Capstone recommendation project

Music recommendation system
Final submission

Massachusetts
Institute of
Technology

Luis Alberto Vázquez Alfonsín

1 Executive summary

Music recommendation is performed on platforms like Spotify and Last.fm and every day becomes more and more important, as the number of songs grows and the songs exploring time is critical for a given user. This id counting of the songs which a given user listen to can be very large and sparse on the user-item matrix. Given that, a model is built to recommend songs which their users neighbors or neighbors or neighbors listen as well.

Music recommendation becomes important when **people don't have time to listen or explore** the enormous variety of music content available, and the help of automated recommender systems can improve the exploring time, effectiveness and quickness.

Searching for nice music becomes a challenging and tedious task which takes a lot of precious time. Platforms like mentioned before recommend music based in historical data and predict suitable songs that the user is likely to enjoy. One very basic recommendation feature is predicting the next song to play (with a maximum likelihood with the user), or recommend new bands similar to the bands which the user usually prefer. Another interesting feature could be recommending music attending to the songs or albums genre information (indie, rock, pop, etc.).

The main objective is to offer a service for **users who listen to songs**, **offering suitable recommendations that they may like**. The music recommendations system is aim to help the **user discover songs based on his neighbors music preferences**, specifically how much specific songs they listen to (number of counts per song and user becomes relevant in this context).

It has to be collected **historical data** related to the number of songs played and the songs details (year, title, etc.). The user-item matrix is very sparse, so it would be a good idea to save the model to avoid extra processing time, for example use the pickle variable type in Python.

It has to be **efficient and quick**. Also, ideally, the **recommendation system should be dynamic**, for **recommending lastest released songs and catch the local or global trends**, so I would be wise to work with last.fm or Spotify API's. This actual dataset is no dynamic, and only recommends songs listened in the past (years ago).

The dataset does not contain any music genre information, and this is very **limiting**; so, if we would know the music genre the quality of the recommendation system would improve a lot. Every person has specific music tastes and prefer a specific genre over others.

Finally, it could recommend songs using NLP techniques, recommending songs similar to another which contains certain lyrics, title, etc.

2 Problem and solution summary

The **datasets** are very large (more than a million rows!), but not every song is listened by every user and not every user listen to all the songs, so the datasets **are sparse**, one could say. The two working datasets are this:

- **song_data**, with a collection of songs id, title of the song, name of the released album, name of the artist and release year.

- **count_data**, with a collection of songs id, users id and the play count (number of times that the user has listen to a specific song).

With **little information** (song id, user id, play frequency, song title, release, artist name and year) it can be built an **interesting recommendation system**.

The negative part of the problem is that the **datasets needed to build the recommendation systems are very big** (a million rows!), and the main part of the rows are not used, so **datasets usage are not optimized**.

-

The data is very sparse, not every user listens to every song and not every song it's listened by every user. Also, there is the need of a **cutoff for performance issues to filter only songs which are listened a certain number of**

times. And the main part of the rows is not used, so **datasets usage is not optimized**.

It would be a wise idea to **set a cutoff based on the behaviour of the song listeners**, for example take the average of listened songs in a day, and from that, set the cutoff to a specific number, both for time performance and for better recommendations issues.

Ideally, the **recommendation system should be dynamic**, for **recommending the lastest released songs and catch the local or global trends**, so working with last.fm or Spotify API's should be a good idea. This dataset is no dynamic and recommends only songs listened in the past (years ago).

In this music recommendation work several recommendations techniques have been used to recommend content to users, like:

→ **User-User similarity-based collaborative filtering**

→ **Item-Item similarity-based collaborative filtering**

→ **Model-based collaborative filtering**

→ **Cluster based recommendation**

→ **NLP recommendation.**

The **best performance in terms of F1 score is for the user-user recommendation system**.

The cluster based recommendation could be used **for recommending certain type of music** to clustered based users (for example, clustered users by music genres.

The principal model that shall be used on the final approach should be **user-user similarity-based collaborative filtering**, not only because it has given the best outcomes, otherwise it may be **important to follow recommendations from people with the same musical tastes than you**; it seems natural that if you like certain type of music you will find more interesting recommendations listening to some bands that your similar-music-taste peers listen to.

It could be used as well item-item similarity-based collaborative filtering and **cluster based recommendations to filter genres of music**, clustering people which listen to similar type of music together (genres). As well, I would use NLP recommendations for lyrics, recommending similar lyric songs for users that find this issue important.

3 Recommendations for implementation

The key aspects to take into consideration for implementing are quickness, effectiveness on recommendation (a good F1 score value), gathering the correct data to recommend songs, choosing the right algorithm to produce recommendations and offering possibilities to gather historical data from other online music service providers (Spotify or Last.fm).

Some important features to attract users, advertising or monetize the system would be possibilities of discovering new exciting music, keep track of the most recent music releases, keep track as well of the global or local trends (or from your friends) and improving your musical culture without any effort.

The **costs are only time to register to the service, pay money or permit advertising**, meanwhile **the benefits are keeping track of music trends** (also among your friends), discovering new music, improve your time for discovering new music, getting personalized music recommendations and exploring in a deep way your musical tastes.

The **principal risks are not getting good recommendations** (mainly get recommendations that an user dislikes), time performance issues and not having enough or recent data.

Some challenges could be picking a minimum number of play count for a song to be considered (a cutoff), picking as well the number of recommendations for not overwhelming the user, **getting data from several online**

music platforms, the processing time to obtain recommendations and **keeping track of recent songs** recommendations.

In order to be **time and storage efficient** the model could be saved in a pickle variable. Also making the model accessible via web services to obtain a multiplatform service (PC, Mac, Linux, Android, IOS, etc).

Finally, it must be investigated to use another **more recent dataset, which includes music genres and more information** to take into account to do, for example, some cluster based recommendation. Also including a feature for special songs that a user likes a lot could be interesting (similar to the obsession feature of last.fm).