Movie Recommendation System

Ankur Garg, Chirag Gupta, Pankaj More, Pranjal Singh
Indian Institute of Technology,Kanpur

September 25, 2013

Abstract

Recommendation System can provide suggestions about movies, videos, newspaper articles to a user. For instance, they may predict whether a user would like a movie or not. These systems collects information from a large number of users and mainly use nearest neighbor techniques to provide recommendations. In this project, we have built a movie-recommendation system using MovieLens Database. We have tried building and using several systems which mainly use a Machine Learning approach called Collaborative filtering. We have tried both the approaches, Item Based and User Based Collaborative filtering and compared the results with the already developed systems.

1 Introduction

With the explosion of Internet Technologies and arrival of Websites such as IMDB, user demand has taken a new turn where they are facing with problem of too much choice. There is a lot of information available everywhere but user doesn't know which one to go for. Same is the case with movies, where a user has a large number of choices but he doesn't know which one to go for. In many cases, they select an option which is very similar to a person with same preferences and also very consistent to his/her past preferences. This basic phenomenon can be exploited to build a system which can solve this dilemma of large number of choices for a user. A number of such systems have been developed but none can guarantee that a user agrees with the choice they provide.

Recommendation Systems are special type of systems that combine the knowledge of user's preferences with the knowledge of other similar users to filter the choices and provide most relevant information. Two main approaches of filtering content based approach and collaborative approach. A content based recommendation system uses the user's past history to recommend new items where as a collaborative approach uses the preferences of other people with similar tastes for recommending items to the user. Recommender systems are a useful alternative to search algorithms since they help users discover items they might not have found by themselves.

We have tried with several approaches using Item Based and User Based Collaborative Filtering on the *100k Movielens Database*. We can predict a movie for a particular user and also predict what rating he is going to give to the movie. The systems have been developed in Java and Python and currently uses a simple console based interface. Evaluation is important in assessing the effectiveness of recommendation algorithms. So we have used Root Mean Squared Error and Mean Absolute Error metrics to evaluate our results.

2 Collaborative Filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly. Various Approaches for Collaborative Filtering are as follows:

2.1 Random Recommendation

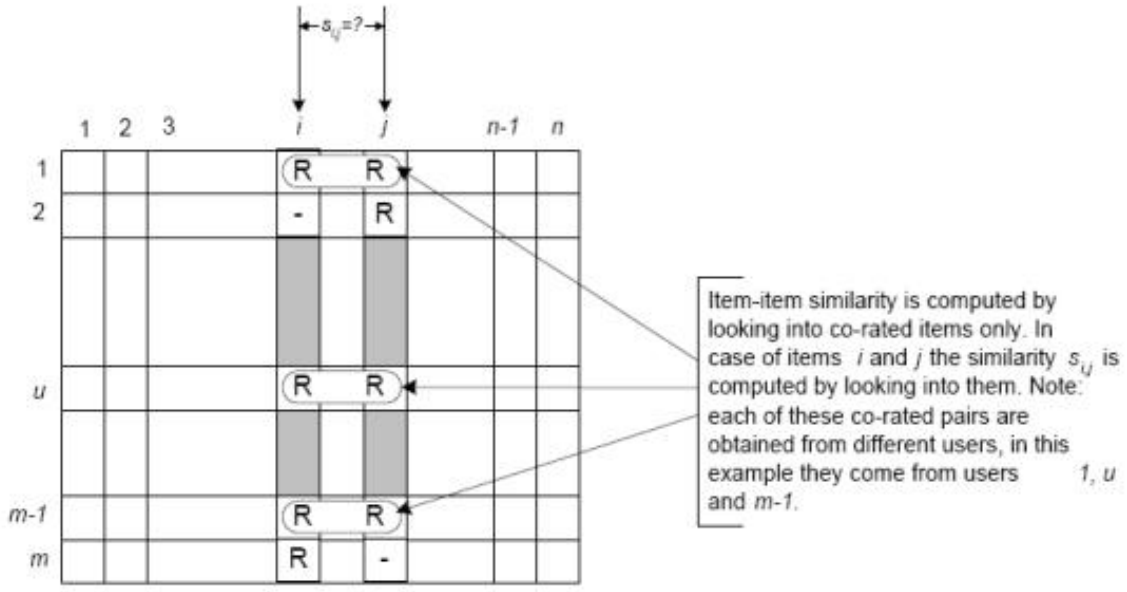
It produces random recommendations and preference estimates. This is likely only useful as a novelty and for benchmarking . It should produce worst result of all the algorithms.

