Recommender System Project Report

Prashant Garg

16201447
COMP 41440

1 Introduction

The goal of this report is to explain the implementation of Recommender System using collaborative filtering approach.

Collaborative Filtering, is a recommendation algorithm which gives the predictions and recommendations on ratings based on the neighborhood. The neighbors are the other users present in the system. The neighborhood plays an important role in collaborative filtering as they are responsible for predicting the other user's opinion.

There are two types approaches in Collaborative Filtering,

- User-based collaborative filtering approach
- Item-based collaborative filtering approach

In our implementation, the **user-based collaborative** filtering approach is used. In this approach the predictions of ratings are made based on similarity of one user with the other users in the system. The similarity between the users is found using Cosine Similarity and Pearson Correlation Coefficient Similarity formula. After the evaluation of similarity between the users, the Predictions are made based on Average Rating, Majority Voting or Resnick formula. The neighbourhood size is also considered with making the decisions about these ratings.

For this project, another technique **Mean Prediction rating** is used to predict the rating for a movie. It is **not a Collaborative Filtering approach**, but does reasonably well for the MovieLens dataset used in the project.

The evaluation of these predictions techniques is done based on root mean squared error, coverage and runtime.

There was a problem of **sparsity** which arise because enough data was not present to predict the rating for some movies for the users while executing the results. This affected the coverage metrics for the recommendations made.

2 Collaborative Filtering

As, discussed in the introduction, the Collaborative Filtering methods are based on neighborhood, and predict ratings by referring to users whose ratings are like the queried user, or to items that are like the queried item.

The following images are the perfect example of the two approaches:

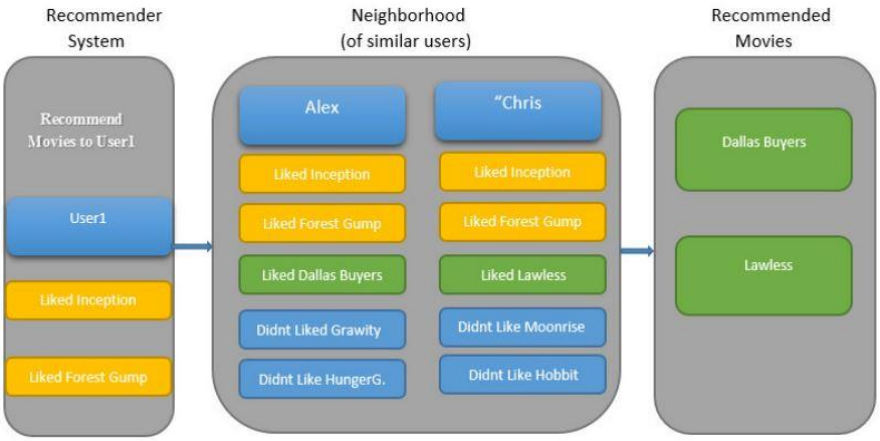


Figure 1: User Based CF Approach

In this image, we are supposed to recommend a movie to User1 based on its similarity with other users in the system. It is found that, User1 had initially liked the movie Inception and Forest Gump. Keeping this preference in mind, users who liked these movies are found. It is found that User Alex and Chris, liked these two movies too. Now, as the similarity between User1 and the other users (i.e. Alex and Chris) is found. The movies liked by User Alex and Chris are recommended to User1 assuming similar users have similar choices.

Let us look at the second image now,

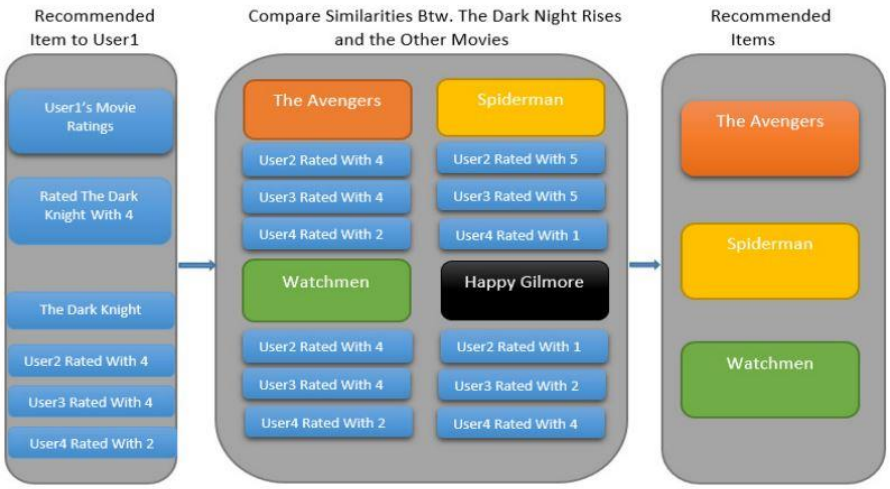


Figure 2: Item Based CF Approach

In this example, we are supposed to predict the movie to User1 based on the similarity of rating for movie already given by the user for example, The Dark Knight which the user rated as 4. The movies Avengers, Watchmen and Spiderman were the most similar movies based on the rating assigned to them by other users in the system. Hence, based on item similarity these three movies are recommended to User1.

2.1 An Overview of Collaborative Filtering

To provide recommendations to users, simply having a recommender algorithm is not sufficient. The algorithm needs a data set on which to operate — a ratings matrix (or its functional equivalent) must somehow be obtained from the users of a system

In our implementation of User based Collaborative filtering, the representation of the Users, Movies and Ratings pairs is done using two data structure.
The first one maps every user with the list of movies rated by him and another one associates every movie with list of user ratings.

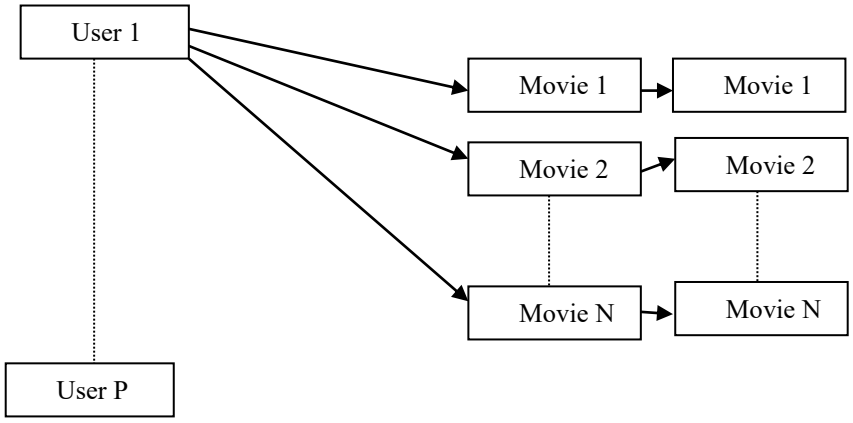


Figure 3: Data Structure 1 (User as Key)

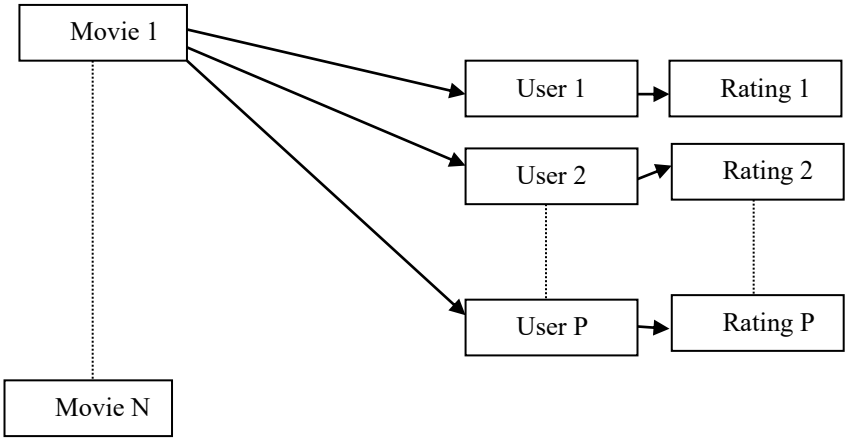


Figure 4: Data Structure 2 (Movie as Key)

.
Reason for choosing this Data Structure:

1. **Data Structure 1:** Finding similarity between users becomes easier by iterating User Map over another User Map and passing the intersected movies-rating lists for comparison.
2. **Data Structure 2**: The second data structure becomes handy for Mean Rating predictions and Leave-One-Out strategy. Here we can simply extract the tuple of [user, item, rating] from this map and implement the LOO strategy.

In the next sub-sections, the discussion on the similarity and prediction techniques used in the project will be discussed.

2.2 Approach 1 – Mean-Rating Prediction

Mean Rating Prediction technique is a **Baseline Prediction technique** which can be used without using the collaborative filtering. It can be used to predict the rating for the new user as well, because it does not require ratings from similar users for prediction.
There are many ways in which the baseline techniques can make predictions. In this project, the **average rating method** is used. The ratings are predicted based on the average ratings of the movies. It does not use any user or item similarity measure.

2.3 Approach 2 – Distance-Based CF

In the distance-based collaborative filtering approach, Cosine Similarity is used to evaluate the similarity between the users of the system. In cosine similarity, each user is represented as a vector of ratings, and the similarity is provided by evaluating the cosine of the angle between their vectors.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Once, the similarity between the users is found, The Neighborhood Policy is used to construct the neighborhood of similar users (relative to the target user).

Based on this neighborhood, predictions are made about the rating of movies for the target user.

In my implementation, two such techniques are used:

1. **Average Rating** – The average rating sums the rating of item retrieved from the neighbors of the target user and then find the average of it.
2. **Majority Voting** – It calculates the rating class in which most ratings of the neighbors fall, and assign that rating to the target user.

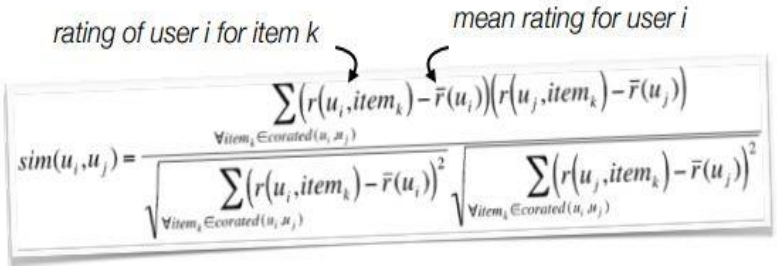
There are **tweaks done** while computing the similarity and giving out the predictions **in my implementation**:

1. Predictions are only made when the similarity between two users is more than **88%**.
2. While, computing the similarity, there is a **penalty** on similarity if there is very **less overlap** of common movies between the two users. Also, there is **bonus** given when the **overlap is more and the cosine similarity** is good too.

The above two tweaks improved the coverage and RMSE for the predictions.

2.4 Approach 3 – Resnick's Prediction Technique

The Resnick's Prediction Technique uses correlation between the users to predict the rating for the target user. This correlation is found using the Pearson Correlation. The Pearson Correlation method computes the statistical correlation between two user's common ratings to determine their similarity. The formula to compute the correlation similarity is as follows.



Pearson correlation is useful because it computes similarity on the relative scale rather than computing over an absolute scale. The user might be like other user but their absolute scale might differ. For example, .

Sports	Hockey	Football	Basketball	Cricket
User 1	2	1	2	2
User 2	4	2	4	4

The rating of User1 and User2 are quite dissimilar but while considering the correlation between the two. i.e. User2 gives double the rating as User1.
Using Pearson Correlation similarity formula, User1 would be nearer to User2.

Once, the Pearson Correlation similarity is found, the predictions are made based on these similarities by the **Resnick formula**, which is as follows

$$pr_{x,k} = m_x + \frac{\sum_{u_y \in N_x} (r_{y,k} - m_y)\,\mathrm{sim}(u_x, u_y)}{\sum_{u_y \in N_x} |\mathrm{sim}(u_x, u_y)|}$$

There is a **tweak done** while computing the prediction **by the Resnick formula**:

1. Predictions are only made when the correlation between two users is more than **0.** This is lead to better accuracy in terms of RMSE.

3 Evaluation

For the evaluation of these three techniques, Leave-one-out Testing strategy is used and three different evaluators used are root mean squared error, coverage, runtime. Leave-one-out is a variation on cross-validation testing where a single set of observation from the dataset (e.g. User – Item -Rating) is used as the test data with the remaining data as the training set. This process is repeated for all the dataset entries.

The results of the Leave-One-Out strategy for the three approaches are stored in different CSV's ***Technique1_Prediction.csv*** (Mean Rating Prediction), ***Technique2_Prediction.csv*** (Cosine Similarity), ***Technique3_Prediction.csv*** (Resnick Predictions) stored under the executable folder of the **16201447.zip** file. The files will be updated on running the Recommender System application with different configurations (**i.e. different neighborhood**).

3.1 Dataset

The Dataset provided for this project is the MovieLens "**100k**" dataset with **1682** movies and **943** users. The user rating for the movie varies on the scale of **1-5** and a total of **100,000** ratings are given by the users for the movies.

The rating density for this dataset is **6.305 %.** The rating density for this dataset simply depicts that **the dataset is quite sparse** to be handled efficiently by 2-dimensional array.

The statistics of Mean, Median, Standard Deviation, Max, Min ratings per user and per movie are stored in two different csv files ***Stats_User.csv*** and ***Stats_Movie.csv*** respectively under the executable folder of the **16201447.zip** file.

3.2 Methodology & Metrics

The methodology used for different techniques are as follows:

1. **Mean Rating Prediction:** The movie map (Data Structure 2 - *refer section 2.1*) serves as training dataset. The testing dataset is the (User – Item - Rating) tuple from input dataset. This tuple is removed from the training dataset and predictions are made based on the average rating given by the other users for the same item.
2. **Distance-Based CF:** The input is the user map (Data Structure 1 - *refer section 2.1*) which is iterated over itself to find similarity between the users. After this, based on neighborhood size, the similar users are extracted and two different methods are used to predict rating from these extracted users for the User_Item _Rating tuple [This tuple is excluded from Similarity List].
3. **Resnick's Prediction:** The input is the user map (Data Structure 1 - *refer section 2.1*) which is iterated over itself to find correlation between the users. After this, based on neighborhood size, the correlated users are extracted and Resnick formula is used to predict the ratings for the User_Item _Rating tuple [This tuple is excluded from Correlation List].

Metrics used for evaluating these techniques are as follows:

1. **Root Mean Squared Error (RMSE) Metrics** - computes the mean value of all the differences squared between the true and the predicted ratings and then proceeds to calculate the square root out of the result
   Advantage is that it produces errors that are on the ratings scale.
   Disadvantage is that large errors may dramatically affect the RMSE rating

2. **Coverage Metrics** – denote the percentage of items for which the method can compute predictions
   Advantage is that it will allow us to check the strength of the system for making prediction
   Disadvantage is that sometimes, attempt to increase the coverage leads to poor predictions.

3.3 Approach 1 Results

Following are the results for the **Mean Rating Predictions**:
The overall root mean squared error comes out to be **1.021** and the system could cover **99.859%.**

Approach	RMSE	Coverage (%)
Mean Rating	1.021	99.859%

```
--------------------------------------------------------------------------------
Recommendation Technique 1 : [Average Rating]   ( L10 : Technique1_Prediction.csv Generated ....)

COVERAGE  = 99.859 %     OVERALL  RMSE = 1.021   TIME ELAPSED = 0 seconds
--------------------------------------------------------------------------------
```

And the overall runtime for this approach on running it **10 times** in a loop was **2 seconds** which is very good.

The model gave such a high coverage because all the users of the system were considered (except the target user) while predicting the rating. This approach can be very effective while predicting the movie ratings for a new user.

3.4 Approach 2 Results

The second technique Distance-Based CF had two approaches for predicting the rating of the users: Average Rating and Majority Rating. The result for the approaches are as follows:

k	Average Rating (RMSE)	Majority Voting (RMSE)	Average Rating (Runtime in seconds)	Majority Voting (Runtime in seconds)	Average Rating (Coverage in %)	Majority Voting (Coverage in %)
5	1.349	1.527	41	42	61.765	61.765
10	1.262	1.411	44	45	78.955	78.955
25	1.173	1.402	49	51	93.409	93.409
50	1.111	1.311	54	54	97.676	97.676
100	1.077	1.262	63	63	99.536	99.536
150	1.061	1.232	71	70	99.868	99.868
200	1.054	1.217	74	78	99.98	99.98
250	1.049	1.206	80	82	99.98	99.98

Table 1: Approach 2 Results

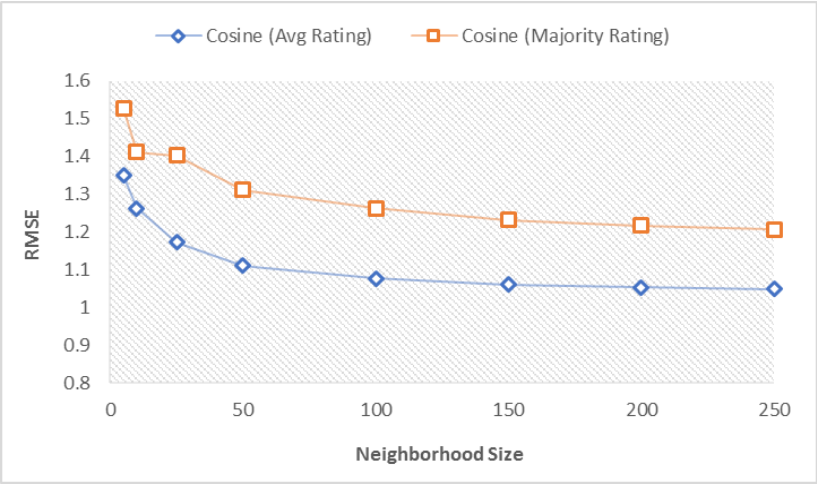


Figure 5: Distance-Based Approach RMSE

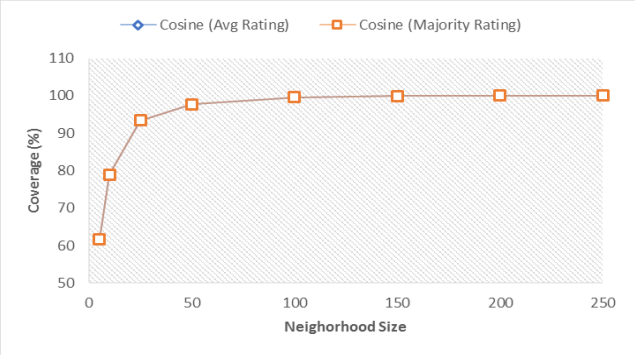


Figure 6: Distance-Based Approach Coverage

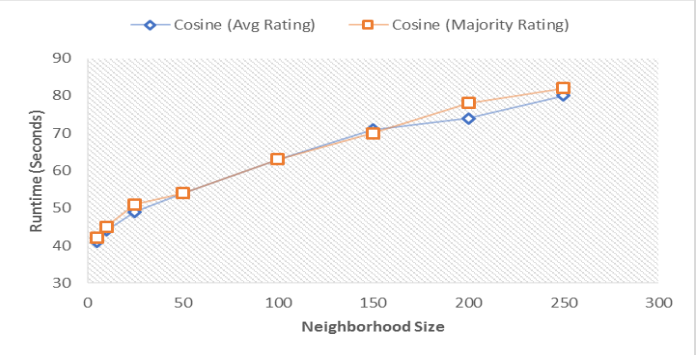


Figure 7: Distance-Based Approach Runtime

Observations:

1. For every neighborhood size, the average rating approach provided better RMSE than the Majority voting approach. The reason behind is that majority voting is sensitive to the neighbor size. For e.g. the if neighborhood size is 5, and the closest neighbor gives accurate rating while the other 4 give inaccurate rating. The target user is probably assigned wrong rating due to majority rating increasing the RMSE.
2. The RMSE for both the approaches is quite high in initial stages, this is due to the tweak made by me while evaluating the similarity to increase the coverage of the system.
3. The RMSE and Coverage of the system both get better with the increase in the neighborhood size. The coverage increase as there is increase in number of target users. The RMSE increased because very similar users (which were penalized by the tweak made) came within the neighborhood and contributed in making predictions.
4. At the **neighborhood size of 150**, the system is yielding the best appropriate results in terms of **coverage, RMSE and runtime**.
5. Beyond the range of 150, there is plateau for both Coverage and RMSE, and runtime becomes more as well. So, it is better to choose neighborhood size of 150 for this dataset if using Distance-Based CF technique for making predictions.
6. For both the approaches used, coverage remains equal for both (as similarity function used is same for both) and **runtime** is **almost equal**.

3.5 Approach 3 Results

The result of the Resnick predictions are as follows:

Neighborhood Size	RMSE	Coverage (%)	Runtime (Seconds)
5	0.735	7.556	60
10	0.723	13.537	66
25	0.724	29.243	68
50	0.734	53.419	69
100	0.736	83.734	80
150	0.733	94.313	100
200	0.743	97.523	121
250	0.758	98.739	140

Table 2: Approach 3 Results

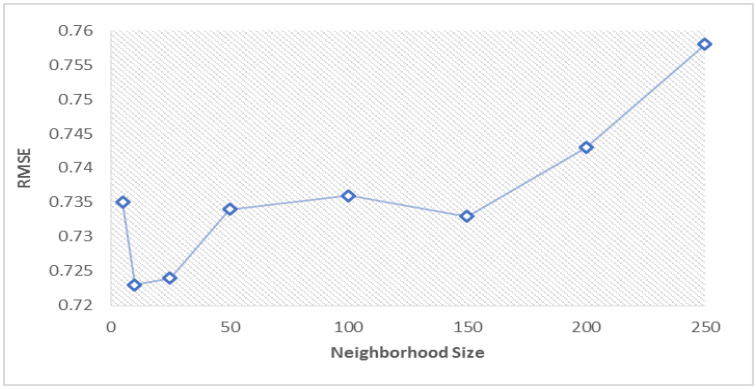


Figure 8: Resnick's RMSE

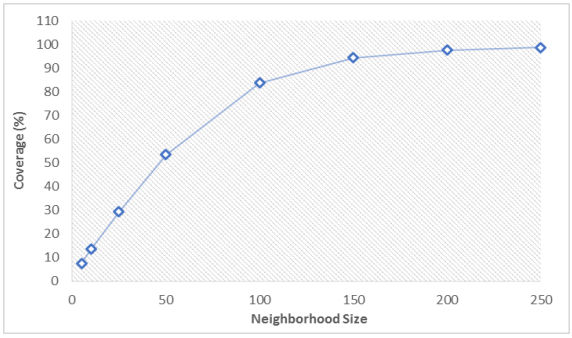


Figure 9: Resnick's Coverage

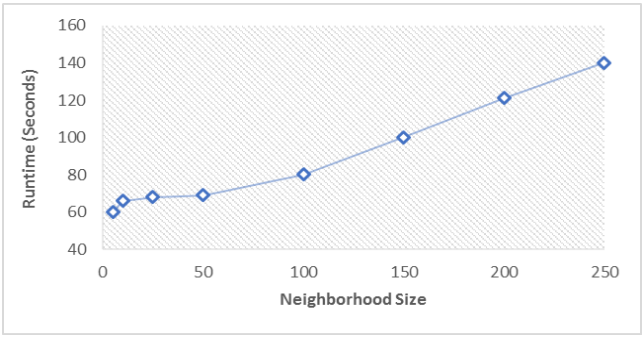


Figure 10: Resnick's Runtime

Observations:

1. In general, with **increase in size of the neighborhood**, all the three parameters **RMSE, Coverage and Runtime tend to increase**. i.e. the **Runtime and RMSE gets poorer with increase.** This behavior is normal, as **more no. of less-correlate neighbors are used to predict the ratings which increase computation time and decreases accuracy**.
2. The **coverage plateaus** at neighborhood size of **250**. On contrary to it, both, **errors (RMSE) and runtime linearly with this increase**.
3. At neighborhood size of 10, the RMSE is better when compared to neighborhood size of 5. That's due to the inconsistency in the dataset.
4. At the **neighborhood size of 100, the system is yielding the best appropriate results in terms of coverage, RMSE and runtime**.

4 Discussion

In this section, the results from the three-different technique will be compared with each other.

Root Mean Squared Error

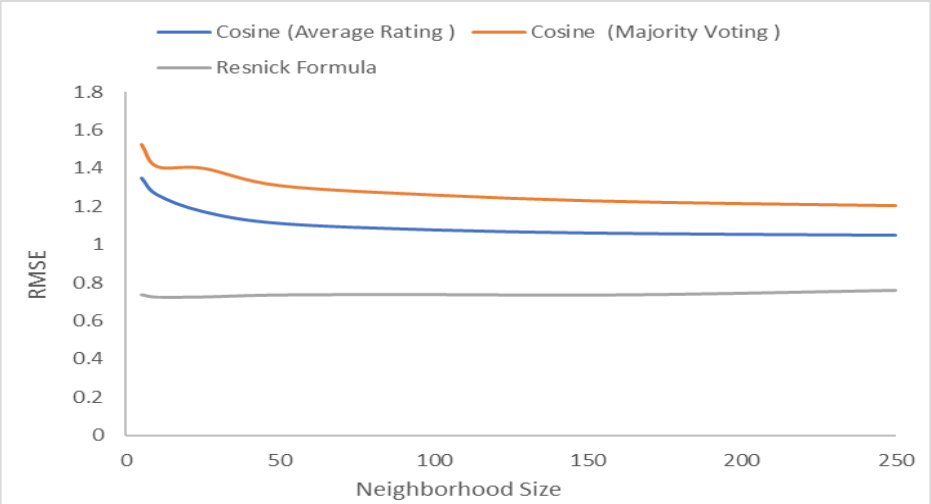


Figure 11: RMSE

As quite easily visible, for the neighborhood size (k) from the range of 5 to 250. **Resnick prediction** yielded the best prediction with the **lowest RMSE rate of 0.723** at **k = 10**. For **k >10**, the **RMSE increased linearly** though at a very low rate deteriorating the accuracy.
In contrast to it, the Average Rating & Majority Voting approach for Distance based Similarity got better linearly with increase in the neighborhood size from k =5 to 200.

A **plateau** in improvement of RMSE was observed **after k > 200**.

The RMSE for Mean-Prediction technique was constant at **1.021**, as it is not affected by the change in neighborhood.

