Netflix Final Report

Rebecca Buerger, Jack Ewert, Nicholas Litterio, Waddah Moghram, Dhruv Vyas, Yuchen Zhang
The University of Iowa

Contents

Abstract

Over the last decade, Netflix has cemented its position as a leading streaming media provider. To maintain its dominance, Netflix has commissioned a one-million-dollar prize in 2009 for the code that improved rating predictions for previously collected real-life customer. Our team has been tasked with the same task for two months. As of the end of the first month, we successfully read and visualized the original dataset provided by Netflix and identified strategies to proceed with the project. Some of these strategies included K-means clustering and Pearsons' R correlation. Approaching the conclusion of our project, we were able to supplement about 60% of the movie titles with IMDB online database. In addition, we included some time-series analysis of movie and user trends. This paper has been submitted as part of a class entitled Big Data Analytics (IE:4172) on December 7, 2018.

1. Introduction

The problem that we have tried to solve is predicting how any given user would rate any given show / movie based on how they rated similar shows / movies in the past. In mathematical terms, for a given user (u_i) and a given movie (m_i) at a given timepoint in the past (t), and ratings $x_i(u_i, m_i, t)$, we aim to predict the future rating, $y_i(u_i, m_i, t, x_i(u_i, m_i, t)$.

However, achieving such a task is difficult for various reasons. First, the sheer number of data points is one of the main obstacles. Second, each movie has many different attributes such as title, main actors/actresses, director, genres, etc. – all of which are expected to influence a user's choice. Similarly, users are unique and as such have many "features" that define them such as race, age, location, and so on. In mathematical terms, that means any user or movie is a function of many unknown parameters, or $u_i(race, age, location, \dots)$ and $m_i(genre, year, actors, \dots)$ As such, solving such a problem is conceptually difficult even if all the features are known and given that most of these variables are categorical in nature.

It is important for big companies like Netflix to do this so that they can accurately predict what movies a user would like to watch in the future. This would help them choose what movies to add to their platforms and allow them to recommend movies to users that they would like to watch. A competitive advantage would be given to companies that can accurately predict their users' preferences as they would be more in tune with their customers.

Another lesser challenge arose from the fact that the data were not organized well, therefore, the file needed to be cleaned up before data analysis. Lastly, for variables such as user ratings, human variability is a huge part of the prediction. There are many more things that can factor into it than just relevancy to other movies. For example, let us say one user rates documentaries relatively higher compared to other types of movies, but really hates the Minnesota Vikings. If this user were to watch a documentary on the Minnesota Vikings, then they would rate it low and this should not be a suggestion for them to watch. There is no way of knowing this beforehand but

could play a huge role in how they would rate the movie.

2. Problem Statement

The dataset consists of 100,480,507 five-star ratings that 480,189 anonymous Netflix users gave for 17,770 movies. This data was collected by Netflix over a span of seven years. Also, included was a testing (or probe) dataset, which is a subset of training dataset, comprising of 1,408,395 ratings. The goal of the project is to predict unavailable user ratings, which account for 98.9% of the ratings-movie matrix.

We decided to approach this problem by dividing it into two main tasks. First, we pruned the test dataset to reduce the number of features and to investigate various machine learning techniques to minimize prediction error (e.g., RMSES in our case). Once we have finalized this step, the second task is to use a qualifying dataset. This qualifying dataset is to be used for submitting actual prediction, which would be used later by Netflix (by generating 50% random quiz-set from a qualifying dataset) to judge the effectiveness of our prediction algorithm. This last step was skipped in this project as we would not benefit from it in the machine learning process. As such, only the probe dataset is used to accuracy of the model.

3. Preliminary Data Analysis

One of the first preliminary analyses was to visualize the rating distribution of our dataset to uncover any potential trends. In Figure 1, the user-rating distribution appears to be skewed to the left for users with most users given a rating of 3 or higher. This can be used to normalize the results to gain better prediction accuracy and to study any underlying trends.

