Movie Rating Estimation and Recommendation

TLDR;

I build an user based and item based movie rating prediction recommender system based on data provided by MovieLens using memory-based Collaborative filtering (CF) technique, by utilizing following algorithms and metrics:

- Similarity: Pearson correlation, Euclidean distances, and Cosine distances
- Neighborhood: K-nearest neighbors, All neighbors
- Rating Prediction: Weighted Sum Of Other's Rating, Simple Weighted Average
- Analysis: MAE, RMSE, NMAE

Introduction:

In everyday life, people rely on recommendations from other people by spoken words, reference letters, news reports from news media, general surveys, travel guides, and so forth. Recommender systems assist and augment this natural social process to help people sift through available books, articles, web pages, movies, music, restaurants, jokes, grocery products, and so forth to find the most interesting and valuable information for them. The fundamental assumption of Collaborative Filtering is that if users X and Y rate n items similarly, or have similar behaviors (e.g., buying, watching, listening), and hence will rate or act on other items similarly.

CF techniques use a database of preferences for items by users to predict additional topics or products a new user might like. In a typical CF scenario, there is a list of m users $\{u_1, u_2, u_3, ..,u_m\}$ and a list of n items $\{i_1, i_2, i_3,....i_n\}$, and each user, u_i, has a list of items,I_{ui} , which the user has rated, or about which their preferences have been inferred through their behaviors. The ratings can either be explicit indications, and so forth, on a 1–5 scale, or implicit indications, such as purchases or click-throughs.

Data:

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota.

This data set consists of:

* 100,000 ratings (1-5) from **943 users on 1682 movies.**
* Each user has rated at least 20 movies.
* Simple demographic info for the users (age, gender, occupation, zip)

The data was collected through the MovieLens web site (movielens.umn.edu) during the

seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up - users who had less than 20 ratings or did not have complete demographic information were removed from this data set.

User-Movie-Preference:

This is a tab separated list of

user id | item id | rating | timestamp.

196	242	3	881250949
186	302	3	891717742
22	377	1	878887116

Algorithms:

Collaborative Filtering:

It's called collaborative because it makes recommendations based on other people— in effect, people collaborate to come up with recommendations. It works like this. Suppose the task is to recommend a book to you. We search among other users of the site to find one that is similar to you in the books she enjoys. Once we find that similar person we can see what she likes and recommend those books to you.

Memory Based Collaborative Filtering:

Memory-based CF algorithms use the entire or a sample of the user-item database to generate a prediction. Every user is part of a group of people with similar interests. By identifying the so-called neighbors of a new user (or active user), a prediction of preferences on new items for him or her can be produced.

The neighborhood-based CF algorithm, a prevalent memory-based CF algorithm, uses the following steps: calculate the similarity or weight, W_{ij}, which reflects distance, correlation, or weight, between two users or two items, i and j; produce a prediction for the active user by taking the weighted average of all the ratings of the user or item on a certain item or user, or using a simple weighted average. When the task is to generate a top-N recommendation, we need to find k most similar users or items (nearest neighbors) after computing the similarities, then aggregate the neighbors to get the top-N most frequent items as the recommendation.

Similarity Computation:

Similarity computation between items or users is a critical step in memory-based collaborative filtering algorithms. For item-based CF algorithms, the basic idea of the similarity computation between item i and item j is first to work on the users who have rated both of these items and then to apply a similarity computation to determine the similarity, $w_{i,j}$, between the two co-rated items of the users. For a user-based CF algorithm, we first calculate the similarity, $w_{u,v}$, between the users u and v who have both rated the same items.There are many different methods to compute similarity or weight between users or

items.

- **Pearson Correlation:**

  In this case, similarity $w_{u,v}$ between two users u and v, or $w_{i,j}$ between two items i and j, is measured by computing the Pearson correlation or other correlation-based similarities. Pearson correlation measures the extent to which two variables linearly relate with each other. For the user-based algorithm, the Pearson correlation between users u and v is

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}},$$

  where the $i \in$ I summations are over the items that both the users u and v have rated and $\bar{r}_u$ is the average rating of the co-rated items of the uth user.