2.2 User-Based Recommendation

User-based Collaborative Filtering (UCF) is one of most popular recommendation technique. Given an unknown test rating (of a test item by a test user) to be estimated, user-based collaborative filtering measures similarities between test user and other users

2.3 Item-Based Recommendation

Item-based collaborative filtering is a model-based algorithm for making recommendations. In the algorithm, the similarities between different items in the dataset are calculated by using one of a number of similarity measures, and then these similarity values are used to predict ratings for user-item pairs not present in the dataset.



2.4 Slope–One Recommendation

It estimates preferences for new items based on average difference in preference value ("diffs") between a new item and the other items the user prefers. The algorithm consists of a significant preprocessing phase, in which all item-item preference value differences are computed:

begin
for every item i
for every other item j
for every user u expressing preference for both i and j
add the difference in u's preference for i and j to an average

And then, the recommendation algorithm becomes:

for every item i the user u expresses no preference for
for every item j that user u expresses a preference for
Find the average preference difference between j and i

add this diff to *u's* preference value for *j*
add this to a running average
return the top items, ranked by these averages

2.5 Item Average Recommendation

A simple recommender that always estimates preference for an item to be the average of all known preference values for that item. No information about users is taken into account. This implementation is provided for experimentation; while simple and fast, it may not produce very good recommendations.

2.6 Item-User Average recommendation

Like Item Average Recommender, except that estimated preferences are adjusted for the users' average preference value. For example, say user X has not rated item Y. Item Y's average preference value is 3.5. User X's average preference value is 4.2, and the average over all preference values is 4.0. User X prefers items 0.2 higher on average, so, the estimated preference for user X, item Y is 3.5 + 0.2 = 3.7.

3 Trade- Off Between User-Based Recommendation and Item-Based Recommendation

The throughput of User-Based Recommendation System decrease as the number of users increases. For Example, it may take a large time predict a value in a large network like internet. But the number of items may be small , thus Item Based Recommendation may work well .Similarly in the case of Item Based Recommendation, it may not be fruitful to calculate the similarity between hundreds of item for example in case of music albums there can be 1000s of song tracks and may be only some users .Thus in this case user based recommendation may work well.

4 Results

4.1 User Based Collaborative Filtering

4.1.1 Pearson Correlation Similarity

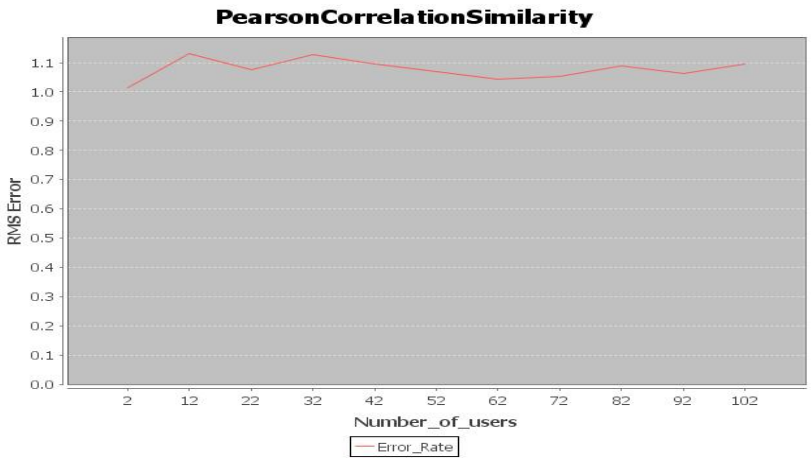


Figure 1: Plot of Number of users vs. RMS Error for Pearson Correlation Coefficient

