Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang

CSE 427 Final Project Report

MapReduce Approach to Collaborative Filtering for the Netflix Challenge

Shuang Wei, Junxu Chen, Nikki Xie, Yiheng Huang

December 14, 2018

Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang

I. Introduction and Motivation

Recommendation systems are vastly applied in the business and industries. Shopping sites recommend products to potential buyers; social media recommend people and topics; book rating forums recommend books to users with similar tastes. In our project, with Netflix data, we are seeking a strategy to utilize current ratings in movies to further recommend movies users might like. However, recommendation system faces obstacles in real world. New customers typically have extremely limited information, based on only a few purchases or product ratings. Many applications require the effective recommendation to be returned in real time. Especially for newer users, data is volatile and the algorithm must respond immediately to valuable new information.

We understand that recommendation systems like those above can create values for industries and enhance experiences for users. For this project, we are using collaborative filtering(CF) as our framework to tackle the problem of predicting ratings, which could be used as a direct measure for building a recommendation system. CF techniques require no domain knowledge and avoid the need for extensive data collection. In most cases, relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes.

To predict ratings for user-movie pairs in our target testing file, we are essentially filling in a user-movie matrix. Although we large set of training data, we should be able to predict all users-movie pairs. We have n users and m movies in our testing data. And, in this case, we only need ratings for specific user-movie pairs of the testing file(not the entire nxm matrix). Our entire project is implemented in Spark on AWS EMR platform. We use Pearson correlation coefficient to calculate all item-item similarity for specific users. For specific users, we generate top-N lists of similar rated items of a predicting items. This pool of rated top-N list is used as the averaging basis of calculation. We optimized our prediction by combining various sizes of N with different weight functions into various configurations.

Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang

II. Data Analysis

For this project, we are given two datasets, the training dataset has 1701 distinct items and 27555 distinct users. And the data is in the format: movie-id, user-id, rating-for-this movie.

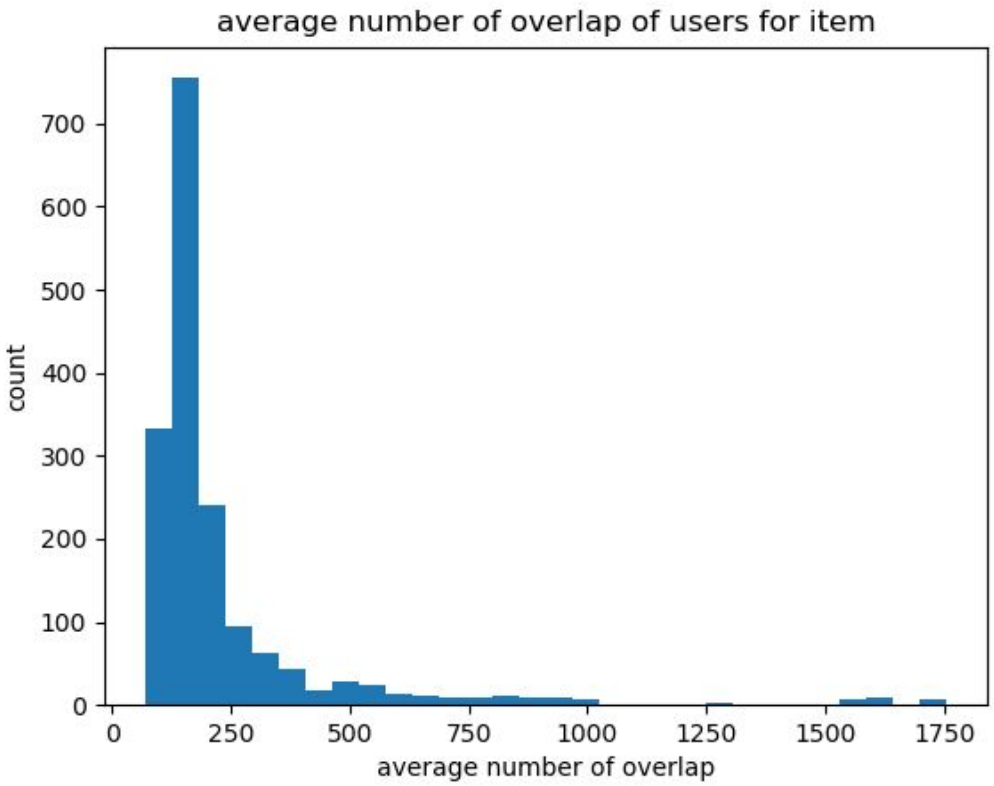
We use the training dataset which contains 3.25 million ratings (movie,user,rating) to train our algorithm and test sets which has 100,000 ratings to test our algorithm. We use spark through the whole project to process the data.
Because the amount of training data is so overwhelming and it has to be processed on cloud. To reduce the money we spent on cluster and also, improve the efficiency, we use the testing dataset as pseudo data to construct the basis of our algorithm. Not only the size of testing data is only about 3% of the training dataset but it also include all the items and movies that exist in training dataset.

