1. Introduction:

1.1 Task:

The goal of this project is to predict movie rating for the given customer. In particular, for a given user and a movie not in the data, predict what the rating would be, that is, predict how the given user would rate the given movie. Also given a user and the movies they like what movies can be recommended.

1.2 Background.

The Netflix prize was a competition to find best algorithm to predict user ratings for the films based on previous ratings. Data was about 100 million rating from 1 to 5. The evaluation metric used was root mean squared error(RMSE) . At the start of the competition the RMSE the Netflix had with their Cinematch algorithm was 0.9525. The expected result to win the prize was improvement of RMSE by 10%.

1.2 Data

Netflix provided 100 million ratings from (1 to 5) of 17K movies and 500K Netflix customers. This is in the form of (User_ID, Movie_ID, Rating). The data of each rating and the title and year of release for each movie ID is also provided.

2. Implementation

The main step in implementing a recommendation is preprocessing. The given data is very large and will not fit memory for most of the computers. The main challenge that I faced was to fit all the data into memory for analysis.

2.1 Data processing

There are two main files in the dataset:

a. User Data(which is further divided into four parts)
b. Movie Titles

User data consists of User, Rating and Date separated by Movie_id. The user-data structure must be preprocessed to extract all ratings and form a matrix, since the file-structure is a messy mixture between movie_id , user_data.

Movie Titles consists of Movie_id, Year and Name.

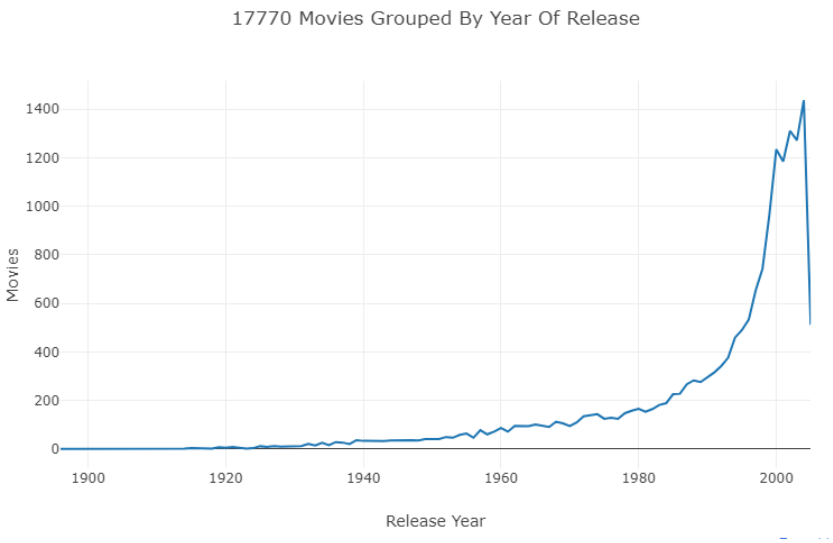
We combine these files to get the following table. All the null values were removed before combining the data.

User	Rating	Date	Movie
258937	3.0	2005-01-20	4123
523957	3.0	2004-05-10	482
2240135	4.0	2005-08-07	518
1691967	3.0	2004-08-03	416

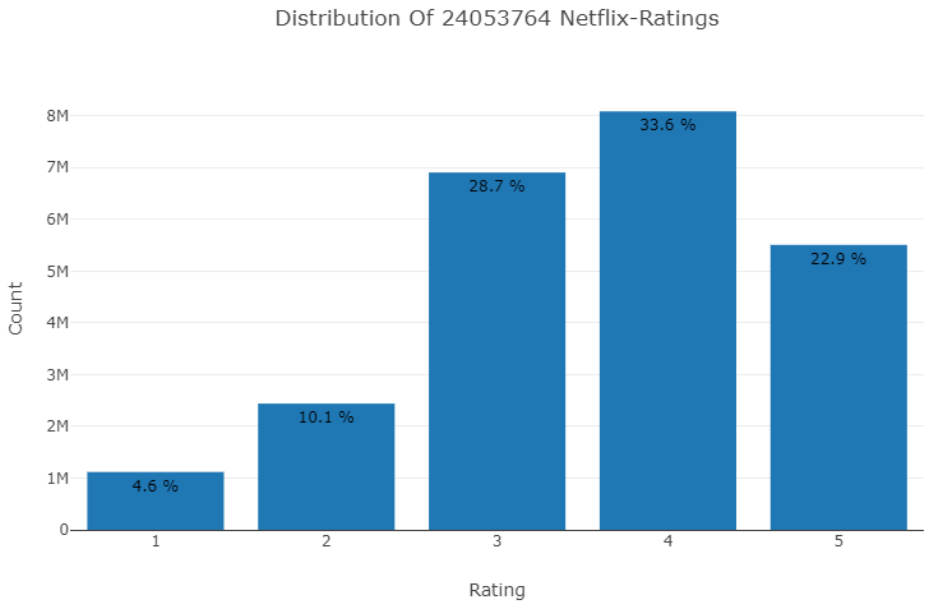
2.2 Data Visualization

The following are the facts for the dataset.

Most of the movies in the dataset are released after 1995. This shows that using Date as a factor cannot give a good recommendation for movies.



Most of the user give a rating of three or a four. This shows a distribution bias in rating. It can happen because people who do not like the movie usually do not rate it.



These factors can be used for cleaning the data.

2.3. Data cleaning

We can follow two approached for data cleaning:

a. Remove users with reviews less the 200 and remove movies with reviews less than 10000
b. Remove data above quantile range of 80%.

Using the second approach have

Movie is reviewed should be above 3884 times

Customer should have given a review above 79 times

The second approach is more flexible.

