Spotify Songs Recommender

*[1] Abril Soler Calderé, *[2] Marta Comas Matas
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
Universitat Politècnica de Catalunya
*[1] abril.soler@estudiantat.upc.edu, *[2] marta.comas.matas@estudiantat.upc.edu

***Abstract*—This paper presents the whole process developed in order to get a song recommender based on the Spotify histories of similar users. After obtaining the last year histories of a group of friends, the aim is to find the one with the most similar tastes in music for a user. Once done, the user will be recommended the songs that he or she has not heard, more played by his or her friend.**

***Index Terms*—Spotify, recommender, recommendation, user, song, genre**

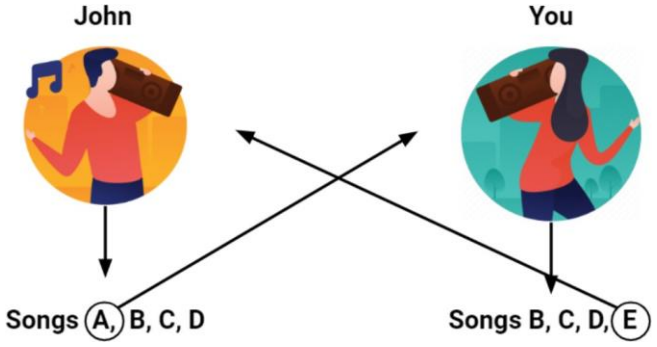

Fig. 1: Goal of the project

I. Introduction

Music is one of the most popular sources of entertainment today. As many of us can assume, the number of songs that exist today is immense. On Spotify we can find, nowadays, around 70 million songs. In this project, a song recommender has been built to avoid wasting time looking for new songs that we might like.

II. State of the Art

A. Recommendation systems

There has been a growth in interest in Recommender Systems in the last two decades (Adomavicius and Tuzhilin, 2005), since the appearance of the first papers on this subject in the mid-1990s (Resnick et al., 1994). In a very general way, recommender systems are algorithms with the aim of suggesting relevant items to users. These items can be books to read, movies to watch, etc. In this case, they are songs to listen to.

Two types of recommenders can be distinguished, collaborative filtering and content-based approaches. The first one is based on a set of user ratings on items while the second one uses item content descriptions and user thematic profiles. It can also be a hybrid model, which combines both. In this project, collaborative filtering has been used.

In collaborative filtering, the input to the system is a set of user ratings on items. Users can be compared based upon their shared appreciation of items, creating the notion of user neighbourhoods.

Recommendation systems are very important in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors.

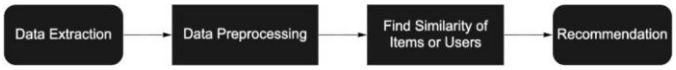

Fig. 2: General Recommendation System Pipeline

B. Spotify

Spotify is a Swedish audio streaming and media services provider founded by Daniel Ek and Martin Lorentzon in April 2006. It is the world's largest music streaming service provider. Users of the services just need to register to have access to one of the largest collections of music, podcasts and other audio content.

It works on a freemium model. It has over 381 million monthly active users, of which 72 million has the premium version.

Below, it can be observed the Spotify key statistics:

- Spotify generated €7.85 billion revenue in 2020, a 16% increase year-on-year.
- Spotify has never published an operating profit. In 2020, it posted a €581 million loss.
- 70 million songs are available on Spotify and 2.9 million podcasts.

C. Spotify recommender

Spotify already has implemented a song recommendation algorithm. It recommends songs when creating a playlist or with Spotify Radio, where you can listen to songs based on any artist, album, playlist or song.