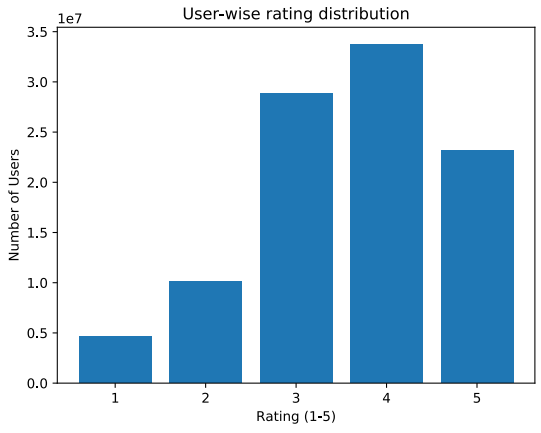

Figure 1 Rating distribution for all users

Next, we analyzed the ratings trends over the weekends compared to weekdays. As shown in Figure 2 and Figure 3, there was no significant difference except for the weekend average ratings being slightly higher compared to weekdays. Thus, we can potentially use this insight to decide whether to weigh weekend and weekdays prediction differently. In addition, we also plotted the daily ratings distribution. However, we could not deduce any useful trends. Hence, it is not included in the report, but can be viewed under plots directory of code.

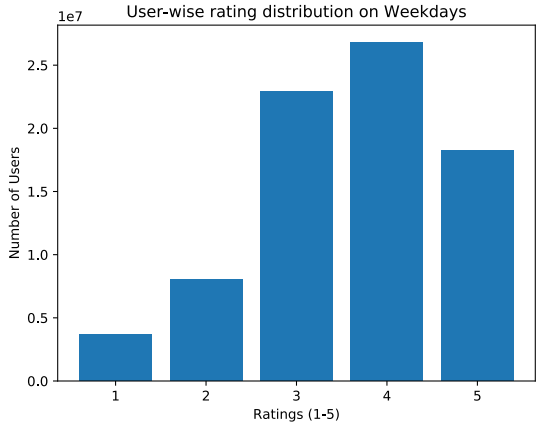

Figure 2 Ratings distribution for all users during weekdays

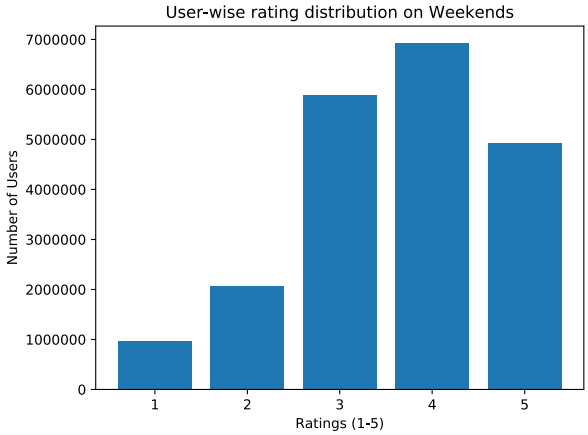

Figure 3 Rating distribution for all users over the weekends

Then, we plotted the historical trend (or time-series) of the ratings for a few movies and users, as seen Figure 4 and Figure 5, respectively. These plots suggested that certain movie or user rating trends and frequency could increase or decrease with time. For instance, we can potentially add weights, or a feature based on a correlation for certain times of the year (e.g., weekends vs. weekdays, holidays vs. working day, etc.).

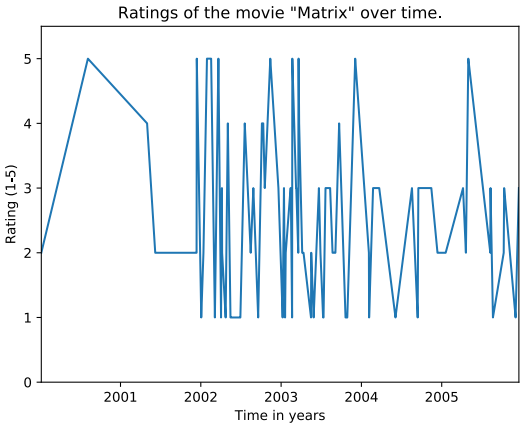

Figure 4 Historical trend of the ratings of a randomly chosen movie (e.g., Matrix in this case) revealing an increased interest between the years 2002 and 2005.