2.4. Creating train test sets

We shuffle a data and create training and testing sets. The trainset will be used to train all models. We use test for comparability between models with the RMSE metric.

2.5. Creating a sparse matrix

The data the we processed until now can be represented as a sparse matrix with users on top and movies down the side. Each cell in the matrix will be either rating (1-5) or is blank. This will give a matrix with 8.5 billion entries out of which only 100 million are filled. The most challenging part is to load all the data to create this matrix. I tried solutions like segmenting and combining approach where I created sparse matrix for each user_data file and combining it. This approach seemed to work but took a very long time. The other approach I tried was using MapReduce in Hadoop. I found a very good MapReduce example for creating a pivot table. Unfortunately, it did not work for very well as it did not have column headers and would usually break when loading the whole dataset. I tried to modify the code but due to the lack of time was unsuccessful executing the modified code.

2.6. Applying models.

I tried the following approaches to get a good prediction for ratings.

2.6.1 Mean Approach

We can predict the rating using the following way.

There are few factors of a rating:

a. Baseline rating: This can be a mean of all user and movie ratings
b. User specific rating: Maybe a user rate a movie higher than normal.
c. Movie specific rating: Maybe a movie gets less rating the normal due to controversy.

We use the following formula for prediction:

Rating = Mean(all ratings) + (Mean(user_ratings) – Mean(all ratings)) + (Mean(movie_ratings)-Mean(all ratings)

This method gives a good prediction, but this approach is biased and favors movies with fewer ratings, since large numbers of ratings tend to be less extreme in its mean ratings. To improve this approach the biases should be removed.

2.6.2 Matrix Factorization using Keras and Deep Learning for fast processing.

This is the most recent approach the gives the best result and can be run on a GPU for fast processing. Using matrix factorization, a large matrix can be decomposed into long but slim matrices. Gradient descent algorithm finds latent variables which gives a structure of the dataset. These variables can be used to reconstruct original matrix that fills up the missing ratings for each user. The model is not hyperparameter optimized and is not trained for many iterations. This model gives the best results and is fast in terms of processing.

2.6.3 Recommendation using surprise library using SVD++ , KNNwithMeans, SVD and KNN

The winners of Netflix prize used various collaborative algorithms like SVD and K- nearest neighbors for prediction. The models based on matrix factorization were found to be most accurate. A typical way to perform matrix factorization is to do a singular vector decomposition. Once you decompose the user movie sparse matrix you use a dot product of user vector and movie vector to get users predicted rating. Neighborhood based approaches are very popular collaborative filtering methods. The Netflix Prize has clearly shown that these methods can easily be beaten in terms of speed and accuracy by simple factor models. As individual models, they do not achieve outstanding low RMSE values [2]. The scikit-surprise library is a good library to test these approaches. This library contains quick executions for the above methods.

SVD++ gives the best results. But these model that I have implemented are without biases or weights. Regression techniques can be applied to find weights and applying these weights to SVD can give better RMSE.

2.6.4 Using Pearson R correlation

This is an approach where instead of predicting a rating it finds correlations to give similar movies. Using Pearson correlation, we find linear correlation between review scores of all pairs of movies. The we provide the top 10 movies with highest correlation score.

The following approaches were not implemented but if used can improve the prediction significantly.

2.6.5 Restricted Boltzmann Machines

Using Restricted Boltzmann Machines on large sparse matrix can give very good results. The paper mentioned in [7] shows how to implement such a model.

2.6.6 Blending or Ensembling techniques

To improve overall RMSE ensemble models that have distinct effects, even the small ones

For example, models like KNN does not perform well on their own but when blended with other models can give significant results

Feature Weighted blending techniques:

Suppose we have collection of models we can blend the models using linear regression. Also, multiple linear regressions can be used for different regions. For example, dividing regions into time periods and applying regression to each region.

Other approaches

1. Neural Networks
2. Gradient boosted decision trees

3. Results

The Netflix data set contains over 100 million observations and none of the above-mentioned approaches can easily deal with such large data sets. The above approaches were implemented on 25% of data to get the following results.

Model	RMSE
Mean Approach	0.998
Matrix Factorization using Keras	0.916
SVD	1.001
SVD++	0.995
KNN	1.053
KNNwithMeans	1.195

Implementing these techniques on the whole data set can significantly improve results. Also using ensembling techniques can significantly improve the RMSE.

4. Conclusion

This paper presented various approaches and techniques that can be taken to predict ratings for users on Netflix Data.The important factor that we need to consider when dealing with such large data sets is the lack of memory for processing the data. In this paper we discussed how to process such large dataset. I provided a few learning methods that can help improve the total RMSE. In the main competition, the techniques used by the Netflix prize winners combine hundreds of models to get marginal improvements in RMSE on the expense of the best available hardware. I have shown some new models that can perform faster and better than the previous algorithms. Libraries like Keras and surprise library for recommendation systems provide faster and simpler methods for recommendation and can provide good results.

5. Future Work

I would like to implement blending techniques and few deep learning techniques to find better RMSE. Also, I would like to implement these models on the whole dataset by implement a better MapReduce version to create the User-Movie sparse matrix. There are many other techniques that I have not mentioned like Global Time Effects that can be implemented.

6. References

[1] https://github.com/john-hawkins/Map-Reduce-Data-Pivot

[2] https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

[3] http://www.cs.toronto.edu/~fritz/absps/netflix.pdf

[4] https://www.mimuw.edu.pl/~paterek/ap_kdd.pdf

[5] https://arxiv.org/pdf/1409.2944.pdf

[6] https://www.netflixprize.com/assets/ProgressPrize2008_BigChaos.pdf

[7] https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf