2022

Music Recommendation System

FINAL REPORT

ANDRÉS ALVEAR

Music Recommendation System
Final Report

Executive Summary:

This project proposes a Music Recommendation System to propose the top songs for a user based on the likelihood of listening to those songs. Our motivation consisted of building an information filter to maintain the attention of users of a Music Platform because of information overload and decisions that they make, avoiding attrition of users using the Music Platform, this is of course undesirable.

The constraints that we main follows given the data that we have, it's the provision of the characteristics of a small dataset constituted mainly of a small number of users and songs of the universe of possibilities of the Million Song Dataset [1].

There are many types of recommender systems in use today, so a major component of the project consisted in benchmarking models to find the more appropriate recommendation system to propose songs for a user based on the user's listening activity of a given song dataset to recommend songs to users based on their previous listening activity using Rank-Based, Similarity-Based & Matrix Factorization, Clustering-Based & Content-Based.

Overall, the **optimized user-user similarity-based recommendation system** has given the best performance in terms of the **F1-Score** = 0.513, i.e., 51,3% indicates that half the recommended songs were relevant and not relevant songs were recommended all the times to the user. The RMSE= 1.0521, here the smaller the better but there are small differences between the RMSE metrics of the recommendation systems evaluated. In this scenario, it means that this metric is not relevant or even significant to compare the models. We are getting a **recall** of 0.637, which means out of all the relevant songs, 63.7% are recommended. We are getting a **precision** of 0.43, which means out of all the recommended products 43% are relevant.

This solution with **F1-Score** = 51,3% means that the model is performing not very good, so it might be improved by using GridSearchCV by tuning different hyperparameters of this algorithm. In terms of recommendation is slightly better than other recommendation systems built. However, for the given dataset we cannot improve the model combining **optimized user-user similarity-based recommendation system** with a Content-Based approach due to lack of user engagement data. The lack of text data about the user prevents the creation of a hybrid recommendation systems. Therefore, to improve the Music Recommendation System the dataset should be extended by adding user engagement data and categorical data for each song. By adding that data to the dataset we can take advantage of the already built models to provides accurate recommendations based on the listening activity of the users getting an improved models because we will have more data of the user interaction with the music platform.

Problem Summary:

Online music streaming platforms like **Spotify, Youtube Music** have plenty of songs in their repositories. A **recommendation system** to recommend songs to the users based on their historical interactions with songs would improve customer satisfaction. Increased customer satisfaction will **increase the revenue of the company**. The techniques that we explored in this project it is not limited to songs but can be any item for which you can build a recommendation system.

In this project, we built a music recommendation system for the Million Song Dataset. Our motivation consists of building an information filter to maintain the attention of users of a Music Platform because of information overload and decisions that they make, avoiding attrition of users using the Music Platform, this is of course undesirable.

The constraints that we main follows given the data that we have, it's the provision of the characteristics of a small dataset constituted mainly of a small number of user_id and song_id of the universe of possibilities of the Million Song Dataset.

There are many types of recommender systems in use today, so a major component of the project will be in benchmarking models to find the more appropriate recommendation system to propose songs for a user based on the user's listening activity of a given song set.

Solution Summary:

In this project we will build a music recommendation system for the Million Song Dataset to recommend songs to users based on their previous listening activity using six different algorithms. They are as follows:

1. Rank-based using averages
2. User-User similarity-based collaborative filtering
3. Item-Item similarity-based collaborative filtering
4. Model-based collaborative filtering - Matrix Factorization
5. clustering-based recommendation systems
6. content-based recommendation systems

To achieve that goal, we load necessary libraries and import the data. Drop unnecessary columns from the Million Song Dataset to understand the data and get insights, pre-process the data. Then, we explore the dataset using some plots and basic data analysis techniques.

Then, we build a baseline recommendation system, then we build recommendation systems using the six algorithms mentioned. The **surprise** library was used to

demonstrate "user-user similarity-based collaborative filtering," "item-item similarity-based collaborative filtering," and "model-based collaborative filtering (matrix factorization)", clustering-based recommendation systems. Grid search cross-validation was applied to find the best working model, and with that the corresponding predictions are made.

To evaluate the performance of these models, **precision@k and recall@k** with k=10 is used in this project. Using these two metrics F_1 score is calculated for each working model. precision@k is a modified performance assessment metric for recommendation systems. Here, k is the number of items (songs here) recommended to a user.