Since the dataset has far more distinct users than items, we expect the average overlap of users to be higher. Users usually have more dimensions and features than items, so the prediction of user-based predictions will be more accurate. However, we decided to use item-based model because efficiency-wise, user-based method is worse because the queries of users is way larger than the pool for items. Therefore, it takes much more time to calculate the user similarity than item similarity. In another words, the number of reduce tasks and work to shuffle and sort for item-item similarity calculation will be smaller comparing to user-based method. Also, item-based recommendation is more efficient also because the item-item similarity can be calculated offline when we have streaming data. The movies are less dynamic and don't change as much as users' states, so we can calculate the item-item similarity table when offline, thus saves time and space for calculating the similarity and increase the efficiency. And the scalability of this approach makes it easy to recommend items to new users.

We plotted the average number of overlap of users for items and noticed that there's little to none items that have overlap less than 70. So we decided not to filter items that have low overlap(for example, lower than 50).

Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang



As a conclusion of our approach after data analysis, we decide to use pearson correlation to calculate the correlation between item-item pairs. The concept is following:

For a given item(item1), choose the users who have watched this item, extract items that a user have watched, make them a item-item pair RDD. So after we have a list of item1-item pairs that both these two items have watched by same users. After above steps, we are able to calculate the pearson correlation by using the formula.

III. Implementation

Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang

Our algorithm is implemented as follow. At first, we read in the training data and preprocessing this data by normalizing the rating. We normalize the data by subtracting the rating by the average rating of that item for each data point. Then, group the data by user, and generate all item pairs of all users. Using this data, we can calculate the pearson correlation and by calling the nlargest method, we can find the top N list for each item.

We then import the testing data, which contains the item-user pairs that we aim to predict, and the true rating for comparison. In order to predict the rating, we need to process all data necessary for prediction into a single RDD. By implementing consecutive map and join steps, a final RDD used for prediction, which has item-user pair as key and values containing Top N list, list of movies rated by this user and average rating for this item, is created. With this RDD available, we call a single predict function to predict the result. Finally, we compare our prediction to the testing data and calculate mean absolute value and root mean squared error

separately.