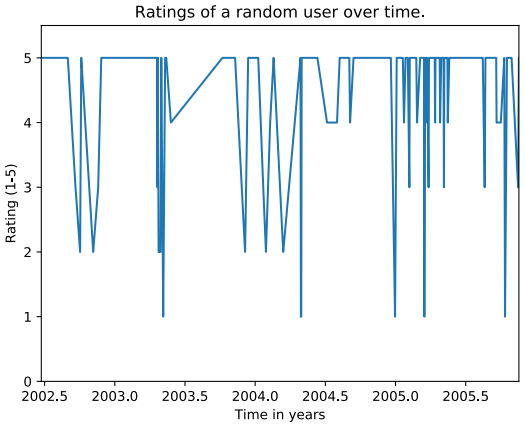

Figure 5 Historical trend of the ratings of a randomly selected user, revealing this user does not like to give a rating of 1.

Next, we evaluated and plotted some basic measures of central tendency for movie-wise and user-wise ratings as shown in Figure 6 and Figure 7, respectively. Some of these measures were the minimum, mean, and maximum ratings. Generally speaking, we could not notice any particular extreme trends in the movie ratings, which can range between 1.5 and 4.5, or centrally distributed on either side, while it is not uncommon to have maximum rating of 4 for some movies and minimum rating of 2 for some movies. As for users that do rate, it seemed that they were more likely to rate in the higher range than in the lower as seen in Figure 7. It can also be seen that the lowest rating a lot of users gave was 2 out of 5.

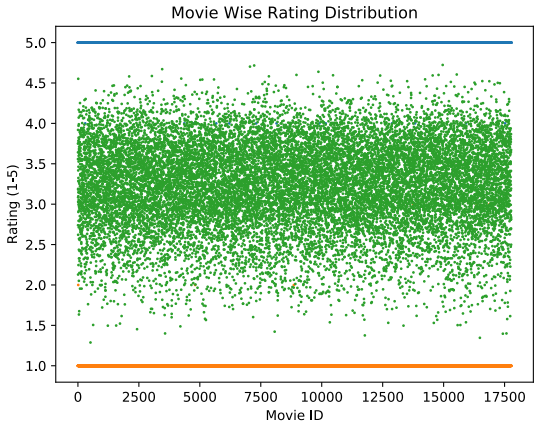

Figure 6: The central tendency for the ratings of all movies, where orange is the minimum rating for a given movie ID, green is the mean rating, and blue is the maximum rating.

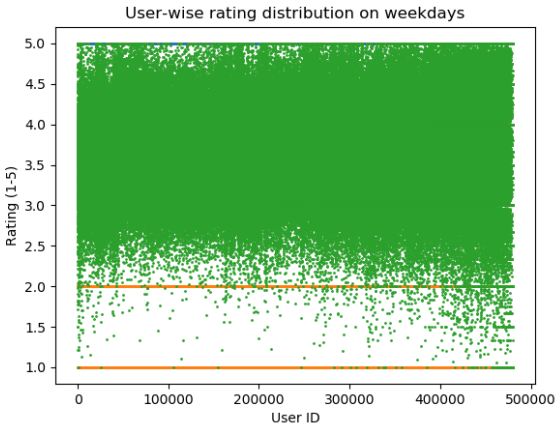

Figure 7 The central tendency for the ratings of all users, where orange is the minimum rating for a given movie ID, green is the mean rating, and blue is the maximum rating.

4. Training and Validation

4.1. Data Processing and Data Enrichment

Since the given dataset is organized into four separate large files, we decided to compile then divide the entire dataset into two files that would simplify further analysis and processing. To do so, we mined the original data files to create a movie-wise dictionary data file and another user-wise dictionary file. The format of these files are as follows:

```
<movie_ID1/user_id1>:<user_id1/movie_id1>|
<date>|<ratings>…,<user_idn/movie_idn>|
<date>|<ratings>
```

Next, given that the original dataset only contains movie IDs and the ratings given by users and the date of each ratings date, we felt that there is not an adequate number of features to use from the original dataset for machine learning techniques. Therefore, we decided to enrich that dataset with IMDB movie dataset to obtain additional movie features. Some additional features include attributes such as genres, runtime, cast, year released, director, movie ratings (e.g., G, PG, R, etc.). After having discussed a plausible number of attributes, we ended up using movie genre information to begin our analysis because we believed it would have the biggest impact on our model's ability to correctly predict user ratings on a given movie. For future work, we would explore which – if any – other IMDB features truly impact our model the most.

Our final dataset for clustering movies contained the movie IDs and 25 genres, runtime and release year for those movies. As a side note, a movie can have one or more genres. Similarly, we built a dataset for clustering users that includes 25 genres and the ratings given by those users for each genre. In other words, 5 ratings for each of the 25 genres for a total of 125 features. Thus, user features are 150 for each.

4.2. Clustering

When predicting the rating for a particular user/movie combination, our main assumption is that similar users will choose similar movies and similar movies were rated the same by similar users in the past. In other words, by clustering movies on one side and users on the other side, we can create a one-to-one mapping between user-to-movie choices.

To obtain these user and movie clusters, we used k-means as one method of unsupervised learning since we do not know how they are clustered in actuality, but rather we would like to infer these clusters from our presumed features, which were detailed in the previous section. We started with 100 clusters for users similar to what is shown in Figure 8, and 10 clusters for movies (or $n_{C,m} = 10$) as shown in Figure 9. Again, this is not cast in stone and further improvement can be made by iterating through different n_c values to obtain the ideal cluster size for this problem.

4.3. Data Reduction

Given the massive number of users and ratings, it was essential to conduct some data reduction for our code to run. On the other hand, there are only about 17,700 movies, (or $n_m \cong 17{,}700$), which is of a more manageable size when it comes to feature addition and movie-based clustering. Nevertheless, for each of those movies, there were a lot of ratings and lot of users. This means to deal with the user ID-based data, we would be required to reduce the amount of data so that the computation time and memory usage become manageable.

To reduce the data, one proposed method is select for users and movies with at least 30 ratings. This way, we remove all the data that cannot be used to obtain useful trends and might even become outliers themselves if they contain extreme values. This seemed to be an efficient way to cut down some of the data.

Another method we planned on using was to splice every 100 to 1000 users a couple times to see which worked best. This would bring down the 100 million instances to 1 million or 100,000. This helped speed up and shrink the memory usage of the calculations and clustering by making it. However, the ultimate implementation of this step is to be seen.

4.4. Prediction Techniques and Parallel Processing

First, we started with simple naïve approaches such as using movie averages, user averages and using weight schemes to include both user and movie to predict the movie rating, as will be explained in more details in the following section. We have initiated an asynchronous parallel programming scheme that would take advantage of all cores using python's multiprocessing library. More precisely, we have used CPU parallel processing without any GPU acceleration, and it seemed to work, albeit we did not have enough time to integrate this as part of our entire program. Our proposed algorithm to use parallel processing to identify the actual cluster sizes that were referred to previously, and to iterate through the entire search space to find the optimum cluster sizes. The optimum size will be found by calculate the RMSE error of the testing dataset as explained in detail next.

4.5. RMSE

In order to quantify the accuracy of our model, we evaluated the root mean square error (RMSE) between the predicted ratings by our model, $y(m_i, u_j)$, and the known ratings as provided by Netflix for the users listed in the probe dataset, $x(m_i, u_j)$, for a given movie, m_i, and user ID, u_j. Our criterion for improving our model and for optimizing the parameters of those models is by minimizing the RMSE. So far, we have minimized the RMSE by choosing the lowest value by varying the weights α and β in a linear combination of the move-wise predicted ratings $y_m(m_i, u_j)$ and used-wise predicted rating $y_u(m_i, u_j)$, or:

$$y(m_i, u_j) = \alpha y_m(m_i, u_j) + \beta y_u(m_i, u_j)$$

For now, we estimated the predicted movie- and user-wise ratings as the arithmetic mean, or average of the

ratings $\overline{y_{m_k}}$ of the k[th] movie cluster (m_k), and the average of the ratings $\overline{y_{u_l}}$ of the l[th] user cluster (u_l), respectively. In other words,

$$y_m(m_i, u_j) = \overline{y_{m_k}}(m_i, u_j) = \frac{1}{n_{r,m_k}} \sum_{a=1}^{n_{m,m_k}} \sum_{b=1}^{n_{u,m_a}} x(m_a, u_b)$$

and

$$y_u(m_i, u_j) = \overline{y_{u_l}}(m_i, u_j) = \frac{1}{n_{r,u_l}} \sum_{a=1}^{n_{m,u_l}} \sum_{b=1}^{n_{u,m_a}} x(m_a, u_b)$$

where n_{m,m_k} is the number of movies in the k[th] movie cluster, n_{m,u_l} the number of movies in the l[st] user cluster, n_{u,m_a} the number of users that rated a particular movie (m_a), n_{r,m_k} the number of ratings in the k[th] movie cluster, and n_{r,u_l} the number of ratings in the k[th] movie cluster; such that,