4.1.2 Euclidean Distance Similarity Measure

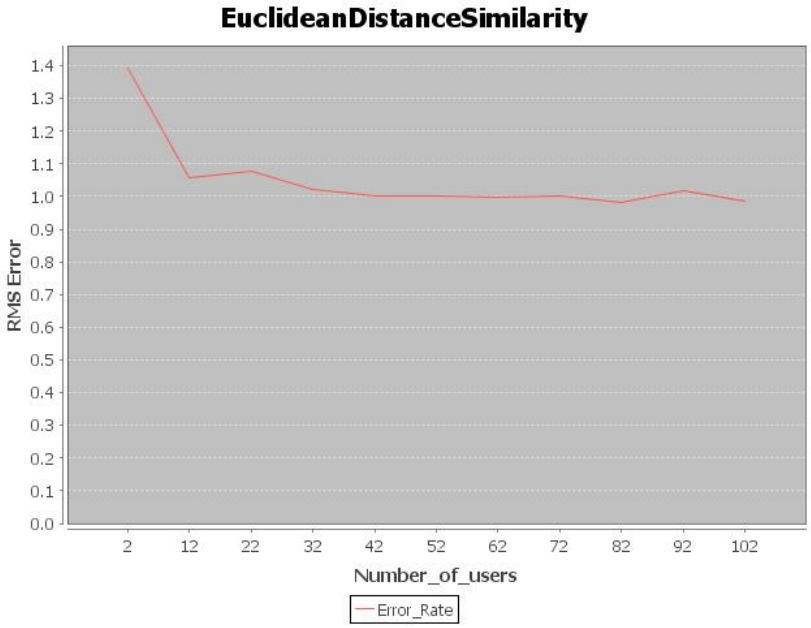


Figure 2: Plot of Number of users vs. RMS Error for Euclidean Distance Similarity