For the present case precision@10 can be interpreted as the fraction of the 10 recommended songs that are liked/listened by the user. If the user listens to 6 songs, then precision@10 will be 6/10, i.e., 0.6 (60%).

We further improve the performance of these models using hyperparameter tuning.

Analysis and Key Insights:

Original Dataset:

The core dataset is the Taste Profile Subset released by The Echo Nest as part of the Million Song Dataset [1]. There are two csv files in this dataset **count_data** and **song_data.** song_data file of 1,000,000 rows of songs contains the details about the song id, titles, release, artist name, and the year of release. The **count_data** file of 2,000,000 rows contains the user id, song id, and the play count of users.

song_data

- **song_id:** A unique id given to every song
- **title:** Title of the song
- **Release:** Name of the released album
- **Artist_name:** Name of the artist
- **year:** Year of release

count_data

- **user_id:** A unique id given to the user
- **song_id:** A unique id given to the song
- **play_count:** Number of times the song was played

What fraction of **listened songs** are known?

Unique users = 76,353

Total **possibility of listened songs** = 76353*10000 = 763,530,000

Unique Songs = 10,000

Number of play_count > 0 = 2,000,000

Fraction known = 2000000 / (76353*10000)= ~0.002619 or ~0.3%

i.e. 3 in every 1000 **listened songs** is known, rest are unknown

Finding these unknown listened songs is the primary goal of Recommendation Systems

Final Dataset:

The core dataset is the Taste Profile Subset released by The Echo Nest as part of the Million Song Dataset [1]. However, we will use a subset generated in Milestone 1. There is one csv file in this dataset **user_song_data.** user_song_data.csv file of 117,876 rows of songs contains the details about the user id, song id, play count, title, release, artist name, and the year of release.

user_song_data

- **user _id:** A unique id given to the user
- **song_id:** A unique id given to every song
- **play_count:** Number of times the song was played
- **title:** Title of the song
- **release:** Name of the released album
- **artist_name:** Name of the artist
- **year:** Year of release

The constraints that we main follows given the subset that we have, it's the provision of the characteristics of a small dataset of 117,876 user_id and song_id pairs of the universe of possibilities of the Million Song Dataset.

How many unique users are in the dataset?

There are **3155 unique users** in the "Million Song Dataset" dataset.

What is the total number of unique songs?

There are **563 unique songs**

Observations and Insights:

- As per the number of unique users and songs, there is a **possibility of 3155 * 563 = 1,776,265 listened songs** in the dataset. But we only have 117,876 played songs, i.e., not every user has listened every song in the dataset, which is quite understandable. This creates the possibility of building a recommendation system to recommend songs to the users which they have not interacted with.
- Because **we don't have rating information we can't tell if the users that listened a song once really like it**.
- There are 117,876 observations and 7 columns in the data.
- All the columns are of numeric data type except the title, release, artist_name.

Comparison of various techniques and their relative performance:

Rank-Based Recommendation System

In Table 1 is presented the dataset based in the count and sum of play_counts of the songs and build the popularity recommendation systems based on the sum of play counts.
The top 10 songs provided by the Rank-Based Recommendation System to users based on their historical listened activity is the following:

1. Joe's Head by Kings Of Leon, in, 2003
2. Secrets by OneRepublic, in, 2009
3. In Person by The Pussycat Dolls, in, 2008
4. If I Ain't Got You by Alicia Keys, in, 2003
5. The Funeral (Album Version) by Band Of Horses, in, 0
6. Weird Fishes/Arpeggi by Radiohead, in, 2007
7. Jamaica Roots II(Agora E Sempre) by Natiruts, in, 0
8. In My Place by Coldplay, in, 2001
9. Black by Pearl Jam, in, 1991
10. Step Through The Door by Soltero, in, 0

Table 1 Rank-Based Recommendation System is built with the average_count and play_freq of the dataset

song_id	avg_count	play_freq
21	1.622642	265
22	1.492424	132
52	1.729216	421
62	1.728070	114
93	1.452174	115

The dimensions of the dataset used for the Rank-Based Recommendation System is 563 rows by 2 columns, i.e. we have aggregated average_count and play_freq for each unique user.

Collaborative Filtering:

User-User Similarity based collaborative Filtering

Table 2 Baseline vs Optimized model

	Baseline Model	Optimized model
RMSE	1.0878	1.0521
Precision	0.396	0.43
Recall	0.692	0.637
F1 score	0.504	0.513

In Table 2 we can observe the comparison of Baseline and Optimized versions of User-User Similarity based collaborative Filtering.