Spotify's recommendations are mostly governed by an AI system called 'Bandits for Recommendations as Treatments' or simply known as BaRT. There are two concepts involved in BaRT: exploit and explore. While exploiting, Spotify uses all activities produced by a user. Exploit usually uses the history of the user, skipped songs, created playlists, social media activity and even location to recommend music. While

exploring, Spotify studies the rest of the world. It looks for playlists and artists similar to the user's taste and it also looks at the popularity of other artists in that area that the user has not heard of or any other related works.
As it has been explained before, there are two different classes of recommendation systems: collaborative filtering and content-based filtering. Spotify uses both, a hybrid recommender system. Spotify also employs Natural Language Processing (NLP). These models analyse news, articles, blogs and reviews written on the web about specific songs or artists.

III. Proposed approach

The aim of this project is to build an algorithm to recommend songs based on Spotify histories of similar users. To achieve that, the algorithm must treat the data (14 histories of users), and run it through a scoring system. In this way, each song will be punctuated based on the times that it has been played, and also by its genre popularity. Then it will be runned through a function that will return the songs recommended to a user by its most similar friend (The amount of songs to recommend and the user are specified in the inputs of the function). Finally, the result is tested using cross validation.

A. Interest

Numerous databases of streaming histories and its punctuations are available already, but this study focuses on two principal characteristics.
First, the aim of this study is to find similarities between a group of friends by studying the songs that each user has listened to during the last year on Spotify.
Second, in this case the punctuation is designed to evaluate each song depending on the user history, as will be explained in the next chapter.

B. Methodology

To develop our song recommender system, which will focus on the similarities between the streaming histories of the users, we pursue the following methodology:

1) Data acquisition
2) Data treatment
3) Recommender method
4) Parameters of the chosen recommender

1) Data acquisition: Every Spotify user, premium or not, has access to request some personal information, including its streaming history of the last year. The data collection of this project has consisted of asking our friends to request their streaming history, so they can be included in the database of our project. In total, the database finally consists of 24 streaming stories and one containing around 7000 to 1000 songs.
Each row of all streaming history files are a song listened by the user, about the songs, the information we have is the time it was reproduced, the artist name, the song title and the milliseconds the song was played.

2) Data treatment: There are numerous ways to extract a qualification of music, for example the number of reproductions that the specific song has on its Youtube videoclip or the times it has been reproduced on an app like Spotify or Apple Music, but in this project the punctuation to a song depends on how many times the listener has reproduced the song and also the genres that the user listens more, are an added punctuation for the songs.
For extracting a punctuation, a recurrence function is used to apply the same punctuation method to all users. This method consists of first, creating a score that is related to the times the song has been played on their Spotify, the most listened song has a 5 and the least has a 1, in between the punctuation is distributed. We prioritise the times the song has been played rather than the seconds because longer songs would be better punctuated than the short ones, and we understand that if a user doesn't like a particular song, the times that this song has been played will be one or closer to one.
The second step is to punctuate each song by the popularity of its genre. The more a genre is listened to, the better punctuation a song belonging to that genre will have. But genres don't come with the streaming history but with developer credentials, it is possible to access to the spotipy library, a python library that gives access to some of the Spotify database, there by using the searching engines, the genres on which song is classified are available. Once each song of the streaming history is related to a list of genres, a punctuation can be created based on the popularity of every particular genre on the history of each user.
Once both punctuations are created, they are ponderated by giving the 60% of punctuation by the times listened and 40% by the genre popularity. Then this punctuation is added as a column to the database and the columns of the time it was played, the name of the artist and the milliseconds played are removed.
Another data treatment is the creation of a dictionary relating the songs of all the histories to a numeric code, the name of the songs in the main database can be substituted by a number, speeding up the following processes.

3) Recommender method: As said in the state of the art, there are some methods that already exist for the purpose of recommending systems based on a database of users.
For the recommender system, we can evaluate two kinds of methods, content-based methods or collaborative filtering. The first kind of methods, the content-based filters, use the characteristics of the features to recommend other items similar to what users like.
The second kind, the collaborative filtering uses similarities between users to recommend items to the user.
The content-based filtering would be adequate if a music database was available and the objective was to recommend music to the user based just on its historial, but as the objective is to recommend music to a user that a similar user already likes, the collaborative filtering is used.
A python scikit for recommender systems that already is available is the surprise library, which includes some methods

and the option to have control over those methods by editing some of the algorithm's parameters.

