RecSys Project Report

María Inés Aranguren

16208714
COMP 30490

1 Introduction

Collaborative filtering is a technique commonly used in recommender systems to make predictions about relationships between items and users. Collaborative filtering is based on the idea that users that rate similarly to each other likely have comparable preferences and are therefore valuable resources in making recommendations for other users. This is one of the biggest benefits of collaborative filtering since it means that only data about the ratings of items by users is necessary to create predictions and no information about the actual items being rated is needed. In the context of this project, no information about the movies such as genre and country of origin or information about the user such as age or sex is used to make predictions. On another note, collaborative filtering works well with large collections of data and makes real-time recommending viable.

The two main types of collaborative filtering are user-based filtering and item-based filtering. This project focuses specifically on user-based collaborative filtering, meaning that similarities between the rating patterns of users, rather than similarities between items, are used to generate predictions and recommendations. Three different methods were used to create the predictions: simple leave one out, distance-based leave one out, and Resnick's formula. The metrics that were used to calculate similarity between users for the distance-based methods and Resnick's formula are cosine similarity and Pearson's similarity coefficient.

The most significant results from project suggest that the best approach for making predictions is using Resnick's formula with distance-based neighborhoods making slight improvements in RMSE, substantial improvements in time, though also characterized by reduction of coverage in comparison to performing mean-rating predictions with distance-based similarity. In this context, coverage describes the ratio of total user and item pairs for which a prediction can be made to the total number of user and item pairs. Furthermore, the results of this project also suggest that the best method for making neighborhoods is using the k-nearest users as neighbors. This is mostly due to the fact that predictions with k-nearest user neighborhoods are calculated much faster than those for neighborhoods made of the top N % of users (in terms of similarity to target user) and neighborhoods made up of all users that fit a certain similarity minimum with the target user.

2 Collaborative Filtering

2.1 An Overview of Collaborative Filtering

As mentioned above, the predictions in this project are calculated through user-based collaborative filtering algorithms that identify users that are closely related to a specific user and compile their preferences to form predictions about the given user's preferences. The three basic steps in user-based collaborative filtering are measuring similarity between users, forming neighborhoods, and computing predictions. These steps are more thoroughly discussed in the following sections detailing the different approaches.

The strongest pros of collaborative filtering is that no specific data about items or users is needed other than the ratings, predictions can be made in real-time with large datasets, and recommendations can be made between unusual product classes.

On the other hand, there are also a few drawbacks to the collaborative filtering approach. As it is with all recommendation generation techniques, the results of collaborative filtering are very reliant on both the quality and quantity of the data available. One of the biggest challenges of collaborative filtering is that predictions are hard to make when ratings data from users is sparse. It so happens to be that this is the case for most applications where recommender systems are used, such as music or movie recommender systems. In the MovieLens dataset used in this project, there are a total of 100,000 ratings represented by 943 different users rating 1,682 different items. With this amount of users and items, there are a total of 1,586,126 user and item pairs for which there can be a prediction made. However, this is very challenging because the data is very sparse and only represents 6.305 % of those pairs. Another challenge of collaborative filtering is known as the "Cold Start" problem. This refers to the situation in which a system is new, a user is new to a system and therefore has not rated many items, or an item is new to the system and therefore has not been rated many times. Without much knowledge about the user's preferences, the recommender system cannot calculate what the user is likely to prefer. Moreover, it also affects new items as they are less likely to be recommended until there is a sufficient amount of available ratings for them.

2.2 Approach 1 – Mean-Rating Prediction

In the first approach, predictions were made using the mean-rating method. In this method, predictions for a given user and item pair are made by averaging the ratings for that item across all other users that have rated it. This characterizes the approach with a very high coverage since predictions can be made for any user and item pair as long as there is one other user that has rated it which in turn makes predictions calulable for all user and item combinations except those which the user is the only user that has rated the item before.

2.3 Approach 2 – Distance-Based CF

In distanced-based collaborative filtering, predictions are more personalized to a user's preferences since predictions are based on ratings of similar users. This requires that similarity be measured between users. In this project, this is done both with the cosine similarity and pearson's correlation coefficient.

In cosine similarity method, the ratings of co-rated items of the two users are treated as vectors and the angle between the two is used to measure their similarity and thus is represented as a value between 0 and 1. It is broadly considered to be a good and efficient metric to use for collaborative filtering. The cosine similarity function is given in the following equation:

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

where *i* and *j* are two different users.

This image illustrates how the cosine similarity can approximate similarity between two users by comparing how they rate different items.

