Movie Recommendation System

1. Introduction

Recommender systems apply data analysis techniques to the problem of helping users find the items they would like to purchase at E-Commerce sites by producing a predicted likeliness score or a list of top-*N* recommended items for a given user. Item recommendations can be made using different methods. Recommendations can be based on demographics of the users, overall top selling items, or past buying habit of users as a predictor of future items. Collaborative Filtering (CF) [[19](http://wwwconference.org/www10/cdrom/papers/519/node37.html#Resnick-94),[27](http://wwwconference.org/www10/cdrom/papers/519/node37.html" \l "Shardanand-95)] is the most successful recommendation technique to date. The basic idea of CF-based algorithms is to provide item recommendations or predictions based on the opinions of other like-minded users. The opinions of users can be obtained explicitly from the users or by using some implicit measures.

2. Collaborative Filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or [taste](http://en.wikipedia.org/wiki/Taste_%28sociology%29) 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.

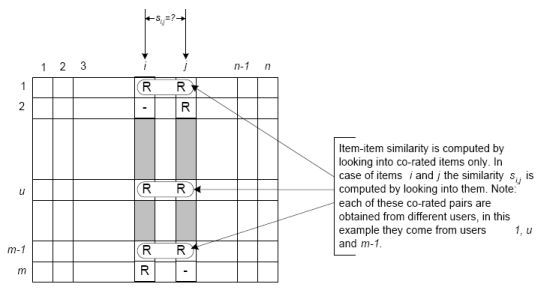
Various Approaches for Collaborative Filtering are as follows-

**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.

**User- Based Recommendation –**

**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 .

**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:  
  
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

Average difference in preference value between all pairs of items.

**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.

**Item-User Average recommendation-**

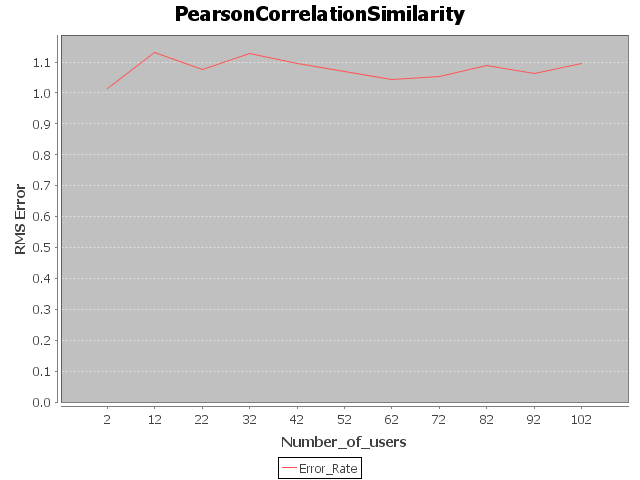
Like ItemAverageRecommender, except that estimated preferences are adjusted for the users' average preference value. For example, say user X has not rated item Y. Item Y's average preference value is 3.5. User X's average preference value is 4.2, and the average over all preference values is 4.0. User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

1. Trade- Off Between User-Based Recommendation and Item-Based Recommendation-

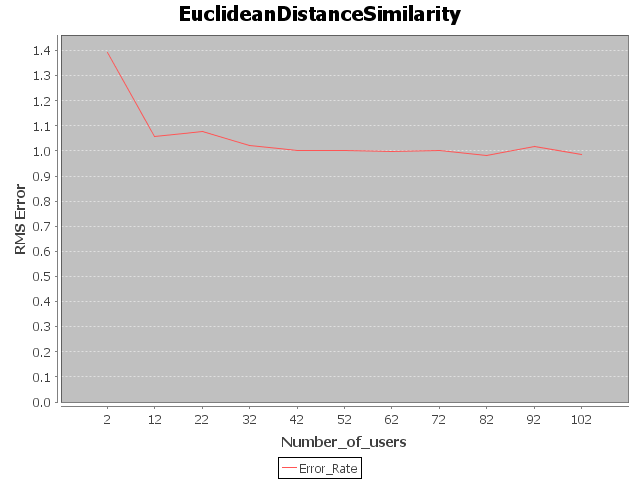
The throughput of User-Based Recommendation System decrease as the number of users increases. For Example, it may take a large time predict a value in a large network like internet. But the number of items may be small , thus Item Based Recommendation may work well .Similarly in the case of Item Based 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.

