Sami Chowdhury
Project 423006267

Project Report

Problem:

An increasing amount of musical information is being published daily in media like Social Networks, Digital Libraries or Web Pages. All this data has the potential to impact in musicological studies, as well as tasks within MIR such as music recommendation. Making sense of it is a very challenging task, as there are numerous ways to rate music and how to recommend it. Programs like Spotify and iTunes use recommendation systems to better enhance the listening pleasure of users. They want to ensure that the user continuously uses their application by recommending them the music that they suspect the user will like the most.
That is why it is imperative that they create a recommender system for new musical artists based on the past music they have listened to. Most of the methods that address this problem are content-based and require item information that is not always available. Another big drawback is how hard it can be to get explicit feedback from users for some models, resulting in the sparsity problem. My goal is to try and create a recommender system similar to the ones Spotify or iTunes would use.

Approach:

There were numerous approaches that were thought of when tackling this problem.
Originally, the algorithm I planed to use for my system was an approach focused on using knowledge-based graphs and vectors. Using a linear-chain conditional random field for my project would have produced results that would make the most sense.

Over time, however, I found that this model proved too difficult and cumbersome for my current skillset to understand. Instead, I decided to use collaborative filtering to work with my project. The model is based on user and artist characterization. Collaborative filtering is one of the most popular recommender system techniques and is being extensively used in many applications such as in music and movies streaming applications, in ecommerce industry for product recommendations, and used in restaurant and place recommendations.

Sami Chowdhury
Project 423006267

In this this application, I have used spark and the collaborative filtering technique to build an artist recommender system, which suggests users, artists based on his/her and other's listening history. I have used the Spark Alternating least square algorithm library to build our model.

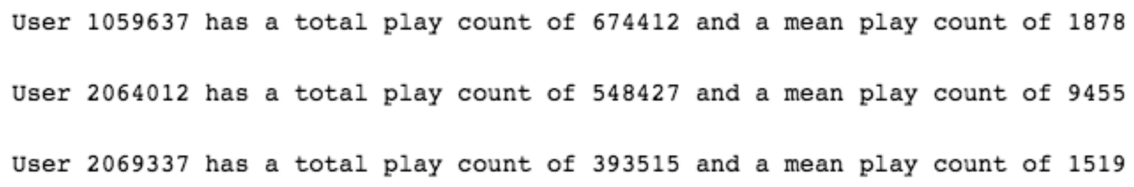
This dataset I had used is from the publicly available song data from AudioScrobbler. The dataset included here is a trimmed version of the original dataset containing information about the 50 most active users. The original dataset contains information about 141K users, and 1.6 million artists.

My implementation followed a basic set of steps to achieve the goal. The first step was to read all three dataset files in RDDs. From the files with the text files with each user's unique artistID and name, I read the file and create a map with the key from the artist ID and value to the artist name. AudioScrobbler has a file which contains information about an artist's other alias' / misspelled names. This information is used to correct the user-artist information by replacing the aliases by its uniqueId. Then the correct entries are mapped in the RDD. A new RDD is then created based on user listening stats. Basically, the data in the RDD's is filtered, manipulated, and transformed to extract information about users like artist play, counts, mean play count, and other data. These RDD's are then split into training, validation, and test RDD's. The model is then trained using the trainingRDD function.

The model's evaluation is done by calculating the fraction of the overlapping artists between the predicted artists and the true artists. This fraction is then summed up for all users and the final score is the sum normalized by the total number of users. In my evaluation model, all artists listed in the training data for that particular user are purged while making the top-K predictions in order to reduce bias in error estimate. The distinct artist IDs are then fetched. The model is then evaluated on the validation dataset based on different rank parameters corresponding to the number of latent factors in the matrix factorization of the ALS algorithm to predict the ratings for the given dataset. The matrix factorization is basically implemented to predict the ratings for the artists. Then the best accuracy model is chosen. That model is then chosen to make recommendations to the user, for a specific user in the list. The model recommends the top 5 artists for that user, but it could be any type of number of artists.

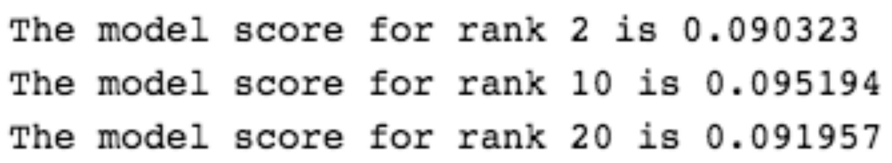
Evaluation Results:

When outputting the user play count and mean count statistics, this is an example screenshot of the output.



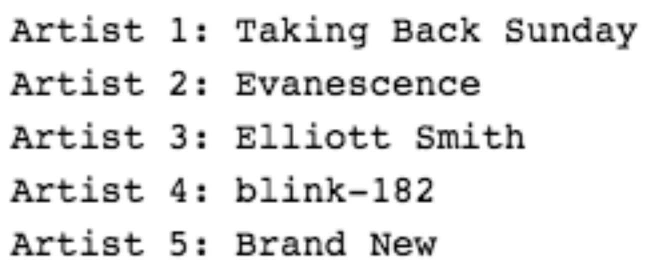
```
User 1059637 has a total play count of 674412 and a mean play count of 1878

User 2064012 has a total play count of 548427 and a mean play count of 9455

User 2069337 has a total play count of 393515 and a mean play count of 1519
```

When ranking the different models based on accuracy, the following output is provided.



```
The model score for rank 2 is 0.090323
The model score for rank 10 is 0.095194
The model score for rank 20 is 0.091957
```

When recommending the top 5 artists for user 1059637 in my dataset, the following output is provided.



```
Artist 1: Taking Back Sunday
Artist 2: Evanescence
Artist 3: Elliott Smith
Artist 4: blink-182
Artist 5: Brand New
```

Notes About Directory

I made a Jupyter notebook to follow along as well as a regular .py script. PySpark uses a lot of different functions and weird settings to run, so the notebook makes it easier to run. Datsets provided in the datasets directory.

References:

https://www.youtube.com/watch?v=XrpSRCwISdk

https://www.youtube.com/watch?v=wi_PPloqRe0&list=PLE50-dh6JzC5zo2whIGqJ02CIhP3ysQLX

https://spark.apache.org/docs/latest/rdd-programming-guide.html