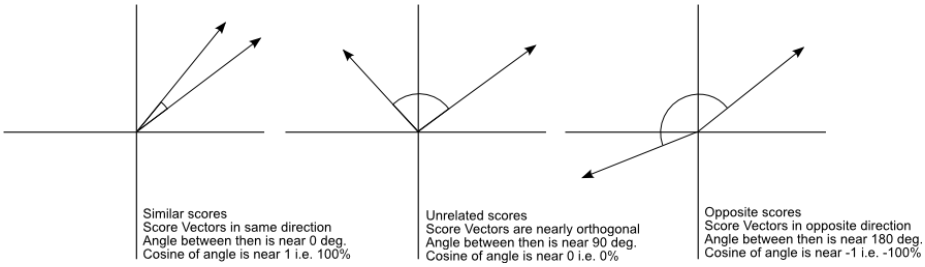


Figure 1. Cosine Similarity Illustration

In the pearson's correlation coefficient method, the items co-rated by both users, along with the mean rating for the two users are considered. The similarity between the two represents the extent to which two ratings for co-rated items are linearly related and is expressed as a value between -1 (as a perfect negative correlation) and 0 (as a perfect correlation). The formula for calculating Pearson's correlation coefficient is given in the following equation:

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}}$$

where *i* and *j* are two different users and *u* represents items in *U* (items co-rated by both users)

Many argue that Pearson's correlation coefficient is a better way to calculate similarity because it is not only magnitude-based as it is sensitive to the case in which users use different internal scales. For example, it could be the case that user A's

ratings, though proportionally related, are very different in magnitude to user B's ratings and quite close to user C's ratings. In this scenario, Pearson's would recognize that user A and user B are more similar since they are closely correlated whereas cosine similarity would suggest that user A is actually closer to user C because the value of the ratings is closer.

The results gathered from cosine similarity and Pearson's correlation coefficient methods are used to identify which users are most similar to a given user and therefore more suitable for predicting the target user's preferences. In this project, three different methods were utilized to create neighborhoods. In the first, the k-nearest users to a given target user were considered part of that user's neighborhood. In the second, all users pairs that met a specific similarity minimum were mutually considered to be neighbors. In the third, the top n percent of users were considered to be part of an target user's neighborhood.

These neighborhoods were then used to create predictions made by finding the mean item rating of the given item across all users in the target user's neighborhood. The calculation process is the same as that in approach 1 but the number of users being considered is reduced and the quality of their ratings is improved as it is more informative of the target user's preferences.

2.4 Approach 3 – Resnick's Prediction Technique

The last prediction generating approach is very similar to the previous approach in that it also utilizes the distance-based similarity metrics and the three neighborhood-forming techniques. The difference between this approach and the previous one is that this approach utilizes Resnick's prediction formula to generate predictions rather than the mean-rating method used before. Resnick's formula makes use of the similarity metrics by assuming a correlation-based similarity function (with values between -1 and 1) and including that in the calculation for the prediction. Since cosine similarity is on a scale of 0 to 1, these values were adjusted to match the -1 to 1 scale. The predictions Resnick's formula makes take into consideration the target user's average rating and create adjusted forms of that rating according to the feedback from its neighboring users. The adjustment of the target user's mean rating is based on how neighboring users have rated it in relation to their own average rating and weighs it proportionally to the correlation between the two users. The equation below represents Resnick's prediction formula:

$$c(i) = \bar{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \bar{p}) sim(c,p)}{\sum_{p \epsilon P_i} |sim(c,p)|}$$

where *c(i)* is the prediction made for target user *c* and item *i* using users in neighborhood *P*.

3 Evaluation

3.1 Dataset

The data used in this analysis was produced by researchers in the GroupLens team in Department of Computer Science and Engineering at the University of Minnesota. The data was collected over various periods of time by the group and is a publicly available online.

There are several different versions of the MovieLens dataset that offer different scales of ratings and types of information. For this project, the 100 K MovieLens ratings dataset was used. The data is given as a CSV file that has a value for a user_id, item_id, rating, and timestamp in each row which corresponds to a specific instance of a movie rating.

3.2 Methodology & Metrics

There were two general prediction calculation methods used in this analysis. The first is the leave one out strategy. In this method, predictions for a specific user and item pair are calculated as the mean of all ratings over that item except for the rating given by the specified user if it exists. One of the major advantages of this approach is that predictions can made for any target user as long as there is at least one other user that has rated the target item.

A more personalized method to generate predictions is using Resnick's formula. This formula takes into consideration the correlation between the rating patterns of two users and creates predictions by adjusting the mean rating of the target user positively or negatively based on the ratings of related users weighted by their similarity with the target user. Similarity is measured with the previously mentioned Pearson's correlation coefficient and cosine similarity.

