Data Scalability Engineering Final Project Report

Statement of objective

Aim of the project is to use algorithms that help recommend users with similar movie preferences, movies that users might like and which matches their genre preference.. The application of these kind of algorithms can be seen across variety of industries such as Netflix, Amazon etc. These algorithms help drive some key decisions like which movie user is likely to watch next or in case of ecommerce industry it helps the company predict what products a user is likely to buy next. We will be analyzing the collaborative filtering algorithm which is used in most of the recommendation engines that helps drive these decisions.

Dataset details

1. Dataset is about the Netflix movies and its ratings given by the users. Data has been divided into three files namely TestingRatings, TrainingRatings and movies_titles..
2. 3.2 million samples are present in the training set and approximate 100,000 samples are present in the testing set.
3. We have movie id, user id and ratings as features of our training and testing data while our movie titles dataset has movie id, year of release and titles as its features.

Problem 2 – Analyzing the Netflix data

1. At first, the basic analysis is done on the dataset to check how many samples are there, do the dataset contains null values or any missing values.
2. As the entire project is based on user and movie related , distinct users and distinct movies details are captured by executing basic pandas and pyspark commands.
3. Loading the dataset to a spark dataframe using pyspark commands by reading it from the S3 amazon bucket. For better understandings spark dataframes have been transformed into pandas dataframes wherever it is needed.
4. Once the dataframe are created and knowing the distinct user and item details, now its time to dig deep into the rating aspects. Average Movie Ratings and Average User Ratings

were calculated. Below screenshots represents the user and movie lists with maximum counts in the testing set.

```
[ ]  averageMovieRatingsTesting.sort_values(by=['counts'],ascending=False).head(10)
```

movieId	ratings	counts
6971	4.123305	811
4640	4.006614	756
6287	3.716418	737
9728	3.807365	706
8915	4.007194	695
4432	3.674855	692
8596	4.091172	691
6408	3.876471	680
1406	3.233129	652
1744	3.752308	650

```
averageUserRatingsTesting.sort_values(by=['counts'],ascending=False).head(10)
```

userId	ratings	counts
1664010	4.300000	70
2439493	1.307692	52
305344	1.711538	52
387418	1.941176	51
1314869	3.263158	38
2118461	4.117647	34
1932594	2.321429	28
491531	2.629630	27
2606799	2.888889	27
727242	1.360000	25

5. The next step involves in finding the Estimated Average overlap of items for users. For this approach I have create a utility matrix by pivoting the table with index as userId column and columns as movieId giving the values as ratings from our training dataset.
6. Once the dataframe is created, many NAN values were found obvious as all users cannot rate all the movies and vice versa. Hence I have replaced all the NAN values with 0 for our analysis.
7. Next, I have selected the top 10 users from the testing dataset which had maximum count as the input and calculated the average ratings on each of these users. Finally using these average scores of 10 users, Estimated Average overlap of items for users has been calculated which is having an value of 2.55.
8. Now its time to find Estimated Average overlap of users for items. Same as above, I selected the top 10 movieIds from testing set which had maximum counts and calculated

the average rating score on each entity. Using these 10 average scores finally, Estimated Average overlap of users for items has been calculated with a score of 3.82.

9. Hence from the above analysis we can say that our user based analysis is good for the next analysis.

Collaborative filtering Implementation

The next step involves in implementing the algorithm on our large training data and check the accuracy of our model on the test data. The next section explain the algorithm approach.

Alternating Least Squares Algorithm

1. Importing the important libraries for the ALS algorithm to work would be our first step. Some of the libraries we will need as part of implementation are the regressor evaluator which will help us to measure the performance of our ALS model and import the ALS model itself.
2. ALS algorithm need some of the additional parameters to be fed like maxIter, regParam, ranks etc with specifying the actual inputs like userCol, itemCol and ratingsCol.I have considered coldStartStrategy which plays a vital role in eliminating the NAN values if the data have.
3. Fitting the model to our training data and predicting the ratings for our testing set is the basic approach.
4. I have considered two approaches by changing the parameters to check the model performance with root mean square error (RMSE) values and the mean square error (MAE) values. The approach in which the maxIter = 10 , regParam = 0.01 were selected, RMSE is about 0.84 and MAE with 0.7 were achieved which are less and the best comparatively.
5. Once the model is fit and predictions are done on testing data, its time to check the user and item predictions.

Own User Creation

In this section I have created my own user with userId as 0 and I have randomly selected the movie and I have rated the movies accordingly.

Creating a new dataframe for the new user details.

Next I put a join on my user dataframe with movie dataframe to get the movie names.

Combing the new user details with the actual training data.

Finally, I can see the movies which I rated in the actual data.

```
UserID = 0
self = df_training_union.filter(df_training_union.userId == UserID)
self.show(30)
```

```
+-------+------+-------+-------------+--------------------+
|movieId|userId|ratings|yearOfRelease|               title|
+-------+------+-------+-------------+--------------------+
|      8|     0|    4.0|         2004|What the #$*! Do ...|
|     10|     0|    4.0|         2001|             Fighter|
|     12|     0|    4.0|         1947|My Favorite Brunette|
|     24|     0|    3.0|         1981| My Bloody Valentine|
|     30|     0|    3.0|         2003|Something's Gotta...|
|     41|     0|    4.0|         2000|       Horror Vision|
|     43|     0|    4.0|         2000|      Silent Service|
|     48|     0|    4.0|         2001|      Justice League|
|     61|     0|    2.0|         1999|Ricky Martin: One...|
|     64|     0|    4.0|         2001|     Outside the Law|
+-------+------+-------+-------------+--------------------+
```

This is how we can recommend the movies to the users based on their ratings.