by

Facebook in 2012 and publicly released in 2016 [19]. An alternative to the REST architecture, but with the ability to query REST APIs. The server is responsible for processing user requests through queries sent in client applications. These requests are intended to collect information from the streamers that the client has.

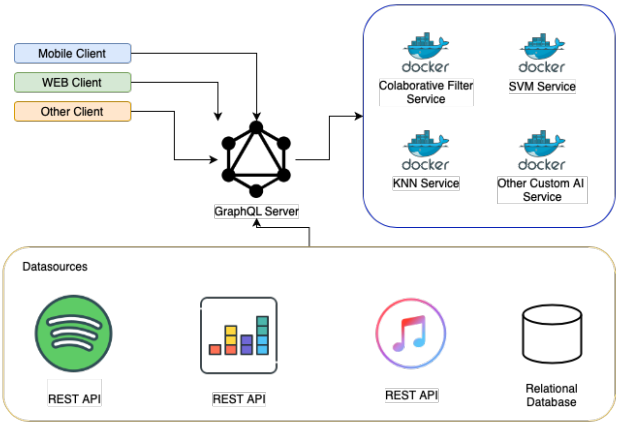

Figure 1: Architecture

6. Modules

6.1. GraphQL Server

For the GraphQL server, we use the Apollo library, officially recommended for applications in Node.JS, this module is responsible for being the clients' access point, the connection with other REST APIs, and the handling of different data sources. Your query is done through queries executed by the client, returning only what is necessary for each request made. [15]

6.2. GraphQL Client

Using Apollo as a GraphQL library makes it easy to integrate with iOS, Android, and Web platforms, as there is a repository for each client, thus facilitating integration, even if it is a React.JS application as a native Android or iOS application, just needing to install the library on the desired platform. Other services can also consume the API through the application's access point using the web client.

6.3. Data Sources

The data sources we use for the platform are Spotify, API where we get the audio features, Deezer and Apple Music, where we get tracks similar to that title, to save the relationships between players in the platform's database, making it easier the execution of the recommendation algorithms, making the default input.

6.4. Services

The services are run inside Docker containers in order to have the necessary isolation, having a playlist as input standard, but also a certain algorithm can take into account the mood of the participants, for example, using GraphQL it is possible to do this request passing the necessary additional information. For the routing of containers, a web server is used, such as NGINX, also as an image, for example. These services, in addition to processing artificial intelligence algorithms, it is also possible to manipulate data between stream services APIs, such as playlist manipulation.

7. Future perspectives

We will test this platform to validate whether how standardized data is made available allows for some gain in music recommendation systems, we will also test whether the use of the Apollo library brings any improvement in the consumption of third-party API data compared to direct consumption through consumption via REST. Test whether the addition of recommendation models with neural networks or deep learning is easily coupled to the platform.

References

[1] Arnt Maasø and Anja Nylund Hagen. Metrics and decision-making in music streaming. *Popular Communication*, 18(1):18–31, 2020.

[2] Martijn Millecamp, Nyi Nyi Htun, Yucheng Jin, and Katrien Verbert. Controlling spotify recommendations: effects of personal characteristics on music recommender user interfaces. In *Proceedings of the 26th Conference on user modeling, adaptation and personalization*, pages 101–109, 2018.

[3] Spotify web api. `https://developer.spotify.com/documentation/web-api/`. Accessed: 2020-12-22.

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[5] Sriharsha Dara, C Ravindranath Chowdary, and Chintoo Kumar. A survey on group recommender systems. *Journal of Intelligent Information Systems*, 54(2):271–295, 2020.

[6] Conor Hayes, Pádraig Cunningham, Patrick Clerkin, and Marco Grimaldi. Programme-driven music radio. In *Proc. of the ECAI*, volume 2, 1996.

[7] Joseph F McCarthy and Theodore D Anagnost. Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 363–372, 1998.

[8] Apple Music documentation. `https://developer.apple.com/documentation/applemusicapi/`. Accessed: 2021-05-12.

[9] Deezer api documentation. `https://developers.deezer.com`. Accessed: 2021-02-15.

[10] MusicKit documentation. `https://developer.apple.com/documentation/musickitjs/`. Accessed: 2021-05-12.

[11] Andrew Crossen, Jay Budzik, and Kristian J Hammond. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 184–185, 2002.

[12] Ingrid A Christensen and Silvia Schiaffino. Entertainment recommender systems for group of users. *Expert systems with applications*, 38(11):14127–14135, 2011.

[13] Mike Gartrell, Xinyu Xing, Qin Lv, Aaron Beach, Richard Han, Shivakant Mishra, and Karim Seada. Enhancing group recommendation by incorporating social relationship interactions. In *Proceedings of the 16th ACM international conference on Supporting group work*, pages 97–106, 2010.

[14] Sarik Ghazarian and Mohammad Ali Nematbakhsh. Enhancing memory-based collaborative filtering for group recommender systems. *Expert systems with applications*, 42(7):3801–3812, 2015.

[15] Apollo framework. `https://www.apollographql.com`. Accessed: 2021-06-20.

[16] Wei Zhou, Li Li, Min Luo, and Wu Chou. Rest api design patterns for sdn northbound api. In *2014 28th international conference on advanced information networking and applications workshops*, pages 358–365. IEEE, 2014.

[17] Creating an apple music api token for musickit js. `https://leemartin.dev/creating-an-apple-music-api-token-e0e5067e4281`. Accessed: 2021-06-20.

[18] Audio features object. `https://developer.spotify.com/documentation/web-api/reference/#object-audiofeaturesobject`. Accessed: 2021-02-15.

[19] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. Graphql-ld: linked data querying with graphql. In *ISWC2018, the 17th International Semantic Web Conference*, pages 1–4, 2018.