The three parameters that are analyzed when making predictions is the coverage, time required to make predictions, and root mean squared error (RMSE).

3.3 Approach 1 Results — Mean-Rating Prediction

Metric	Value
Overall RMSE	1.0205926320022516 stars
Total calculation time *	155.56400771 seconds
Coverage	0.9999111041619644

Table 1. Results of mean-item rating predictions with simple leave one out

The table above illustrates the results of the mean-rating prediction cycle performed with simple leave one out. As can be seen, the best characteristic of this technique is the high coverage of predictions it yields. However, this is at the cost of making predictions that are over 1 star off from the actual rating.

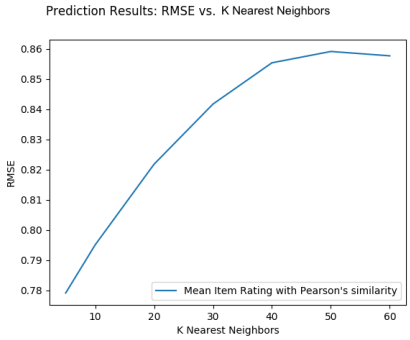
The results of this test were saved in simple_l10_results.csv

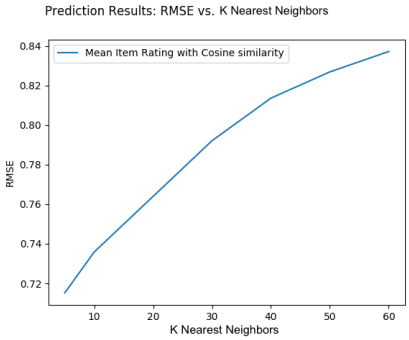
* This time is averaged throughout 10 different runs.

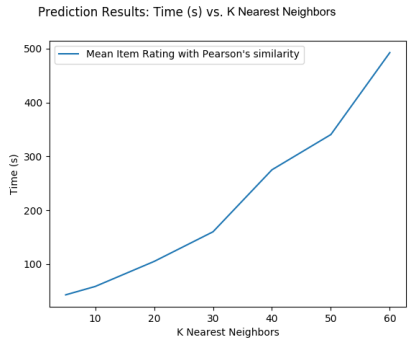
3.4 Approach 2 Results – Distance-Based CF

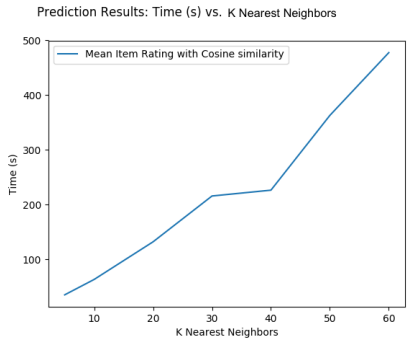
The following graphs illustrate the results of the distance-based collaborative filtering approach performed with three different methods of identifying neighborhoods and two different similarity metrics.
In efforts to look particularly at the time needed to make predictions, the time plotted in these results only includes the time spent on making predictions after having calculated the similarity between all users and created neighborhoods for the corresponding parameter of neighborhood similarity requirement.