  For the item-based algorithm, denote the set of users $u \in$ U who rated both items i and j, then the Pearson Correlation will be

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}},$$

  where $r_{u,i}$ is the rating of user u on item i, $\bar{r}_u$ is the average rating of the ith item by those users.

- **Euclidean Distances:**
  The euclidean distances between two points p and q is the length of the line segment connecting them. The distance from p to q, or from q to p is given by:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

- **Cosine Distances:**
  The similarity between two documents can be measured by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the frequency vectors. This formalism can be adopted in collaborative filtering, which uses users or items instead of documents and ratings instead of word frequencies.

  Formally, if R is the $m \times n$ user-item matrix, then the similarity between two items, i and j, is defined as the cosine of the n dimensional vectors corresponding to the i

th and j th column of matrix R.

Vector cosine similarity between items i and j is given by

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|},$$

where "•" denotes the dot-product of the two vectors. To get the desired similarity computation, for n items, an $n \times n$ similarity matrix is computed.

As a side note, most of the analysis is performed using the Pearson correlation. In fact, Pearson correlation performs cosine similarity with some sort of normalization of the user's ratings according to his own rating behavior.

Prediction and Recommendation Computation:

To obtain predictions or recommendations is the most important step in a collaborative filtering system. In the neighborhood-based CF algorithm, a subset of nearest neighbors of the active user are chosen based on their similarity with him or her, and a weighted aggregate of their ratings is used to generate predictions for the active user.

- **Weighted Sum of Other's Rating**:

  To make a prediction for the active user, u , on a certain item, i , we can take a weighted average of all the ratings on that item according to the following formula.

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U}|w_{a,u}|},$$

  where $\bar{r}_a$ and $\bar{r}_u$ are the average ratings for the user a and user u on all other rated items, and $w_{a,u}$ is the weight between the user a and user u. The summations are over all the users $u \in$ U who have rated the item i.

- **Simple Weighted Average**:

  For item-based prediction, we can use the simple weighted average to predict the rating, $P_{u,i}$, for user u on item i

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N}|w_{i,n}|},$$

  where the summations are over all other rated items $n \in$ N for user u, $w_{i,j}$ is the

weight between items i and n, $r_{u,n}$ is the rating for user u on item n.

Top-N Recommendations:

Top-N recommendation is to recommend a set of N top-ranked items that will be of interest to a certain user. For example, if you are a returning customer, when you log into your www.amazon.com account, you may be recommended a list of books (or other products) that may be of your interest. Top-N recommendation techniques analyze the user-item matrix to discover relations between different users or items and use them to compute the recommendations.

User-Based Top-N Recommendation Algorithms:

User-based top-N recommendation algorithms firstly identify the k most similar users (nearest neighbors) to the active user using the Pearson correlation, in which each user is treated as a vector in the m-dimensional item space and the similarities between the active user and other users are computed between the vectors. After the k most similar users have been discovered, their corresponding rows in the user-item matrix R are aggregated to identify a set of items, C, purchased by the group together with their frequency. With the set C, user-based CF techniques then recommend the top-N most frequent items in C that the active user has not purchased. User-based top-recommendation algorithms have limitations related to scalability and real-time performance.

Item-Based Top-N Recommendation Algorithm:

Item-based top-N recommendation algorithms have been developed to address the scalability problem of user-based top-N recommendation algorithms. The algorithms firstly compute the k most similar items for each item according to the similarities; then identify the set, C, as candidates of recommended items by taking the union of the k most similar items and removing each of the items in the set, U, that the user has already purchased; then calculate the similarities between each item of the set C and the set U. The resulting set of the items in C, sorted in decreasing order of the similarity, will be the recommended item-based Top-N list. One problem of this method is, when the joint distribution of a set of items is different from the distributions of the individual items in the set, the above schemes can potentially produce suboptimal recommendations.