4.1.3 Cosine Distance Similarity Measure

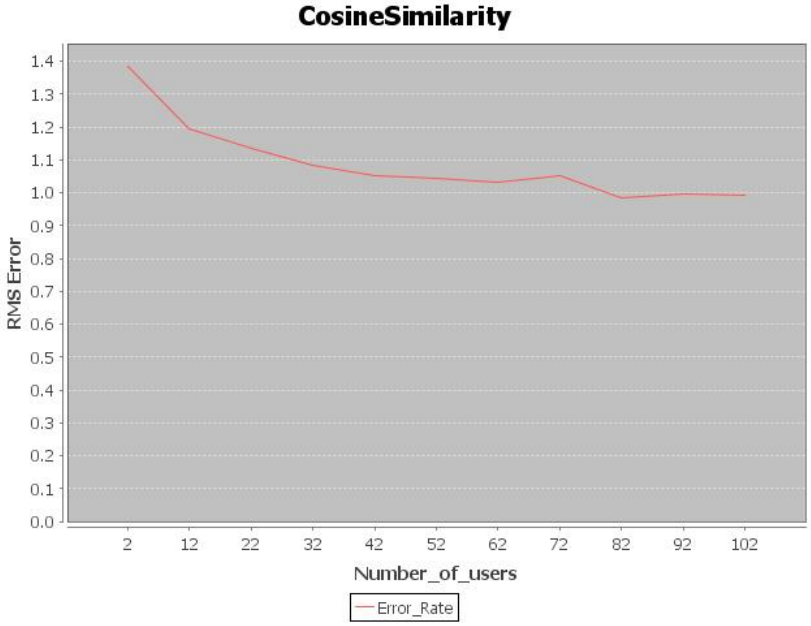


Figure 3: Plot of Number of users vs. RMS Error for Cosine Similarity

4.1.4 k-NN u1base/test to u5base/test

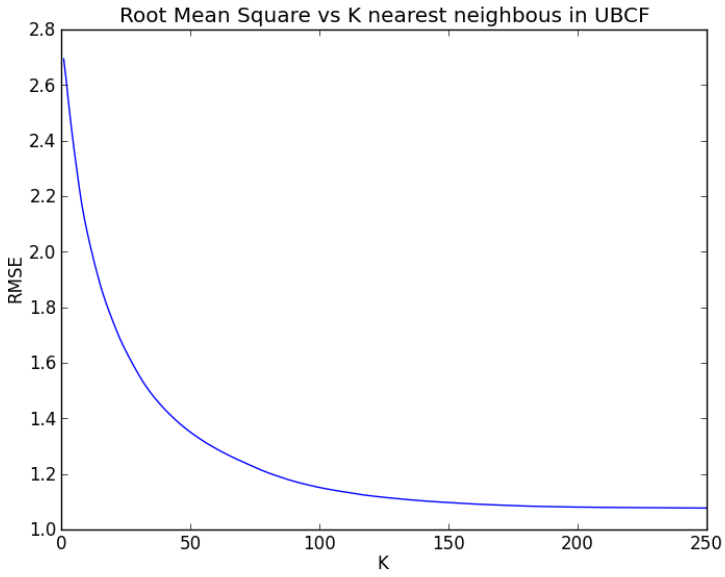


Figure 4: u1base/test

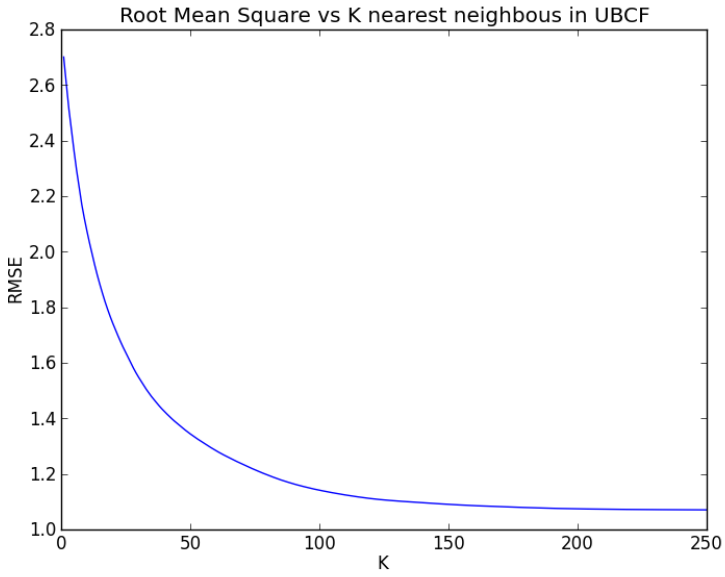


Figure 5: u2base/test

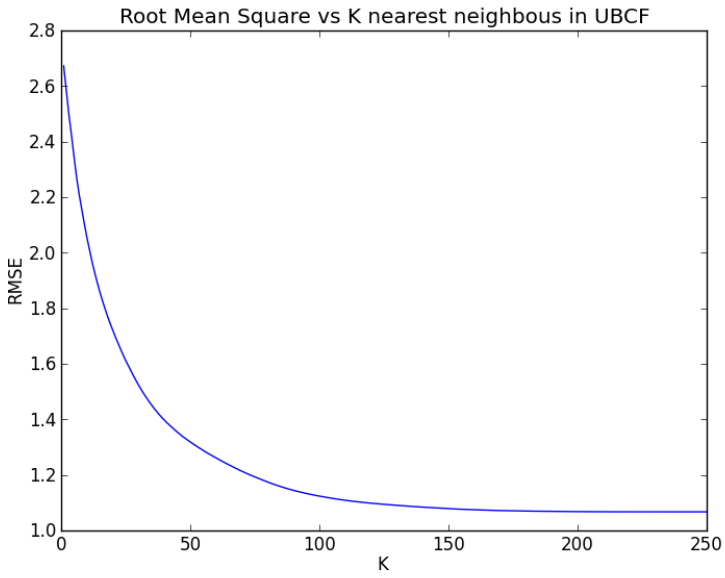


Figure 6: u3base/test

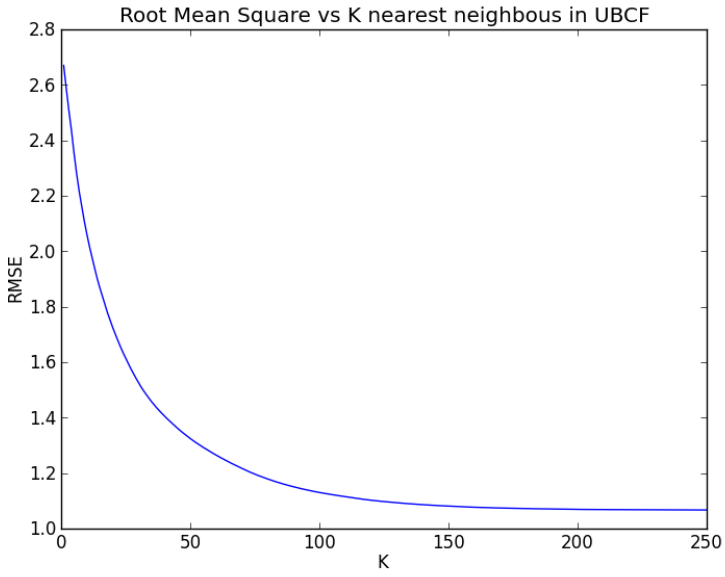


Figure 7: u4base/test

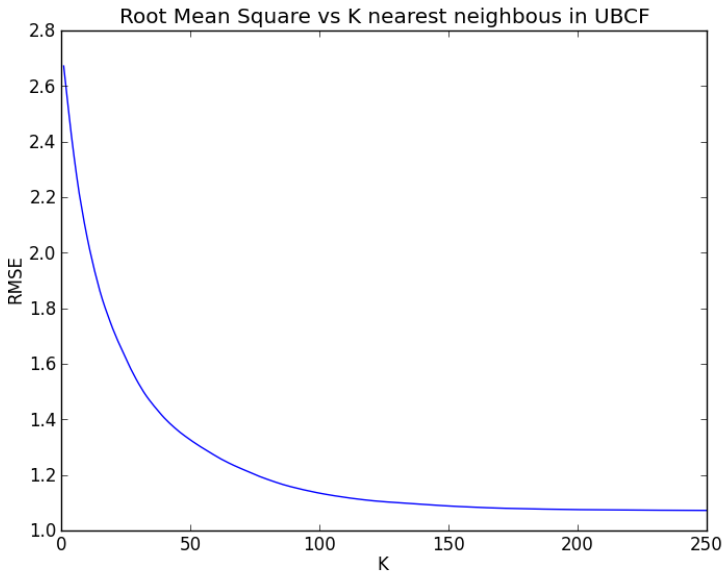


Figure 8: u5base/test

4.2 Item Based Collaborative Filtering

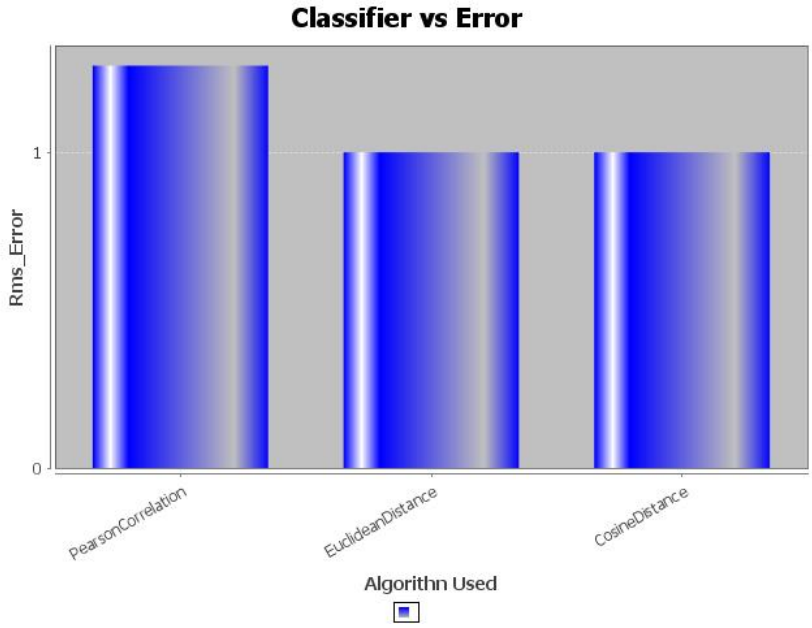