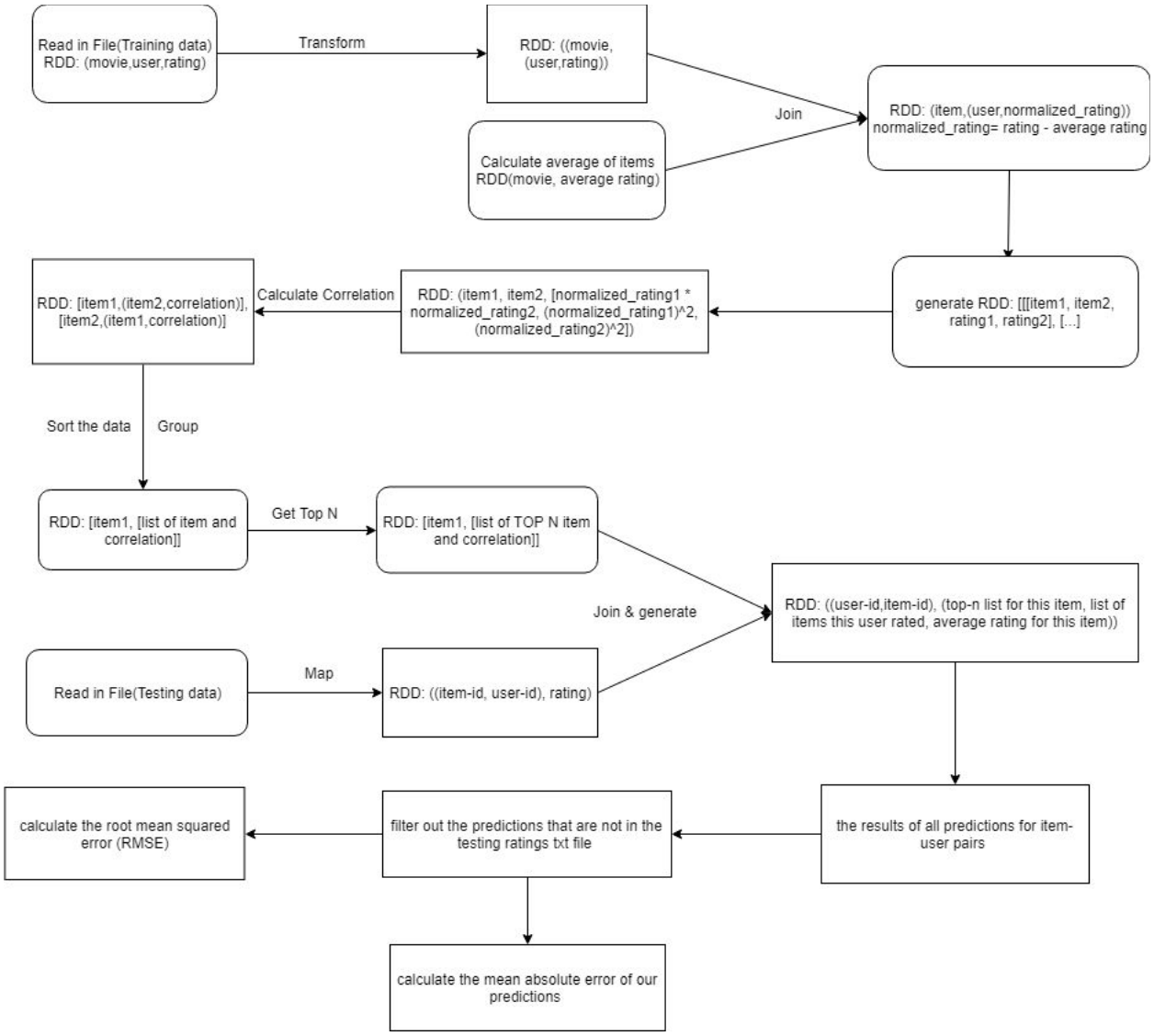


Fig 2. DAG of our algorithm

IV. Results and Discussion

A. Pseudo cluster

As we using testing dataset to construct our algorithm, so what we do in the pseudo cluster is to use testing dataset to predict itself. The baseline analysis result, which means we predict directly using the average rating of that item, the MSE is around 0.79 and variance is around 1.03. For the collaborative filtering, we use pearson correlation to take top 20 most similar item, and the MSE is about 0.803 and variance is 1.002.

B. Real cluster

1. The performance and efficiency of our algorithm is evaluated based on the accuracy of prediction, the unpredicted ratings, the time consumption of each run and the memory cost of the overall program. Specifically, we base the accuracy of our prediction on the root mean square error(RMSE) and the mean absolute error(MAE).
   We notice that K, which is the factor that decides how much items we include in our top-K list, and N, which is the factor that decides how much ratings we iterate in our prediction model, will have relatively significant impact on the resulting RMSE and MAE. So we manipulate our K in all of our prediction models.

2. Baseline analysis:

First of all, with the training set, we set a baseline of RMSE and MAE with the RMSE and MAE derived when predicting directly with the average rating, without any weight or collaborative filtering, for each movie. Under this situation, we had an MAE of 0.797553989147 and a RMSE of 0.994343045628 when K = 10.

Then We manipulate K to be several data points from the range of 1 to 200 to see the influence of K on the error stats. We had the following result as in Figure 2.

We noticed that except when K is below a threshold of around 7, increasing K doesn't improve our prediction accuracy. The MAE and RMSE are mostly constant after the threshold.

3. Model comparison:

We decided to compare 3 models with different configurations:

1) Top K similar movies to movie1 with cosine similarity measure
2) Top K similar movies to movie1 with pearson correlation similarity measure
3) Top K similar movies with threshold on the number of rated movies to be considered, or the threshold on the correlation

For all the models above, we factor in the 4 measures of evaluating our algorithm discussed in above.