$n_{r,m_k} = \sum_{a=1}^{n_{m,m_k}} \sum_{b=1}^{n_{u,m_a}} 1$, and $n_{r,u_l} = \sum_{a=1}^{n_{m,u_l}} \sum_{b=1}^{n_{u,m_a}} 1$,

The ultimate RMSE obtained thus is what we refer to as the "naïve approach" and is evaluated by

$$RMSE = \sqrt{\frac{1}{n_m}\left(y(m_i, u_j) - x(m_i, u_j)\right)^2}$$

where n_m is the total number of movies, or

$$n_m = \sum_{k=1}^{n_{C,m}} n_{m,m_k}$$

Granted that there are other heuristics and algorithms that can arrive at the optimum more quickly and efficiently, such as gradient descent to optimize $y(m_i, u_j, \alpha, \beta)$ for a given (m_i, u_j) data point. However, that was not necessary for the naïve approach, although it might be more necessary for more advanced machine learning techniques.

For our hybrid approach, we use following formula in order to combine naïve approaches and clusters. We use values of α and β as optimized for our weighted naïve approach.

$$y(m_i, u_j) = \alpha\left[\bar{x}_m(m_i) + \frac{1}{2}\left((\bar{x}_m(m_i) - \overline{y_{m_k}}(m_i, u_j)\right)\right] + \beta\left[\bar{x}_u(u_j) + \frac{1}{2}\left((\bar{x}_u(u_j) - \overline{y_{u_l}}(m_i, u_j)\right)\right]$$

, where $\bar{x}_m(m_i)$ is the average rating for a given movie m_i belonging to the k[th] movie cluster, and $\bar{x}_u(u_j)$ is the average rating that a given user u_j that belong to the l[th] user cluster. Notice that the weighing parameters α and β are different in this case.

Lastly, we also experimented with a similar formula that was proposed by Hong and Tsamis as shown in section 7 of reference [4]. However, it did not result in any improvements over what we already have. In other words, the RMSE was about the same.

5. Visualization and Results

In order to validate our clustering scheme for users and for movies, we decided to plot them in 2D using Principal Component Analysis (PCA) to reduce the number of attributes to two for the purpose of producing a comprehensive visualization of movie and user cluster. This was done using Python Scikit-learn library and are shown in Figure 8 and Figure 9.

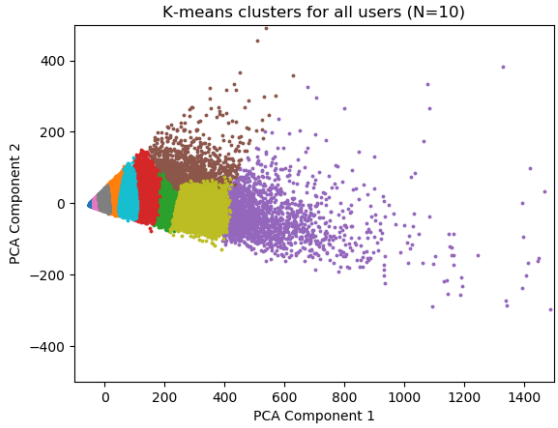

Figure 8 Netflix users classified into 10 clusters, using k-means clustering. The user distribution is plotted against two principle component analysis (PCA) components vectors, which are a combination of all 150 user features. 10 clusters are used for better visualization

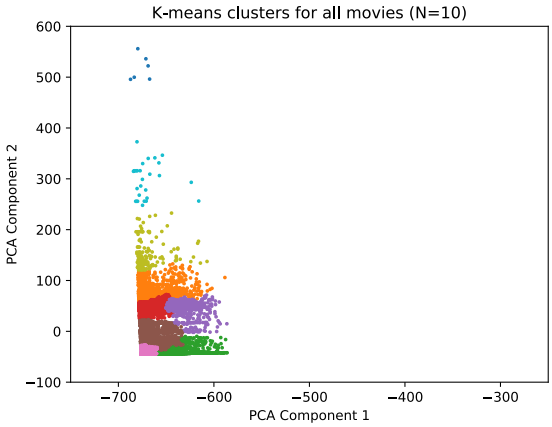

Figure 9: Netflix movies classified into 10 clusters, using k-means clustering. The movie distribution is plotted against two principle component analysis (PCA) components vectors, which are a combination of all 25 movie features. One outlier point at PCA 1 was at a value of approximately 1250.