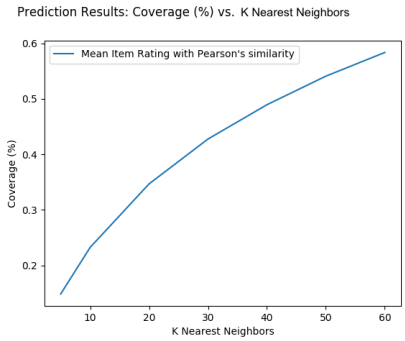
3.4.1 K-Nearest Neighbors

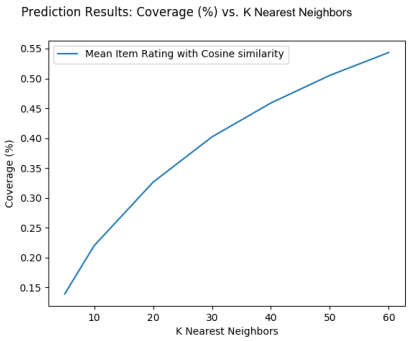






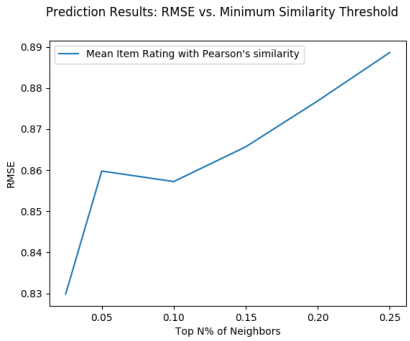


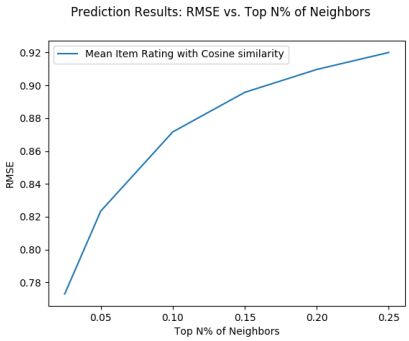


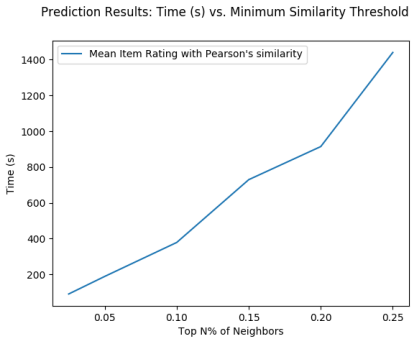


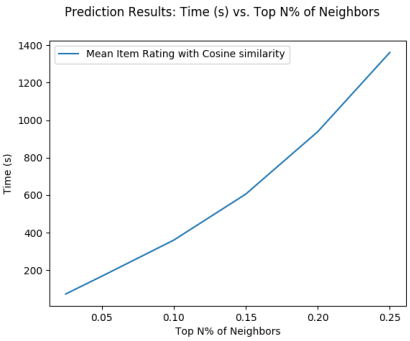
As can be expected, the general trend is that all of the three parameters (RMSE, time, and coverage) increase with a greater k value for predicting ratings with k-nearest neighbors. With slight differences in the curvature of the plots, Pearson's similarity metric and cosine similarity seem to be quite comparable.

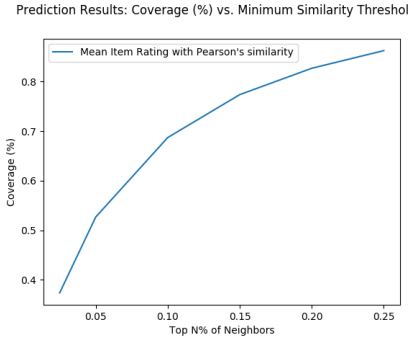
3.4.2 Top N% of Neighbors

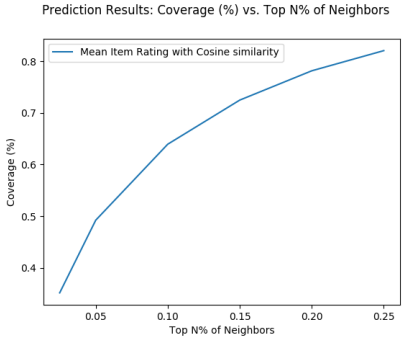










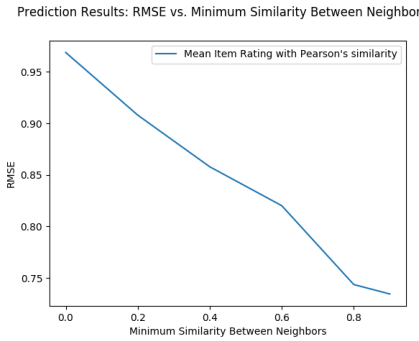


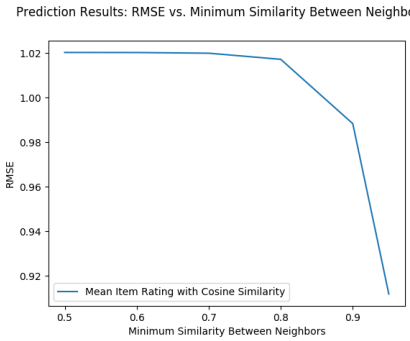
In these graphs, there is a general increasing tendency for RMSE, time, and coverage with increasing N for top N % of neighbors. This can be expected since increasing N means that neighborhoods are larger. With larger neighborhoods, several effects may be occuring. Firstly, there is a higher chance that users that are not as closely related to the target user will negatively affect the accuracy of the predictions. Moreover, since there are more neighbors, more data has to be processed in order to make a single suggestions. Lastly, the coverage also increases because there is more of a chance that the neighbors in the target user's neighborhood have rated the target item and can contribute to that prediction.

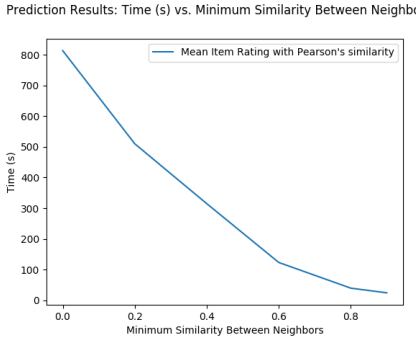
On another note, there does not seem to be a significant difference in the performance of the prediction generation between using Pearson's similarity and cosine similarity.

