Songs Recommendation Algorithm with Spark Project Report

Chen Zhang
Department of Mathematics and Statistics

Abstract

Due to a large number of streaming media services such as music and movies generate a large amount of user historical data every day, the recommendation algorithm is not as challenging as it used to be. This article analyzes Spotify data to create personalized song recommendations and enhance the interaction between users and the platform. To better achieve this goal, we use Spotify's API data to extract each track data and store it in the corresponding Postgres SQL database. Then use Pyspark to connect to the database to analyze and process the data. Finally, we will use the least square method ALS and random forest to select the best algorithm to build the recommendation model. And we will fit the corresponding ALS models for Implicit Rating and Explicit Rating respectively. In summary, our project demonstrates how to use Postgres SQL, PySpark, ALS, and Random Forests to build a song recommendation system using Spotify tracker data. Our system provides users with personalized song recommendations based on their listening history.

1 Keyword

Spark, ALS, Recommendation System, Implicit Rating, Explicit Rating

2 Introduction

Under the high-intensity pressure of society, music has become an entertainment activity for most people to relax and decompress. Music is a way of communication without borders, and more and more people like music in an artistic way. However, how to let people hear fresh and favorite music has obviously become a problem faced by major music platforms

Music streaming services such as Spotify thoroughly address the issues mentioned above. Accessing the browsing history of millions of songs helps people explore playlists of different genres and styles to find their favorite music. However, with such a huge amount of music to choose from, it can be challenging to sift through the options and find songs that really resonate with us.

The main idea behind recommender systems is that by analyzing a user's listening history, these systems can recommend new songs or new similar artists they might like. This not only helps users discover new music but also increases their engagement with the platform.

In this project, we mainly use some historical track data from Spotify to build a recommendation system algorithm. We will compare some conventional machine learning algorithms and some popular recommendation system algorithms. And we will combine the Implicit Rating and Explicit Rating mentioned by David M. Nichols[4] to optimize our recommendation algorithm. Ultimately, our goal is to find the best algorithm that can predict users' music preferences with the smallest error and recommend songs they might like. In the project, I will use the Postgres SQL database and Pyspark to analyze the data and generate the final machine-learning algorithm.

Through this project, we will gain a better understanding of how recommender systems work and how they can be applied to real-world problems. I will also practice using some big data software to deal with real big data problems.

3 Related Work

Song recommendation systems have been widely studied and implemented in music streaming platforms such as Spotify, Apple Music, and Pandora. Collaborative filtering is a popular technique for recommendation systems, which involves analyzing user behavior and preferences to provide personalized recommendations. Alternating least squares (ALS) is a commonly used collaborative filtering algorithm that has been employed in var-

ious music recommendation systems[3]. In addition to collaborative filtering, content-based approaches have also been used to recommend songs based on their audio features, such as tempo, key, and loudness. Random forests are a machine-learning algorithm that has been used for content-based music recommendation systems. Hybrid recommendation systems, which combine collaborative filtering and content-based approaches, have been shown to provide more accurate recommendations than using either technique alone[2]. In this project, we will focus on the used collaborative filtering algorithm with an implicit rating and explicit rating.

4 Data Summary

In order to better achieve our goal, we selected the three-year Spotify tracking data from 2010 to 2015. Our data sources are mainly divided into three different databases, namely customer data, track-id data, and each tracking data. We have made some basic statistical information.

4.1 Customer Data

Customer data shows the relevant information of all customer data participating in this project, including customer id, name, gender, address, registration time, and so on. After statistics, I found a total of 5,000 different customers, including 2,984 male customers and 2,016 female customers. For further model training, I will only use 'custid', 'gender' as features for random forests regressor.

4.2 Music Data

The music data shows all the historical music information that 5000 customers may have listened to. It includes the song's id, name, album, and song length information. We found a total of 1345 different pieces of music from 438 different albums. and the average duration of these songs is 241.153.

4.3 Tracker Data

The last tracking information data mainly includes each information on the historical tracking of 5,000 customers, which contains 1,000,000 tracking events. Each tracking event includes event id, customer id, track id, time of listening to the song, whether to use a mobile phone to listen to the song and the address of listening to the song. We found that among the 1 million tracked data, 390,000 of them were music listened to by mobile phones, and 610,000 were not music listened to by mobile phones. This is also a relatively normal phenomenon during 2010-2015.

5 Methodology

To make sure the project will success, the project is divided into four phases: data ingestion and aggregation, exploration data analysis, data processing, model fitting and evaluation. Details of each step will be discussed further.