It worth noting that movie-clusters in Figure 9 are not centered around the axis of PCA Component 1 because of one cluster that had PCA1 value in excess of 1200, which counterbalanced all other clustered. This seems to highly suggest that one of the features of movies was left out, or some kind of feature that very strongly clustered those movies against all other movies.

Lastly, we clustered the movies using three of the parameters of central tendency as features, namely minimum, maximum, and average rating in addition to the total number of ratings a movie received, as shown in Figure 10. In this case, the x-axis represents the average ratings, while the y-axis represents the total number of ratings for each movie.

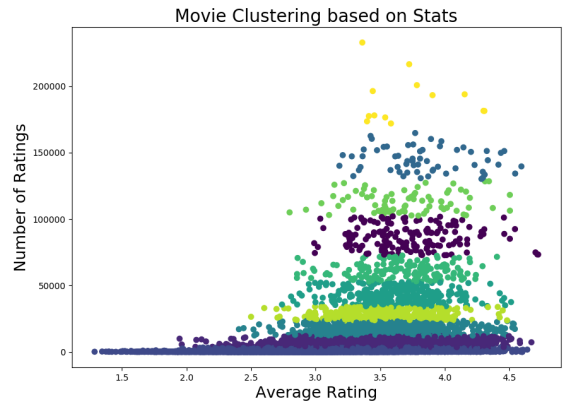

Figure 10: Netflix movies classified into 10 clusters using four features, namely the total number of rating, and a movie's minimum, maximum and average rating.

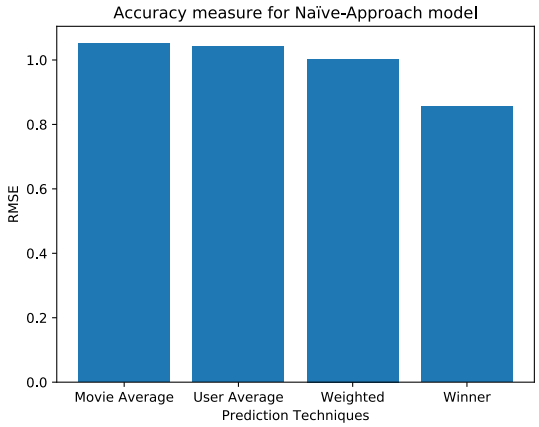

Figure 11: A comparison of the RMSE achieved by our naïve-approach predictions (first three columns) compared to that of the winner team.

The other type of visualization is a comparison of the RMSE for the various methods as previously explained in section 4.5. Figure 11 compares the RMSE results of various Naïve approach techniques, starting with movie-wise average rating alone, then user-wise average alone, and then the weight naïve-approach predicted rating. The last column is the RMSE of the winning team at 0.8554 [5]. As can be seen, predictions based on movie-wise average rating performed similarly to predicting based on user average rating. However, we were able to reduce the RMSE even more using a weighted average. These weights were We also compared our results to the RMSE the winner of the Netflix prize was able to produce, which was about 0.8554 as compared to our 1.003 of using weighted.

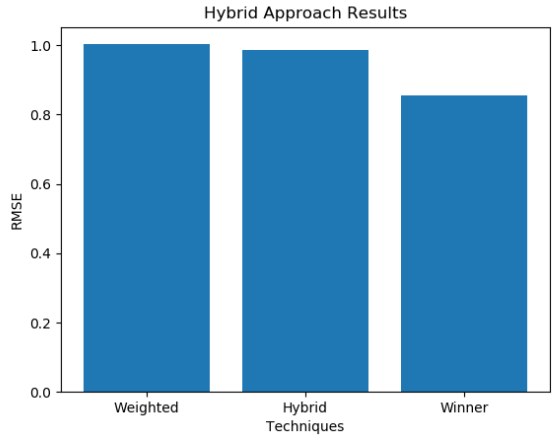

Figure 12 A comparison of the RMSE achieved by our weighted naïve-approach prediction, our hybrid technique compared to the winning RMSE.