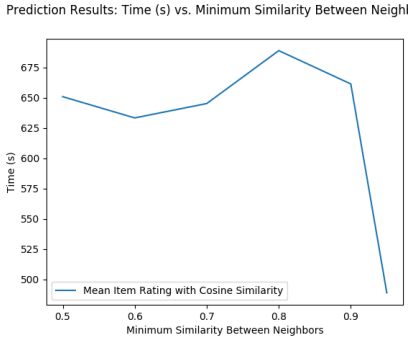
Furthermore, the curves shown in the time graph are tremendously sloped illustrating very slow prediction generation with times reaching up to 1,400 seconds.

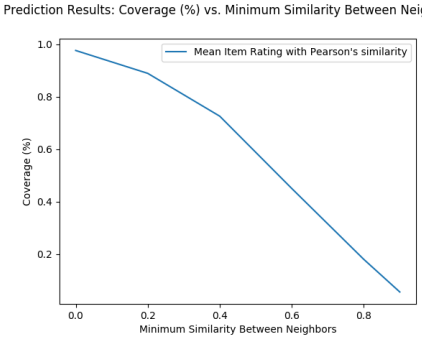
3.4.3 Minimum Similarity Threshold Neighbors

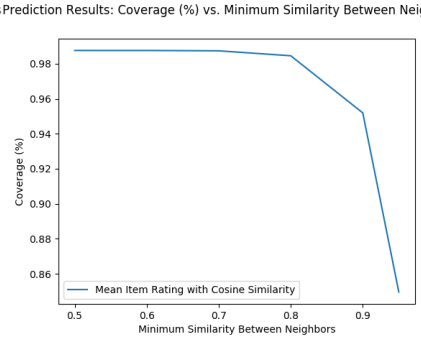












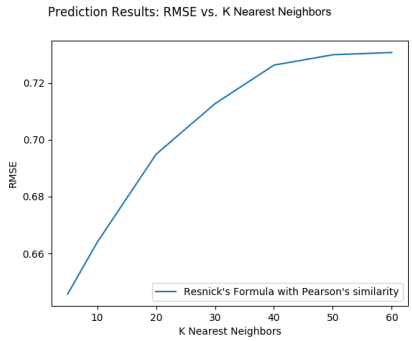
Since Pearson's similarity runs on a scale from -1 to 1 and cosine similarity runs on a scale of 0 to 1, the x axes of the two sets of graphs have been adjusted so that they are comparable to each other. This means that though they do not follow the same values, the values chosed correspond to the same scale of similarity in their corresponding similarity metric.

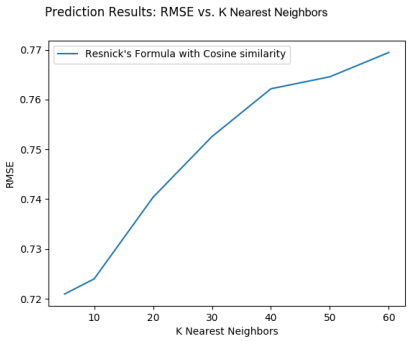
In these graphs, we can observe a general decrease in RMSE, time and coverage with increasing minimum similarity between neighbors. This makes sense because the higher the minimum similarity, the lesser amount of users that will meet that requirement and form part of the neighborhood. This makes the calculations more accurate and fast since it reduces the number of ratings and considers only the ratings of users most similar to the user target. Moreover, the coverage drops as well since the higher similarity requirement makes it more complicated for neighborhoods to be formed for all users.

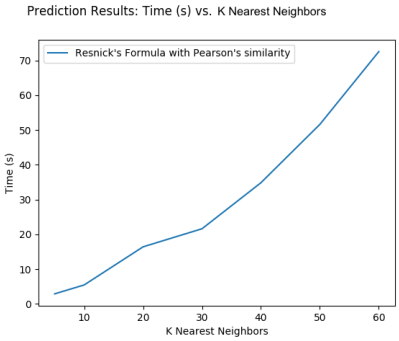
We can also notice remarkably high coverage. This is probably the case because the neighborhoods are made up of any users that fit the similarity requirement and so neighborhoods can be broader and it is more likely that there will be users in a target user's neighborhood that have ranked the target item and can contribute to creating a prediction. The coverage is particularly high in the case of cosine similarity. However, it seems that Pearson's similarity has slightly better performance in terms of RMSE and time.