5.1 Data Ingestion and Aggregation

The data sets we have is from Kaggle, for the convenience of further analysis and model training, I created a Postgres SQL database and import the all data sets. After that, we will use Python to connect to my database and use SQL query to get the data to create the final table we want to use for each step. And the example code of how I created the database is shown below.

```
psql
CREATE USER user WITH \
PASSWORD 'xxx';
CREATE DATABASE database \
WITH OWNER 'user';
psql database user
CREATE TABLE music (
    trackid INTEGER,
    title VARCHAR(2000),
    artist VARCHAR(2000),
    length INTEGER
);
\COPY music FROM 'music.csv'\
DELIMITER ',' CSV HEADER;
```

5.2 Exploration Data Analysis

For the next part of data visualization, we first use the Left Join function to pass our three data through track id and customer id to get the most complete data. Mining this final data reveals some notable features. We found that for customers, Gregory Koval is the person who has listened to the most songs, listening to a total of 1617 different songs. And Paula Peltier is the person who uses mobile phones to listen to songs the most, he used a total of 1060 times to listen to songs. For songs, 'Caught up in You' was the most viewed song, with a total of 4190 unique customers.

Through deeper data mining, I found that the time of listening to songs is very important in-

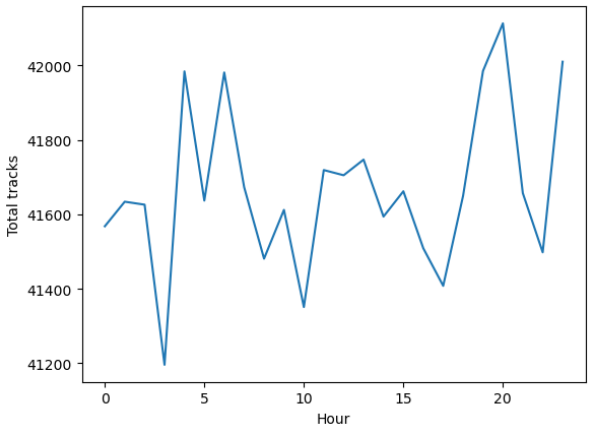

Figure 1: Line plot of total track number within 24 hours

formation, so I dismantled the original data 'date time' and added two columns of new information. One is the time in the 24-hour clock, and the other is the period of listening to music. There are generally only four different time periods, morning, afternoon, evening, and midnight. From figure 1, I found that most people would like to listen to musics during their bed time around 19-20 at evening. Also I created the line plot of total track number within 4 periods, it shows that most people like to listen to music around evening. Then I will use these hour features to fit further random forest regression model.

5.3 Data Processing

Recommender systems commonly use two methodologies, explicit rating, and implicit rating. The explicit rating relies on direct feedback from users such as reviews or ratings. This method is easy to collect but may not fully capture user preferences. Implicit rating, on the other hand, analyzes user behavior to infer their preferences, such as observing clicks or view times. Although implicit rating requires more effort to interpret, it can provide a more accurate representation of user preferences. According to Adomavicius and Tuzhilin[1], the implicit rating is an essential technique for the next generation of recommender systems, as it can collect data passively without requiring direct feedback from users.

In recommendation algorithms, implicit and explicit rates are often used to predict user preferences and make personalized recommendations. Explicit rates refer to explicit feedback provided by users, such as ratings or reviews. These ratings are typically used to train models that predict user preferences based on item attributes and user demographics. However, explicit ratings can suffer from sparsity and bias, as users may not provide feedback for all items they interact with, or may not be truthful in their feedback. On the other hand, implicit rates refer to user behavior and preferences that are inferred from indirect signals, such as clickstream data or purchase history. These signals are used to train models that predict user preferences based on patterns in user behavior. Implicit rates can be useful in overcoming the sparsity and bias issues of explicit ratings, and can also provide insights into user preferences that may not be revealed through explicit feedback. However, they can also suffer from ambiguity and noise, as user behavior may not always reflect true preferences.