The following observations are for the **Baseline model:**

- We have calculated **RMSE** to check **how far the overall predicted songs** are from the **actual songs**.
- Intuition of Recall - We are getting a **recall of 0.692**, which means out of **all the relevant songs, 69.2% are recommended**.
- Intuition of Precision - We are getting a **precision of almost 0.396**, which means **out of all the recommended products 39.6% are relevant**.
- Here **F_1 score** of the **baseline model is almost 0.504**. It indicates that **half recommended songs were relevant and not relevant songs were recommended all the times** to the user. We will try to improve this later by using **GridSearchCV by tuning different hyperparameters** of this algorithm.

We trained the best model found by gridsearch algorithm, using the optimal similarity measure for user-user based collaborative filtering:

- k = 30
- min_k = 9
- sim_options: {'name': 'pearson_baseline','user_based': True, 'min_support': 2}}

Observations are for the **Optimized model:**

- We can observe that after tuning hyperparameters, **F_1 score of the optimized model is slightly better with a value of 0.513 and the baseline model has a F_1 score 0.504**. Along with this, **the RMSE of the optimized model is 1.0521 vs 1.0878 of the baseline. So, it has sightly gone down compared to the model with default hyperparameters**. Hence, we can say that the model performance has been slightly improved after hyperparameter tuning.

Item-Item Similarity based collaborative Filtering

Table 3 Baseline vs Optimized model

	Baseline Model	Optimized model
RMSE	1.0394	1.0328
Precision	0.311	0.411
Recall	0.442	0.584
F1 score	0.365	0.489

In Table 3 we can observe the comparison of Baseline and Optimized versions of Item-Item Similarity based collaborative Filtering.

The following observations are for the **Baseline model:**

We trained the best model found by gridsearch algorithm, using the optimal similarity measure for item-item based collaborative filtering:

- k = 30
- min_k = 6
- sim_options: {'name': 'pearson_baseline','user_based': False, 'min_support': 2}}

Observations are for the **Optimized model:**

- We can observe that after tuning hyperparameters, **F_1 score of the optimized model is better with a value of 0.489 and the baseline model has a F_1 score 0.365**. Along with this, **the RMSE of the optimized model is 1.032 vs 1.0394 of the baseline. So, it has sightly gone down compared to the model with default hyperparameters**. Hence, we can say that the model performance has been improved after hyperparameter tuning.

Model-based collaborative Filtering- Matrix Factorization

We will perform matrix factorization on a subset of our data to extract latent features for our users and songs.

Table 4 Baseline vs Optimized model

	Baseline Model	Optimized model
RMSE	1.0252	1.0141
Precision	0.421	0.427
Recall	0.548	0.535
F1 score	0.476	0.475

In Table 4 we can observe the comparison of Baseline and Optimized versions of Model-based collaborative Filtering- Matrix Factorization.

The following observations are for the **Baseline model:**

We trained the best model found by gridsearch algorithm, using the followings values:

- k = 30
- min_k = 9
- sim_options: {'name': 'pearson_baseline','user_based': True, 'min_support': 2}}

Observations are for the **Optimized model:**

- Due to the lack of user engagement data both recommendation systems perform poorly. To increase the chance to model relationship between user and songs we might **add more data** (information about user) to get more options to obtain better **similarities** between users and between items.

Cluster Based Recommendation System

In clustering-based recommendation systems, we explore the **similarities and differences** in people's tastes in songs based on how they rate different songs. We cluster similar users together and recommend songs to a user based on play_counts from other users in the same cluster.

Table 5 Baseline vs Optimized model

	Baseline Model	Optimized model
RMSE	1.0487	0.446
Precision	0.405	0.407
Recall	0.496	0.498
F1 score	0.446	0.448

In Table 5 we can observe the comparison of Baseline and Optimized versions of Cluster Based Recommendation System. Due to the lack of **user engagement** data both recommendation systems perform poorly. The only was to address this issue consist in adding user engagement data to the dataset.

Content Based Recommendation Systems

To build the content-based recommendation system we created the feature - text, i.e., the combination of the song's columns title, release, artist_name to find similar songs. Merging the "title", "release", "artist_name" columns we have created a different column named "text" to find out similar songs. We kept only four columns user_id, song_id, play_count, and text (title, release, and artist_name), and we set the title of the songs as the index. Use NLTK and the scikit-learn library to extract features using the TF-IDF technique. Finally, we created a function to find the most similar movies to recommend for a given movie.