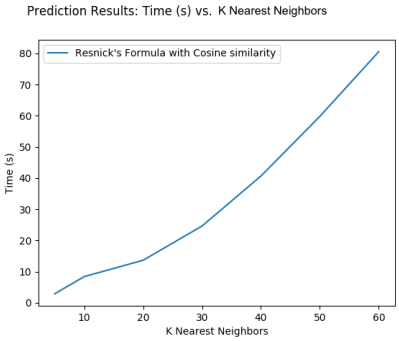
3.5 Approach 3 Results – Resnick's Prediction Technique

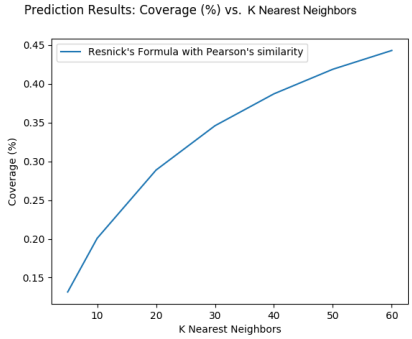
3.5.1 K-Nearest Neighbors

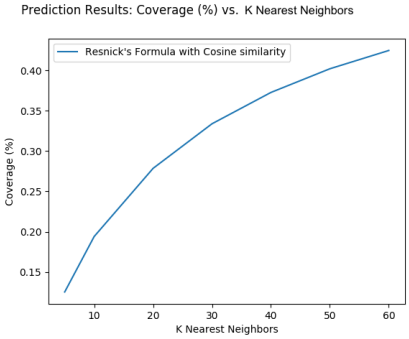








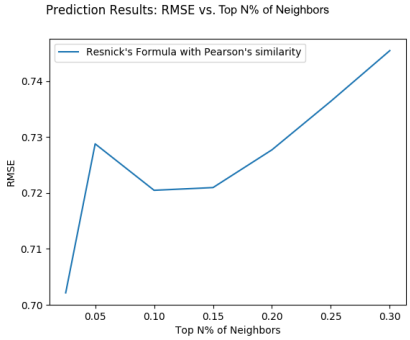


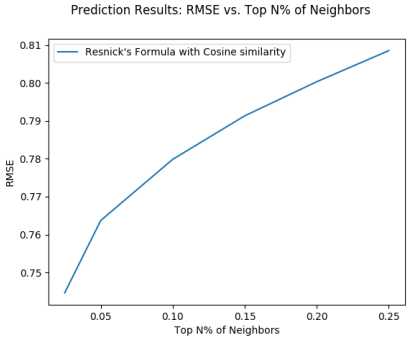


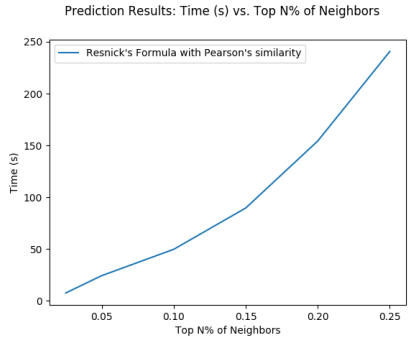
These tests show similar trends to those shown in section 2.4.1 in that an increasing k value results in increasing RMSE, time, and coverage. This makes sense because with a bigger neighborhood, there are more users with different profiles which make the RMSE increase, there is more data to be processed which makes the time increase, and there is more data to work with to make more predictions which increases the coverage. There seems to be slight differences in curvature between the Pearson's similarity and cosine similarity curves. However, their performances are

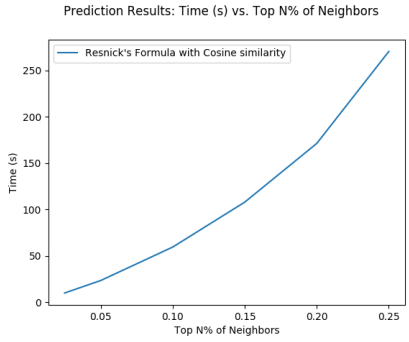
overall very similar. Interestingly, the results for this set of tests which uses Resnick's formula as opposed to the mean rating method does provide for significant improvements in both time and RMSE. Perhaps the most shocking of the two is time which drops from a range that goes up to 500 seconds in the previous approach to a range that only reaches up to 80 seconds in this approach.