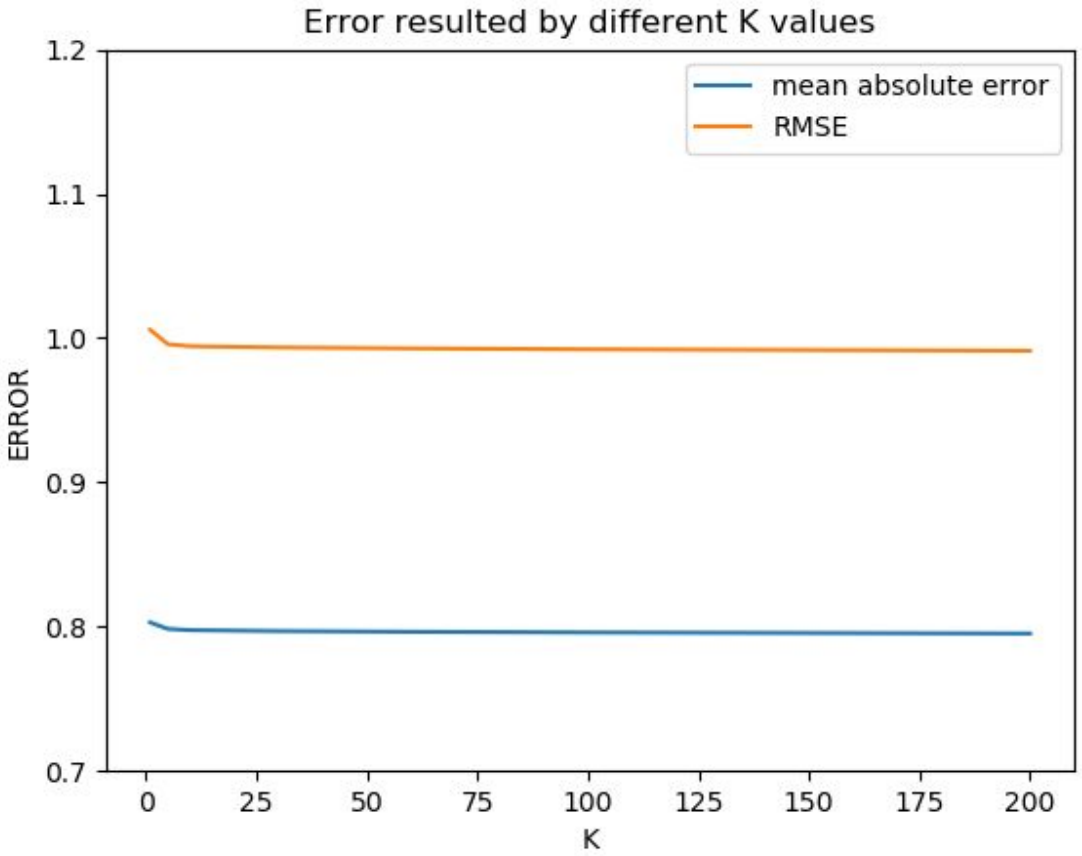


Fig 3. Error for different K values

Model 1: Top K similar item list with Cosine Similarity

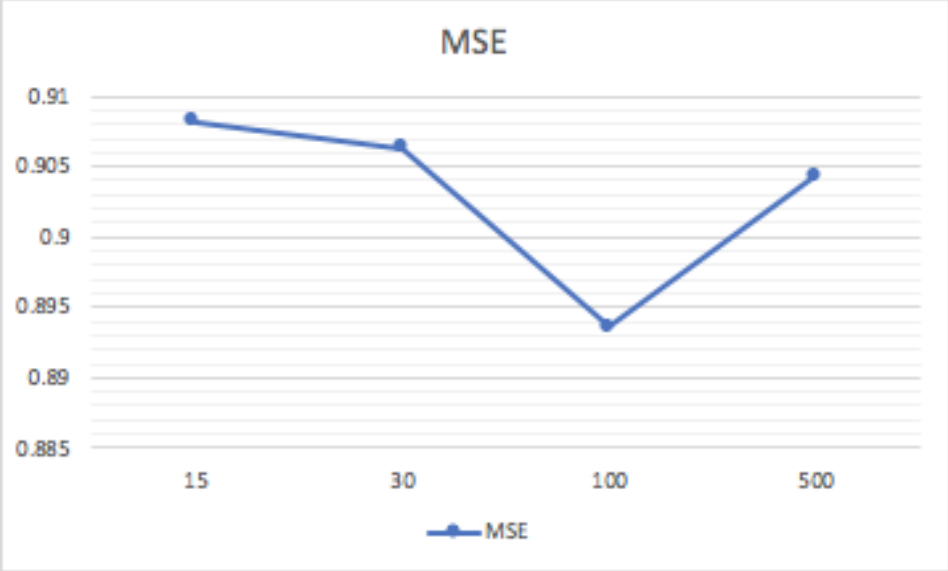


Fig 4. MAE of model 1