Figure 9: Plot of Result of Item-Based Recommendation based on Different Similarity Techniques

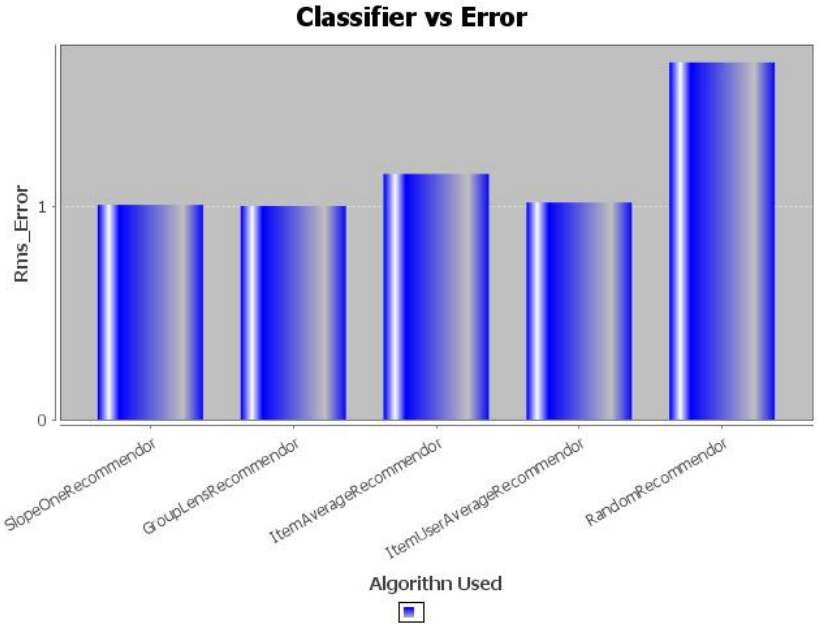


Figure 10: Plot of Result of Different Recommendation Algorithms based

4.3 Table

S.No.	Algorithm Used	RMS Error Found
1	Random Recommender	1.706745107579322
2	GroupLens Recommender	0.9604833884849696
3	Item Average Recommender	1.046489529489477
4	Item User Average Recommender	0.992324821381628
5	Slope One Recommender	0.9596826270668479
6	Item Based with Pearson Correlation Similarity	1.2848360576796047
7	Item Based with Euclidean Distance Similarity	1.0270202156297646
8	Item Based with Cosine Similarity	1.0434303652370585
9	User Based with Euclidean Distance Similarity(k=50)	0.9796846711922585
10	User Based with Euclidean Distance Similarity(k=100)	1.0307653814022009
11	User Based with Pearson Correlation Similarity (k=50)	1.077093120362919
12	User Based with Pearson Correlation Similarity (k=100)	1.0768530493114155
13	User Based with Cosine Similarity (k=50)	1.025580177537051
14	User Based with Cosine Similarity (k=100)	1.018564613113902

5 Conclusion

6 References

- Java Doc-Mahout Core 0.9 API
- A Programmer's Guide to Data Mining ($http://guidetodatamining.com/$)