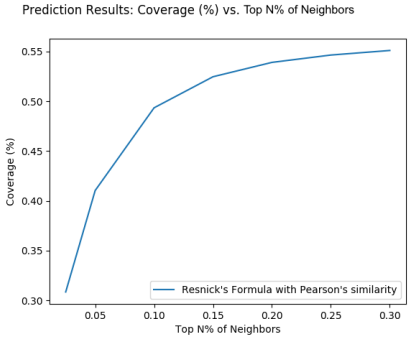
3.5.2 Top N% of Neighbors

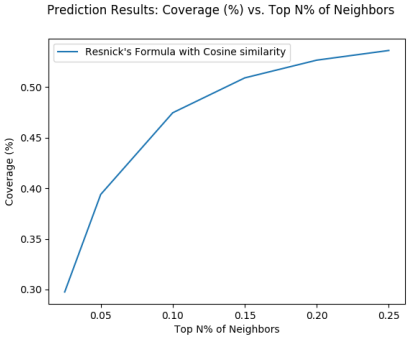










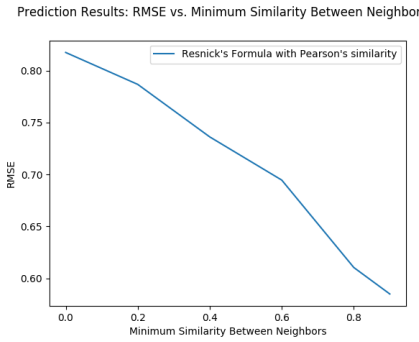


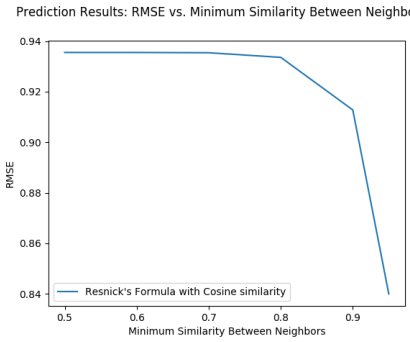
Once again in these graphs as was the case in the Top N % of Neighbors test runs with the mean-rating approach, there is a tendency for the RMSE, time, and coverage to increase with an increasing N.

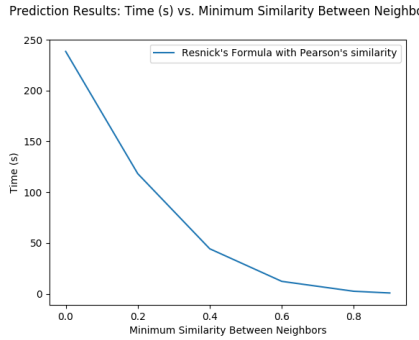
There does not seem to be considerable differences in performance between using Pearson's similarity and cosine similarity.

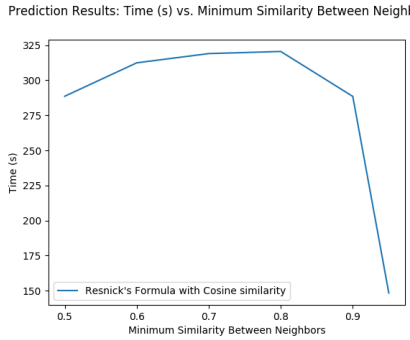
Interestingly, there seems to be a spike in RMSE in the trial performed with Pearson's similarity around N = 0.05. Though the reason for this is not certain, it could be that the neighborhood was so small that differences with only a few neighbors were enough to gravely misguide the prediction.

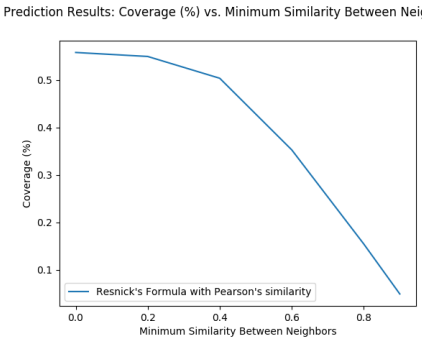
3.5.3 Minimum Similarity Threshold Neighbors

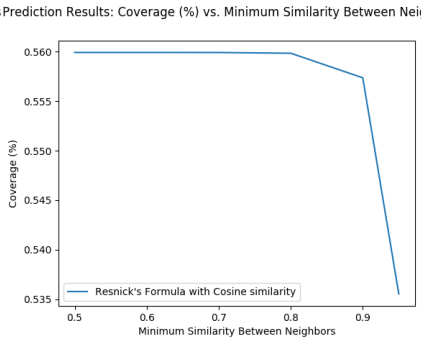












Notice here again that the x-axes of the two sets of graphs here have different values but have been adjusted to fit their corresponding scale.