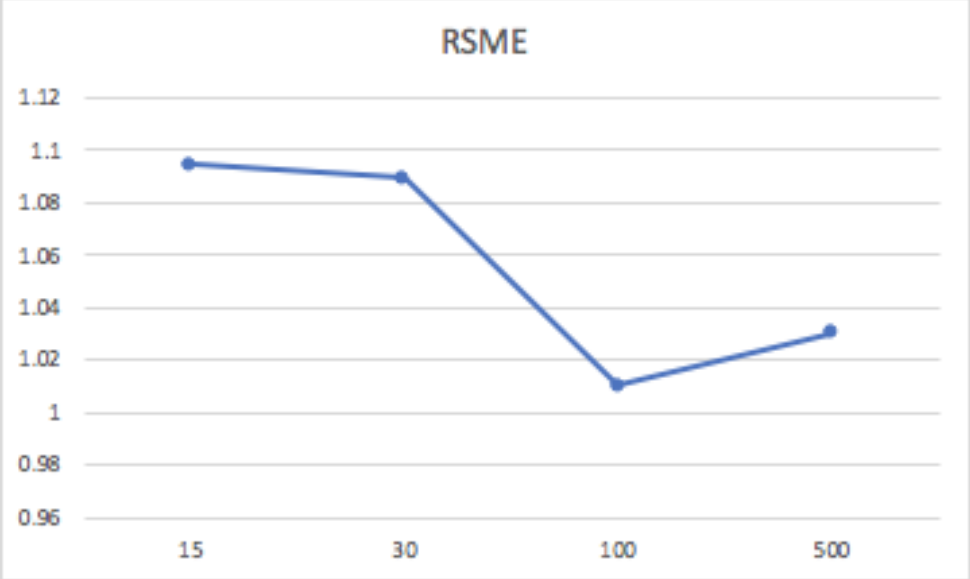


Fig 5. RMSE of model 1

Based on cosine similarity, we generate the TOP N list for a given item and make the prediction. The above graph shows the MSE and RMSE changes by implementing different k.