Some of the recommender methods that are present in the surprise library are:

- Algorithm Base class module that defines the base class called AlgoBase from which every single prediction algorithm has to inherit, the prediction is based on a normal distribution where the mean and the variance are estimated from the dataset using the Maximum Likelihood Estimation:

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui} \quad (1)$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})^2}{|R_{train}|}} \quad (2)$$

- BaselineOnly algorithm which predicts the baseline estimate for a given user and item. Estimating the prediction $\hat{r}_{ui}$ and the sum of the mean M and the bias for the user b_u and for the item b_i:

$$\hat{r} = b_{ui} = \mu + b_u + b_i \quad (3)$$

- KNN inspired algorithms that are directly derived from a basic nearest neighbours approach. There are some kinds of kNN including KNNBasic, which is the simplest one, KNNWithMeans that takes into account the mean ratings of each user, the KNNWithZScore that takes into account the z-score normalisation of each user and finally the KNNBaseline which adds the baseline (calculated as in the Baseline only algorithm).
- SVD (Singular Values Decomposition) algorithm is based on the SVD of algebra for matrix decomposition. It uses a matrix structure where each row represents a user, and each column represents an item and its elements are the rating that the users have given. This matrix is factored by the singular value decomposition to find the factorization of the user-item-rating matrix. This method is one of the most widely used methods.
- SVD++ algorithm which is an extension of SVD taking into account implicit ratings.
- NMF, that is a collaborative filtering algorithm based on Non-negative Matrix Factorization.
- SlopeOne that is one of the most simple methods.
- CoClustering where the items and the users are assigned to some clusters, C_i and C_u, and some co-clusters C_{ui} and the prediction is computed as:

$$\hat{r}_{ui} = \bar{C_{ui}} + (\mu_u - \bar{C_u}) + (\mu_i - \bar{C_i}) \quad (4)$$

Also, the surprise library also includes functions for evaluating the methods proposed before based on accuracy. Some of the metrics available are Compute RMSE (Root Mean Squared Error), Compute MSE (Mean Squared Error), Compute MAE (Mean Absolute Error), Compute FCP (Fraction of Concordant Pairs).

For choosing the algorithm to predict the most liked songs, the Root Mean Square Error (RMSE) method for evaluating accuracy is used, because it is one of the standard ways to evaluate the methods. The different recommender methods are evaluated with the RMSE by comparing the results with the database with the data of the real users and its punctuations. This final database, is composed of the rows user, item and rating, then the accuracy evaluation with RMSE has been done with some of the recommender methods explained before. In particular the methods that have been tested are SVD, NormalPredictor, KNNBasic, KNNWithMeans and the KNNBaseline. The SVpp hasn't been tested because it was computationally expensive and took too much time to evaluate. Then the results of the evaluation are shown in the next table:

Algorithm	test rmse	fit time	test time
SVD	0.544345	4.474368	0.4168228
KNN Baseline	0.619550	0.540391	1.840594
KNN with Z score	0.652081	0.166307	1.755256
KNN with means	0.652291	0.104341	1.695025
KNN basic	0.652971	0.071450	1.499665
Normal Predictor	1.164675	0.148505	0.318984

TABLE I: Results of the evaluation

The method that gives the best results of RMSE is SVD and even though it takes much more time to fit than the others, it is the method that will be used in this project.

4) Parameters of the chosen recommender: Once the SVD method has been chosen, its parameters have to be adjusted. To understand the parameters, it's necessary to understand how the SVD method works.