In these graphs, we again observe a general decrease in RMSE, time and coverage with increasing minimum similarity between neighbors. Moreover, the curvature of the graphs again varies between the Pearson similarity graphs and the cosine similarity graphs posing a stronger curvature in the cosine similarity set. Though this is favorable in the coverage because it suggests that coverage slowly decreases even with increasing minimum similarity (which yields in smaller neighborhoods of closer neighbors), it is really unfavorable in terms of time and RMSE since it means that the minimum similarity has to be significantly increased in order to reduce error and time. In this way, Pearson's similarity outperforms cosine similarity in terms of RMSE and time but is outperformed by cosine similarity in terms of coverage. This can also be observed by noting the range of values of the y-axis on these graphs comparing between the two similarity metrics.

4 Discussion

The results of the experiments described above show that different approaches for a common task can lead to significantly different results. Moreover, it is often the case that one approach performs well in one way but might not perform as well in another sense. Overall, it seems that the best-performing approach is using Resnick's formula with distance-based neighborhoods. The results from the tests that employ Resnick's formula are distinguished with lower RMSE and faster time, though sometimes also slightly lower coverage. This is probably due to the fact that Resnick's fomula parts from the user's mean rating score and then adjusts it using the ratings of other users and weights those according to how close the neighbors are to the target user. This is unique and much more personalized than the mean rating approach, even when the users being considered are reduced to a specific distance-based neighborhood, as in the case of the second approach.

On another note, the results of this project also show that, for this data, the best way to form neighborhoods is to create them with the k-nearest users. Some of the biggest advantages of this approach are simple and powerful implementation and the fact that no training is needed. Some of the short-comings of this technique are the difficulty and trickyness in determining an appropriate to strike a balance between reducing noise yet maintaining a broad and diverse user pool. This algorithm can also be quite slow but in the case of this project, it was still faster than the neighborhoods formed by the top N % neighbors and comparable to minimum similarity neighborhoods.

With regards to the difference in performance between Pearson's similarity and cosine similarity, the two metrics proved to have similar performance in many of the trials. However, it also tended to be the case that Pearson's similarity

outperformed cosine similarity in RMSE and time though was outperformed by cosine similarity in terms of coverage. This can observed particularly clearly in the experiments in which neighborhoods are made with users of a minimum similarity.

Last but not least, one of the crucial components of this project is the manner in which data is organized in different data structures. The most important data components in making these predictions are the ratings and the similarity between different users. In this implementation, the ratings data is stored in two ways. In the item-based version, the ratings are stored in a dictionary of dictionaries where the key of the first dictionary is an item id and the dictionaries within that dictionary are made up of user keys and rating values. In the user-based version, the ratings are stored in a dictionary of User objects where the key is a user id and the value is it's corresponding User object. Each User object contains a user id, a dictionary with the ratings of that user organized by item id keys, simple statistics about the ratings such as minimum value, maximum value, mean, and standard deviation, and a list of neighbors. Organizing the data in this way allowed for faster searching of information and comparison across users. For example, having the item-based version of the ratings organized by item keys allowed for calculating the mean rating of that item excluding one target user in a single line. On the other hand, having the user-based dictionary made it very simple to find ratings of a specific user and items that were co-rated by two users.

5 Conclusions

In this project, experiments were done generate predictions using three different approaches: mean-rating predictions, distance-based predictions using mean-rating, and distance-based predictions using Resnick's formula. For the second and third approaches, distance between users was calculated both with Pearson's correlation coefficient and cosine similarity and neighborhoods were created with three different techniques: k-nearest neighbors, top N % of users (in terms of similarity), and minimum similarity threshold. This means that there were 37 different experiments in total. For the mean-rating predictions, results were calculated and presented in a table and for the distance-based predictions, results were calculated and plotted in graphs shown above.

There exist different ways in which this project could be improved. For example, the experiments could be repeated a number of times to measure more accurately and avoid bias that could have affected the results. For example, it could be the case that if the experiments were re-run the could be completed in a faster time if the machine they operated on permitted them to do so. Furthermore, it could be the case that the core data structures and general design of the code could be redesigned and better organized.

6 References

"Algorithms: Item-Based Collaborative Filtering". Cs.carleton.edu. N.p., 2017. Web. 10 Mar. 2017.

Cacheda, Fidel et al. "Comparison Of Collaborative Filtering Algorithms". ACM Transactions on the Web 5.1 (2011): 1-33. Web. 10 Mar. 2017.

Donovan, John and Barry Smyth. "Trust No One: Evaluating Trust-Based Filtering For Recommenders". (2017): n. pag. Print.

Perone, Christian. "Machine Learning :: Cosine Similarity For Vector Space Models (Part III) | Terra Incognita". Blog.christianperone.com. N.p., 2017. Web. 10 Mar. 2017.

Smyth, Barry. "Recommender Systems". 2017. Lecture.