Model 2: top K similar items with pearson correlation

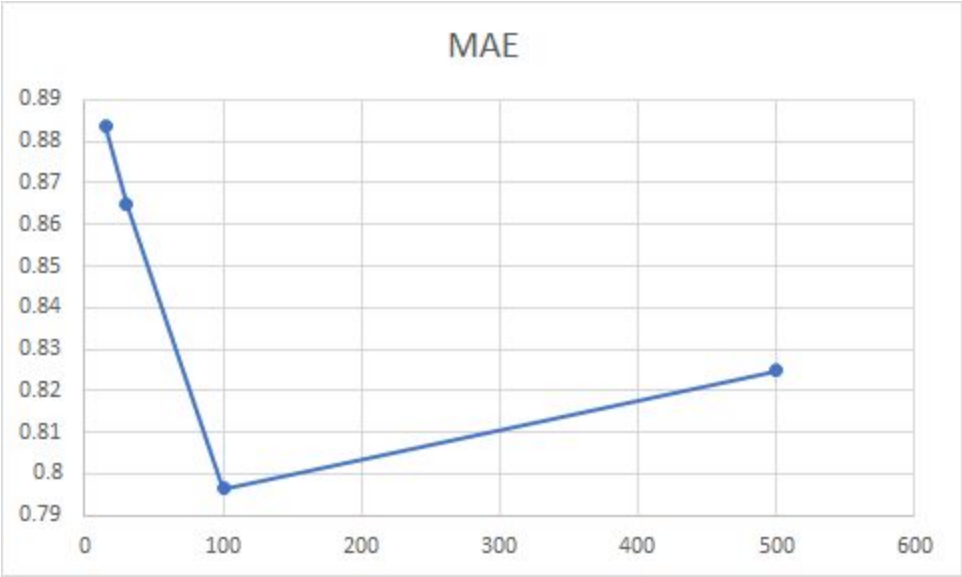


Fig 6. MAE of model 2

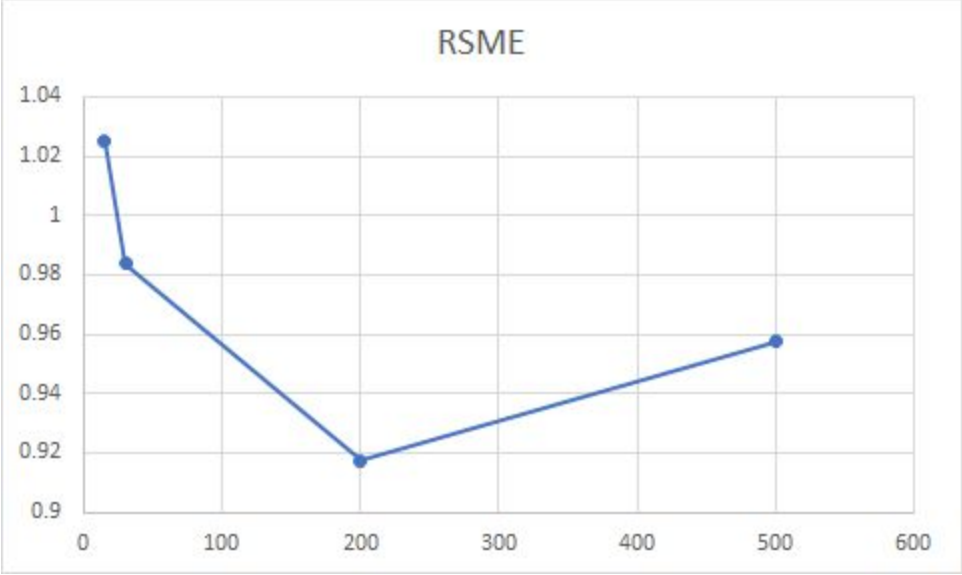


Fig 7. RMSE of model 2

The pearson correlation performs better than cosine similarity but slightly change the result.

Model 3: pearson correlation and threshold limitation(Best)

We now consider adding a restriction based on model 2: pearson correlation. We filter out correlated items with a pearson correlation less than 0.75. And then we found 20 items in the top N list that this users has rated. Note that we test many different values and found that 0.75 and 20 are the optimized combination. The best result we achieved by implementing such restriction is MAE of 0.764 and RMSE of 0.954.

In conclusion:

Approach	MAE	RMSE
Model 1: Top K list with cosine similarity measure	0.8936	1.0102
Model 2: Top K list with pearson similarity measure	0.796	0.919
Model 3: Top K list with pearson and threshold limitation on correlation and rated items	**0.764**	**0.954**

Thus, our best result is with model 3, when MAE is 0.764 and RMSE is 0.954.

V. Conclusion/lesson learned/future

In this project, we constructed a system that for a given user and movie we would be able to predict the rating that this user most likely would give by using collaborative filtering and average rating prediction. We started with using spark to analysis the data and decided to use item-item based instead of user-user based measurement. Then we construct the whole code outline by using the pseudo cluster and execute it to the real cluster on Amazon EMR service. Finally we improved our code and algorithm by bounding the TOP N list which filters the pearson correlation that lower than x(x can be either bound we set, range (0,1)).

There is still some further consideration about the predicting algorithm. By observing the top N list generated from the algorithm, we found that many item-item pairs have a pearson correlation of value 1.0. The reason might be that there is only 1 user or very few user have rated

both items. Thus, the resulted pearson correlation might be biased. We can improve the algorithm by filtering out item-item pairs which has a co-occurrence below certain threshold.

VI. References

Koren, Yehuda. "Factorization Meets the Neighborhood." *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 08*, 2008, doi:10.1145/1401890.1401944.

Linden, Greg. "Amazon.com Recommendations ." *Www.cs.umd.edu/*, 2003, www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf.

Junxu Chen, ID: 467525 Shuang Wei, ID: 467487 Nikki Xie, ID: 448385

Yiheng Huang, ID: 452257 Repository: yihenghuang

Appendix A: Group Role Specification

Name	Role	Tasks
Shuang Wei	**Developer Cloud**	**-testing pseudo cluster -cloud execution -documentation of results**
Junxu Chen	**Developer Local**	**-implementation -testing locally -testing pseudo-cluster**
Yiheng Huang	**Project Manager**	**-communication/ coordinate among all team members -manages submission to SVN repository -write up**
Nikki Xie	**Key User**	**-data processing -find and process Big Data application/real-world data -write-up**