1. Results –

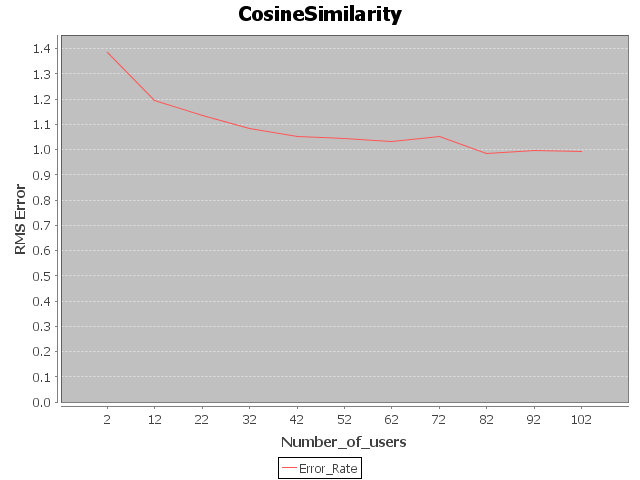
**User- Based Collaborative filtering-**

1. **PearsonCorrelationSimilarity -** 

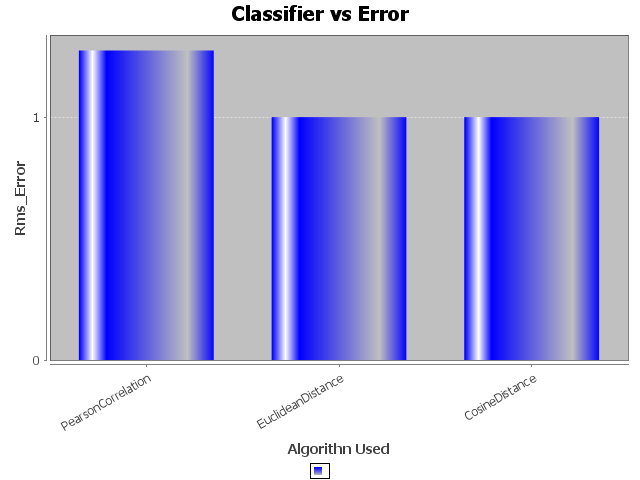
**Plot of Number of users vs. Rms Error for PearsonCorrelation Coefficient**

**2. Euclidean Distance Similarity Measure-**

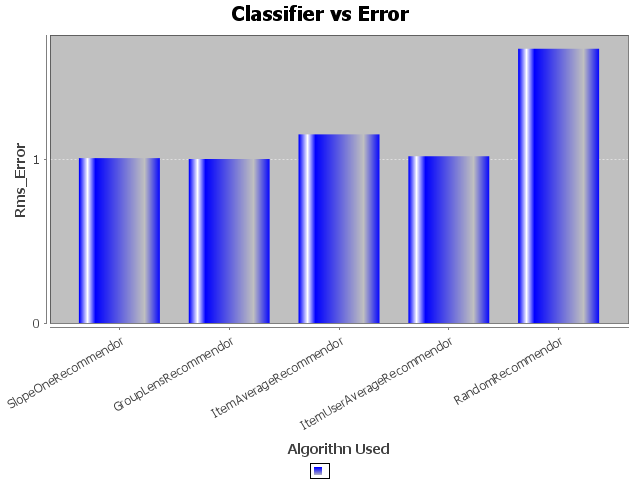
**Plot of Number of users vs. Rms Error for EuclideanDistanceSimilarity**

**3.Cosine Distance Similarity Measure –**

**Plot of Number of users vs. Rms Error for CosineSimilarity**

**Item- Based Collaborative Filtering**

**Plot of Result of Item-Based Recommendation based on Different Similarity Techniques**

**Comparison Between different Recommendation Techniques-**

**Plot of Result of Different Recommendation Algorithms based**

**Result Table-**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Algorithm Used** | **RMS Error Found** |
| 1 | Random Recommender | 1.706745107579322 |
| 2 | GroupLens Recommender | 0.9604833884849696 |
| 3 | ItemAverageRecommender | 1.046489529489477 |
| 4 | ItemUserAverageRecommender | 0.992324821381628 |
| 5 | SlopeOneRecommener | 0.9596826270668479 |
| 6 | Item Based with PearsonCorrelationSimilarity | 1.2848360576796047 |
| 7 | Item Based with EuclideanDistanceSimilarity | 1.027020215629764 |
| 8 | Item Based with CosineSimilarity | 1.0434303652370585 |
| 9 | User Based with EuclideanDistanceSimilarity(k=50) | 0.9796846711922585 |
| 10 | User Based with EuclideanDistanceSimilarity(k=100) | 1.0307653814022009 |
| 11 | User Based with PearsonCorrelationSimilarity (k=50) | 1.077093120362919 |
| 12 | User Based with PearsonCorrelationSimilarity (k=100) | 1.0768530493114155 |
| 13 | User Based with CosineSimilarity (k=50) | 1.025580177537051 |
| 14 | User Based with CosineSimilarity (k=100) | 1.018564613113902 |

**Conclusion-**

**Worst Technique** -**Random Recommendation** Technique perfor

ms the worst among all the techniques.

**Most Popular** – **SlopeOneRecommender** performs the best in this data set.

**References-**

# Java Doc-Mahout Core 0.9 API

# [A Programmer's Guide to Data Mining](http://guidetodatamining.com/)(<http://guidetodatamining.com/>)