For my project, I use this idea to generate a rate similar to the implicit rate based on historical data. In our case, the implicit rate refers to Customer number has listened to the song in a total of n times over the period of t, which n is the rating. The specific calculation method is as shown in the following SQL code.

```
SELECT CustId, TrackID, \
SUM(CASE WHEN Length>0 \
THEN 1 ELSE 0 END) AS rating
FROM final_table
GROUP BY CustId, TrackID
```

Our explicit rate is based on implicit, and we have made a sort. For those who listen to songs less than 2 times, it is 0, between 2 and 4 times I assume it is 1, and between 4 and 7 times which will be 2, more than 7 times will be 3. The specific calculation method is as shown in the following SQL code.

```
SELECT custid, trackid, \
CASE WHEN SUM(CASE when \
length>=0 THEN 1 ELSE 0 END)<2\
THEN 0 WHEN SUM(CASE \
WHEN length>=0 THEN 1 ELSE 0 \
END)<4 THEN 1 WHEN SUM\
(CASE WHEN length>=0 \
THEN 1 ELSE 0 END)<7 \
THEN 2 ELSE 3 END AS rating
FROM final_table
GROUP BY custid, trackid
```

After I got the new implicit rate and explicit rate data. In order to take the historical time into ac-

count, I converted the historical tracking time into the data of the first week of the year. And through the information of the week, the data will be divided into a training set and test set, that is, the data from 40-51 weeks will be used as the training set, and the data from 52 and 53 weeks will be used as the test set.

5.4 Model Fitting and Evaluation

In this part, I used and compared four different models, namely the random forest regression model using implicit rating, the random forest regression model not applicable to implicit rating, the ALS model using implicit rating, and the ALS model using explicit rating. In order to compare the difference between the models more intuitively, I evaluate each model by using RMSE and compare the models. The reason why I choose to use a regression-type machine learning algorithm is that as a recommendation algorithm, giving a prediction score can help do more analysis in the future. It is also more valuable than the score corresponding to the simple classification prediction.

The first model I fit was a random forest regression model using some of the time information we generated, as well as the original information. Through a combination of cross-validation, we found the best random forest model for these features was fitted using 20 decision trees. But the result is not very ideal. In the end, we calculated the root of the mean squared error of the test set, and the specific results are shown in the table below.

Next, I chose to include implicit rate as an additional feature to fit the regression model. The purpose is to observe whether the implicit rating will be of great help to the regression model. Still using a similar approach, I found that the most suitable number of decision trees was still 20, and with that model, I calculated the RMSE of the model at the moment. Although as mentioned before, the implicit rating can reduce the error rate of the recommendation algorithm, its effect is not as good as we expected.

I also tried to fit an alternating least squares model, which updates user and item feature vectors alternately, minimizing the sum of squared errors of rating predictions, to obtain optimal feature vectors. For this method, we must give a rating, so I first tried to fit the model using the implicit rating mentioned earlier. For this model, I only used 'custid', 'trackid', and implicit rating. Combined with the method of cross-validation, I found the best ALS model for this case. and by calculating I got its RMSE to be 1.62.

Finally, in order to compare the implicit rate and explicit which one is more helpful to the recommendation system algorithm, I choose to use the explicit rate as the rate input of the ALS model to fit the fourth model. Still, through cross-validation, I got the most suitable model in this case. From the table below, I can also find that comparing the four different models, the RMSE of the last one is the smallest. From the introduction, we can also conclude that the ALS model algorithm is more suitable for recommendation algorithms than ordinary machine learning algorithms. And the model fitted with an explicit rate will work better than the model fitted with an implicit rate.

Model	RMSE
Random Forest without implicit rate	485.64
Random Forest with implicit rate	492.63
ALS with implicit rate	1.62
ALS with explicit rate	0.33

Table 1: RMSE for four different models

6 Discuss and Limitation

Finally, through comparison, I found that implicit rate and explicit rate are not very helpful to ordinary random forest regression models. But for the ALS model, this is a very good means to fit the recommendation algorithm model. In the end, I found that the ALS model fitted with an explicit rate will more accurately recommend relevant music based on the customer's preferences.

But for my project, there are also some limitations, because, of time, I did not add the process of the data stream to the pipeline. So in the next step, I will mainly focus on how to improve the process of the data stream. Second, there may still be some problems if the recommendation is only made from the perspective of historical data, so next, I will try to analyze the audio data of music, and then combine it with the ALS model to make a more complete hybrid model.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Arodh Lal Karn, Rakshha Kumari Karna, Bhavana Raj Kondamudi, Girish Bagale, Denis A. Pustokhin, Irina V. Pustokhina, and Sudhakar Sengan. Customer centric hybrid recommendation system for e-commerce applications by integrating hybrid sentiment analysis. *Electronic Commerce Research*, 23(1):279–314, 2023.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] David Nichols. Implicit rating and filtering. *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36, 1998.