The SVD has its predictions based on the following formula:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \quad (5)$$

Where the $\hat{r}_{ui}$ is the prediction, the b_u and b_i are the user and item bias and the q_i and the p_u are the user and item factors. To estimate all the unknown, the regularised squared error, which is calculated as the next formula shows) is minimised:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r_{ui}})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \quad (6)$$

Iterating b_u, b_i, p_u and q_i are recalculated by a very straightforward stochastic gradient descent, where the error is calculated as the actual value minus the predicted value.

These steps of stochastic gradient descent are repeated over n_epochs times, a parameter that will be evaluated, its default value is 20.

Another parameter is Ir_all, which is the learning rate for all parameters, the default value is 0.005, and the last parameter that will be evaluated, is reg_all which is the regularisation term for all parameters, the default value is 0.02.

The parameters commented before are tested with these values:

- n_epochs: 5, 10, 20, 30
- Ir_all: 0.001, 0.002, 0.005, 0.01
- reg_all: 0.001, 0.02

The results are:

- n_epochs: 30
- Ir_all: 0.005
- reg_all: 0.001

Because of its definition, the n_epochs parameter will be better as it's bigger, but the problem here is that it's very computationally expensive, the n_epochs is assigned as 20. The same happens with the Ir_all parameter, as the learning rate is bigger, the prediction method is better. So the parameters finally chosen may not be the best ones, but the ratio of accuracy and time of the prediction is the optimal.

IV. Implementation and results

As it is explained in the data treatment paragraphs, the fist step of the implementation is to assign a punctuation, but this has to be done separately due to the high computational power needed to evaluate each user and their songs.

The final implementation of the recommender system is done by creating a function, that as inputs has the user to give a recommendation, the predictions, which are the output of the trained algorithm and the number of songs you want to include in the recommendation.

With these inputs, the function searches which song is each code that the algorithm has given as a recommendation and transforms them into the real titles of the songs, and prints them.

To train the algorithm the fit() function has been used, by previously declaring that the method used is the SVD and working with the previous data frame converted to a dataset. Finally, the result is tested using cross validation, using five folds. So the final output of this code is the number of recommended songs the user has chosen and the result of the cross validation.

V. Conclusions

One of the conclusions of this project is that the library surprise is a very powerful tool for implementing machine learning but, the better the results are, the more computationally expensive the code is, so the equilibrium is between having the best results and having the fastest results, there isn't a specific point that is correct, it all depends on the objectives of the project. Here the parameters have been chosen to optimise the process, make it possible in an average computer and also make it with a good level of accuracy. To do so, some methods have been tested and once the method has been chosen, some parameters have been chosen by taking into account the acceptable time to produce a recommendation.

As the next step, it would be interesting to use this code to create an application where you can make groups and when all the members upload the streaming history, the algorithm gives personal recommendations based on your close friends.

References

[1] Iqbal, M. (2022, May 4). Spotify Revenue and Usage Statistics (2022). Business of Apps. https://www.businessofapps.com/data/spotify-statistics/

[2] Balaganur, S. (2020, September 15). How Spotify's Algorithm Manages To Find Your Inner Groove. Analytics India Magazine. https://analyticsindiamag.com/how-spotifys-algorithm-manages-to-find-your-inner-groove/

[3] Hug, N. (2016, December 28). prediction_algorithms package — Surprise 1 documentation. Prediction algorithms Package. https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html

[4] Content-based Filtering — Recommendation Systems —. (2021, February 5). Google Developers. https://developers.google.com/machine-learning/recommendation/content-based/basics

[5] Collaborative Filtering — Recommendation Systems —. (2021, February 5). Google Developers. https://developers.google.com/machine-learning/recommendation/collaborative/basics

[6] What are concordant and discordant pairs? - Minitab. (n.d.). (C) Minitab, LLC. All Rights Reserved. 2022. https://support.minitab.com/en-us/minitab/18/help-and-how-to/statistics/tables/supporting-topics/other-statistics-and-tests/what-are-concordant-and-discordant-pairs/