The information used to generate the new data set allows us recommending songs only based on the title of a song. For instance, recommending 10 songs similar to **Learn to Fly** by Foo Fighters:

1. Everlong by Foo Fighters
2. The Pretender by Foo Fighters
3. Nothing Better (Album) by Postal Service'
4. From Left To Right by Boom Bip
5. Lifespan Of A Fly by the bird and the bee
6. Under The Gun by The Killers
7. I Need A Dollar by Aloe Blacc
8. Feel The Love by Cut Copy
9. All The Pretty Faces by The Killers
10. Bones by The Killers

The song 'Learn To Fly' belongs to **Rock, Pop** genres, and the **majority of the recommendations** lie in one or more of the genres **Rock, Pop, Indie pop, electronic**. It implies that the resulting recommendation system is working well is highly probable that people love the recommended songs.

Conclusions:

Overall, the **Optimized User-User similarity-based collaborative filtering** has given the best performance in terms of the F1-Score = 0.513. In terms of recommendation is slightly better than other recommendation systems built. It is concluded that with the given data set the best recommendation system based on the comparison of RMSE, precision, recall, and F1 score metrics was better because the play_count metric is the only user engagement metric.

The dataset contains enough information about the users' interactions with the songs to build six different music recommendation systems for the Million Song Dataset. Every register in the dataset is composed by (user_id, song_id, play_count, title, release, artist_name, year), that information was used to recommend songs to users based on their previous listening activity using Rank-Based, Similarity-Based & Matrix Factorization, Clustering-Based & Content-Based. It is shown that the more songs the user listens to and the more times he listens to them, the more similarities between users can be found by detecting the preferences of the songs.

Limitations and Recommendations for Further Analysis:

The Music Recommendation system requires a huge amount of data to provides accurate recommendations based on the listening activity of the users. The number of user and songs pairs is large, so we are training on a small dataset of the universe of possibilities of the Million Song Dataset. More data is necessary for a better F1 score. It is difficult to create a good recommender system with a small amount of data. Although not big enough, the subset of the Million Song Dataset is still very large. The size of the data affects the quality of hyperparameter tuning of the model. Time and computing power limits our ability to use a grid search method for tuning our hyperparameters for a large dataset.

Recommendations and Next Steps:

For the given dataset it is recommended the **optimized user-user similarity-based recommendation system** because that model extract better the similarities between users mainly due to the interactions that are described by the user_id, song_id and play_count. However, the recommendation system can be improved by following the next recommendations:

- We can still try to improve the performance of these models using hyperparameter tuning.

- The dataset lack of **user engagement** information. We might add data from the analytics of the music platform to the current dataset to improve our Recommendation

Systems. So, we don't have access to important information that would model the interactions between user songs to build a **Content based model** using user information such as comments, likes, so on. The only value that we used to build the model was play_count that give us information about the frequency that the user use to listen certain song

- We can enrich the data to get a better chance to get similarities between users-users models, song-song models. For the songs we can add categories such as genres of the music such us, **Rock, Pop, Indie pop, electronic, reggae, salsa.** On the other hand, the **user engagement** data that would help to build better models might be:
    - Listening time and average listening duration
    - Likes, Dislikes
    - Comments
    - ...
- We can try also combine different recommendation techniques to build hybrid recommendation systems, but it only can be possible if we have access to **user engagement** information, due to all the models that were built perform poorly with a maximum F1 score 51.3%, that indicates that **half recommended songs were relevant and not relevant songs were recommended all the times** to the user.
    - To improve the performance of the User-User similarity-based collaborative filtering, user information such as demographics including age, gender, occupation, philosophy of life, style (Street Wear, Punk, Urban Style, Emo, Skater, etc.) is required. Surfer, Biker, Rocker, gothic, emo, cowboy, and so on), profession, work position, and so on. That information might be extracted from the user database under user's consent.
    - To improve the performance of the Item-Item similarity-based collaborative filtering models, in addition to play_count, new columns with genre categories are required (rock, heavy metal, progressive rock, classical, opera, pop, electronic, hip-hop, rap). , trap, reggae, reggaeton, and so on), and themes (romantic love, sex, rebellion against "The Establishment", social concerns, life styles, spiritual, happy, party, and so on) to the data set. By incorporating this information, better Model Based Collaborative Filtering could be created, and this result could be integrated with Content-based recommendation systems to create a hybrid recommendation system.

References:

[1] **Million Song Dataset: http://millionsongdataset.com/**