While Figure 12 compares the improvement of our naïve approach with a hybrid approach that considers individual

movie ratings, user ratings as well as clustered user and movie average ratings. This hybrid approach reduces the RMSE even further to 0.98. Lastly, we experimented with the number of clusters and found that setting the number of movie clusters ($n_{c,m}$) to 100 and the number of user clusters ($n_{c,u}$) to 1000 as the best parameters.

6. Discussions

In conclusion, we found that the movie-based clustering and analysis were easier to deal with than user-based ones. This is simply because movies had 25 features and 17770 data points. On the other hand, users had 150 features and 480,000 data points. Therefore, data and feature reduction techniques were necessary as well as other big data techniques in this case.

Another challenge for this project was dealing with the sparse matrix. We used two methods to deal with that. First, we discarded movies and users with less than 30 ratings. Second, we filled in the gaps with average ratings for the users and movies on the cluster.

Also, since we supplemented movie data to our project with the IMDB data, our project became more complicated. Not only did It add more features to the movie IDs, but it also added more features to user IDs because the features to these two are related.

Overall, we feel confident that had we had more time, we would have been able to reduce the RMSE closer to that of the winner algorithm using time-series data. We realize that this is a challenging project since it took three years for a team of three professional groups to win the prize and bring the RMSE down to 0.8554.

7. Future Work

There are few things that come to mind how to improve our predictions. First of all, we can make use of time-series to study trends of user-based ratings, and movie-based ratings. These might uncover weekly, seasonal or yearly trends. Figure 4 and Figure 5 are some examples that show that. In addition, we can make use of statistical variability to normalize those trends for individual movies and users when extrapolating the predictions from their respective clusters. Another example would be giving season-related movies. For example, during Halloween time, horror movies seem to be a popular choice while during Christmas time, comedy/holiday movies seem to be in demand at least by intuition, and we might even find the trend to prove that.

Second, we can enrich the rest of the movies that did not find a match in the IMDB database using other database or using other APIs that can crawl Wikipedia or the internet.

Third, we can choose the right model for each movie cluster and for each user cluster instead of using a one-size-fits-all models. Even more than that, we can let the computer choose the right model for each individual movie or user.

Fourth, to deal with the sparsity problem, we supplemented our data with IMDB dataset features that are rich in nature. In other words, there is a lot of information available in order to cluster the data, organized data but a lot of these features are not simple (i.e., Boolean or float) and hence it might be a challenge to breakdown these features into simple categories for example a particular cast is known for a class / genres of movies but he/she might not be very popular choice for other class of shows / movies. If we had time, we would have tried extracting extra features out of IMDB dataset in order to prune clustering and hence results.

Fifth, we can add more features to the movies, and subsequently to the users, such as movie ratings, names of actors and directors, and so forth, to see if that will improve our results.

Lastly, we are confident that we can reduce RMSE error further by implementing all of the above ideas. We can use more complicated models such as those listed in the top-three winner algorithms as well [1,2,3]. It is worth noting that those models did not supplement the original data with anything else. Therefore, it is not inconceivable that we can even exceed the results of the winning team – theoretically!

8. Appendix

8.1. Code Repository

The source code for these programs are publicly available on UIowa GitHub through this URL: <https://github.uiowa.edu/wmoghram/BigDataNetflixProject2018>

Please note that some data files could not be uploaded to GitHub due to the size limit of 250 MB allowed by server. However, these files are available by request and the result-producing code can be obtained by running the existing data files and source files if that the needed python libraries are installed properly.

9. References

[1] The BellKor Solution to the Netflix Grand Prize : https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

[2] The BigChaos Solution to the Net ix Grand Prize : https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

[3] The Pragmatic Theory solution to the Netflix Grand Prize : https://www.netflixprize.com/assets/GrandPrize2009_BPC_PragmaticTheory.pdf

[4] Use of kNN for Netflix Prize : http://cs229.stanford.edu/proj2006/HongTsamis-KNNForNetflix.pdf
[5] https://techcrunch.com/2009/07/26/the-netflix-prize-comes-to-a-buzzer-beater-nailbiting-finish/