Overall Accuracy(RMSE):
Resnick > Mean Prediction > Distance Based CF

Coverage and Runtime performance:

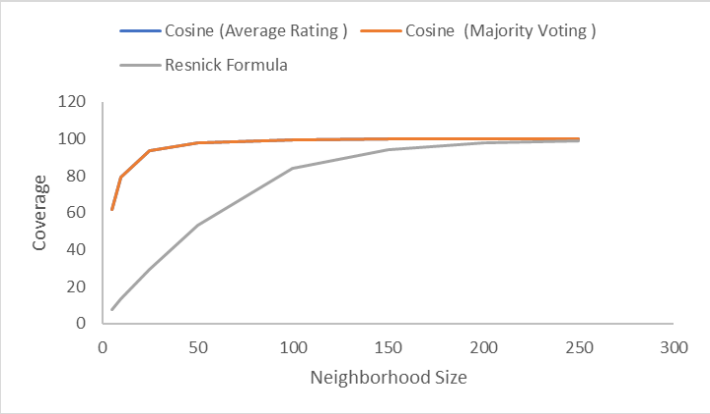


Figure 12: Coverage

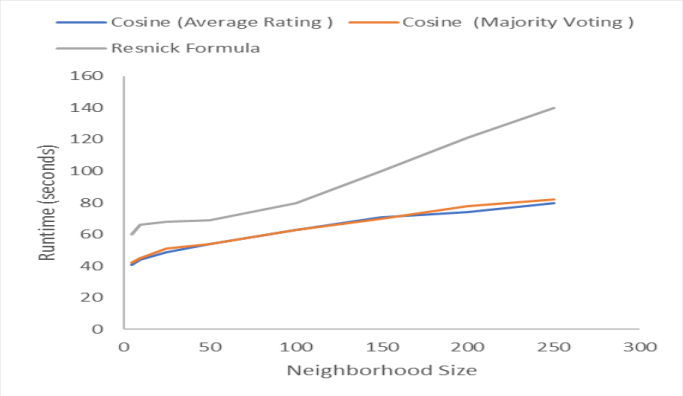


Figure 13: Runtime

There is a trade off seen between coverage to that of the Runtime of the system, for the second and third technique. As the efforts were made to increase the coverage of the recommendations for these systems, the runtime increased linearly on increasing the neighborhood size. The runtime for Mean-Prediction technique remained constant at zero seconds as it is not affected by the neighborhood size.

From the coverage point of view, the Resnick model was very poor from the **'k'** valued of range from **5 to 50**.The cosine similarity model was the fron-runner in terms of its coverage which start with around **60%** at **k = 5,** and went upto **99% at k=150**. The coverage of the Mean-Prediction technique remained constant at **99.85%.**

Overall Performance Analysis [Coverage & Runtime] : Mean Prediction > Distance Based CF > Resnick

As, the most important aspect of the Recommender system is Accuracy, we can simply say that the **Resnick Prediction offfers the best result for the dataset** given to us.

5 Conclusions

- Selecting a high-quality (that is, very similar) neighbors from a large set of candidates largely enhanced the prediction computation.
- For my implementation, the neighborhood size of 100 for Resnick and 150 for distance based CF (Cosine) is yielding the best results for the two approaches respectively.
- Mean-Predicting technique is also very handy, if the user is new to the system. They don't compute the similarity or correlation between the users of the systems.
- Resnick Predictions were the best among the three models used in terms of accuracy (RMSE)
  Reason: Subtracting the users mean rating compensates for the differences in user's usage of the rating scale.
- Resnick's predictions consider correlation, which is a very vital component. Some users have the tendency to strictly rate any movie, while others rate freely. They might have same liking but different ratings. This concept of neglecting the scale of rating and considering them to be similar is
- The tweak made by me while computing the similarity for the users in approach 2 increased the coverage to great extant but had negative impact on RMSE too. But the RMSE wasn't getting impacted much, hence proceeded with the tweak.

6 References

- http://homepages.abdn.ac.uk/advaith/pages/teaching/abdn.only/AIS/lectures/abdn.only/CollaborativeFiltering.pdf
- http://www.stat.osu.edu/~dmsl/Lee_2012.pdf
- http://herbrete.vvv.enseirb-matmeca.fr/IR/CF_Recsys_Survey.pdf
- http://aircconline.com/ijcsea/V6N3/6316ijcsea01.pdf