Evaluation Metrics:

- **Mean Absolute Error (MAE):**

  Instead of classification accuracy or classification error, the most widely used metric in CF research literature is Mean Absolute Error (MAE), which computes the average of

the absolute difference between the predictions and true ratings

$$MAE = \frac{\sum_{\{i,j\}} |p_{i,j} - r_{i,j}|}{n},$$

where n is the total number of ratings over all users, $p_{i,j}$ is the predicted rating for user i on item j, and $r_{i,j}$ is the actual rating. The lower the MAE, the better the prediction.

- **Root Mean Square Error (RMSE):**

  Root Mean Squared Error (RMSE) is becoming popular partly because it is the Netflix prize metric for movie recommendation performance:

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}} (p_{i,j} - r_{i,j})^2},$$

  where n is the total number of ratings over all users, $p_{i,j}$ is the predicted rating for user i on item j, and $r_{i,j}$ is the actual rating. RMSE amplifies the contributions of the absolute errors between the predictions and the true values.

- **Normalized Mean Absolute Error (NMAE):**

  Normalized Mean Absolute Error normalizes MAE to express errors as percentages of full scale:

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}},$$

  where r_{max} and r_{min} are the upper and lower bounds of the ratings.

Challenges:

• **Sparsity**. In practice, many commercial recommender systems are used to evaluate large item sets (e.g., Amazon.com recommends books and CDnow.com recommends music albums). In these systems, even active users may have purchased well under 1% of the items (1% of 2 million books is 20, 000 books). The user-item matrix used for collaborative filtering will thus be extremely sparse and the performances of the predictions or recommendations of the CF systems are challenged. Sometimes, there are users who only rated 1 or 2 movie, so a recommender based on nearest neighbor may be unable to make any recommendation for those users.

• **Scalability**. Nearest neighbor algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.

• **Cold Start**. The cold start problem occurs when a new user or item has just entered the system, it is difficult to find similar ones because there is not enough information (in some literature, the cold start problem is also called the new user problem or new item problem). New items cannot be recommended until some users rate it, and new users are unlikely given good recommendations because of the lack of their rating or purchase history.

Analysis:

Training Set: 70%
Testing Set: 30%

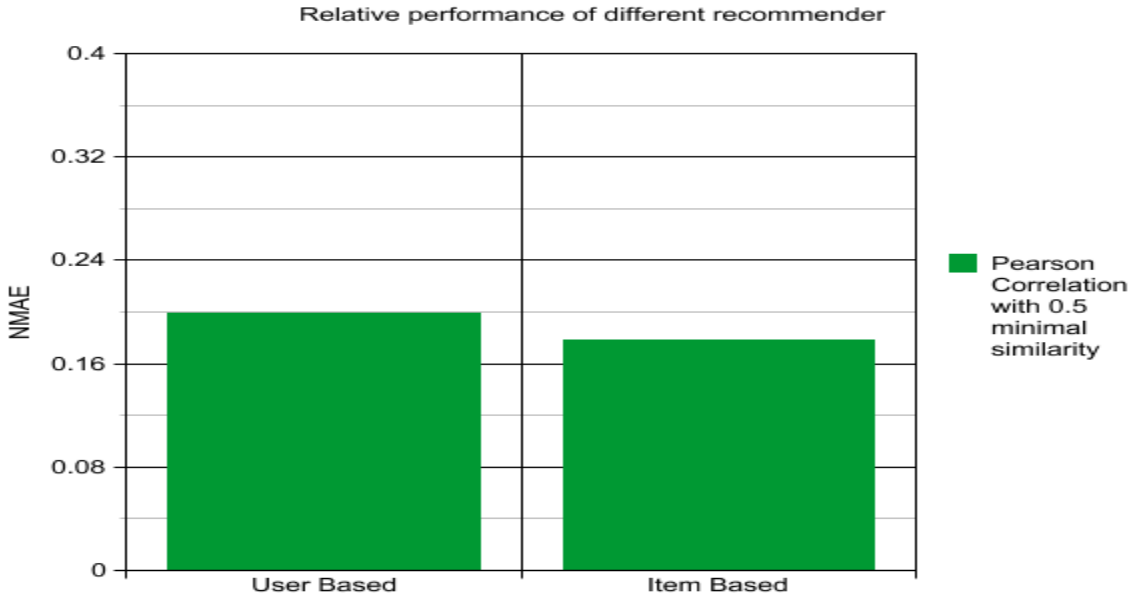
Comparison of User Based and Item Based Recommender:

Evaluation Metrics:

- **Mean Absolute Error (MAE)**

- **Root Mean Square Error (RMSE)**

- **Normalized Mean Absolute Error (NMAE):**



Item based recommender is outperforming user based recommender in all three metrics i.e. mean absolute error (MAE), root mean square error (RMSE), and normalized mean absolute error (NMAE), where similarity is computed using Pearson correlation with 0.5 minimal similarity. Minimal similarity signifies the minimum similarity between users to

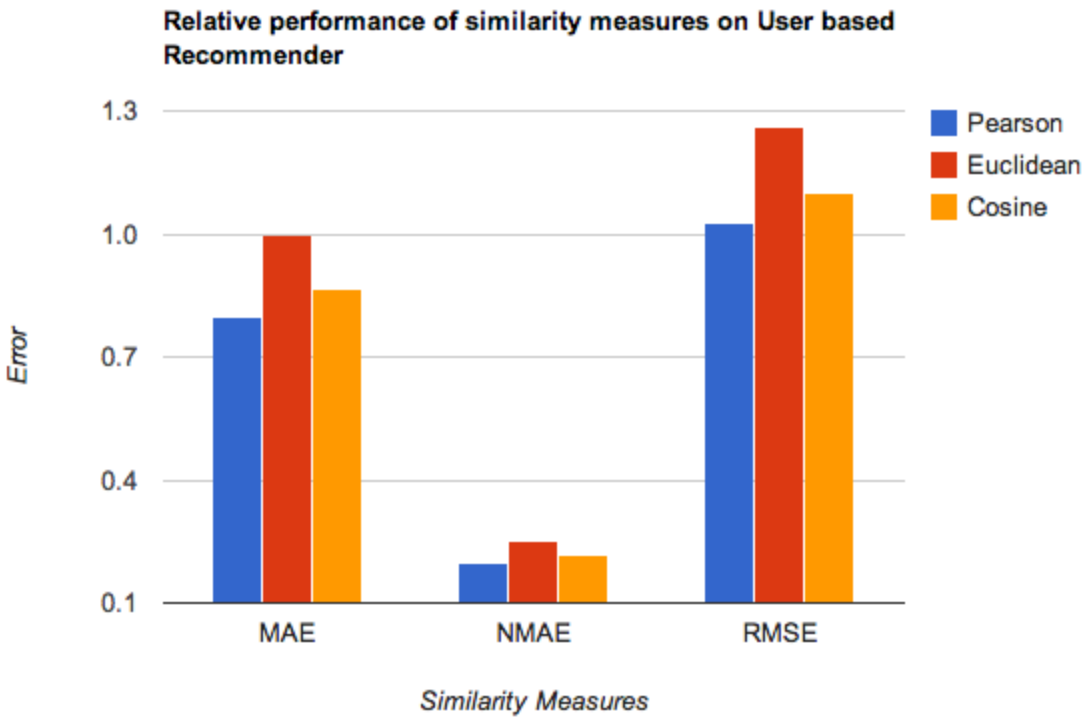
compute the neighborhood. In terms of timing, performance of Item based recommender is better than user based recommender.

Every item is rated by at least 20 users but not every user rated at least 20 items. So this factor comes into play computing the neighborhood and predicting the rating. There are some users who only rated single movie, so it's hard to predict movie for less active users, so it leads to increase in error.

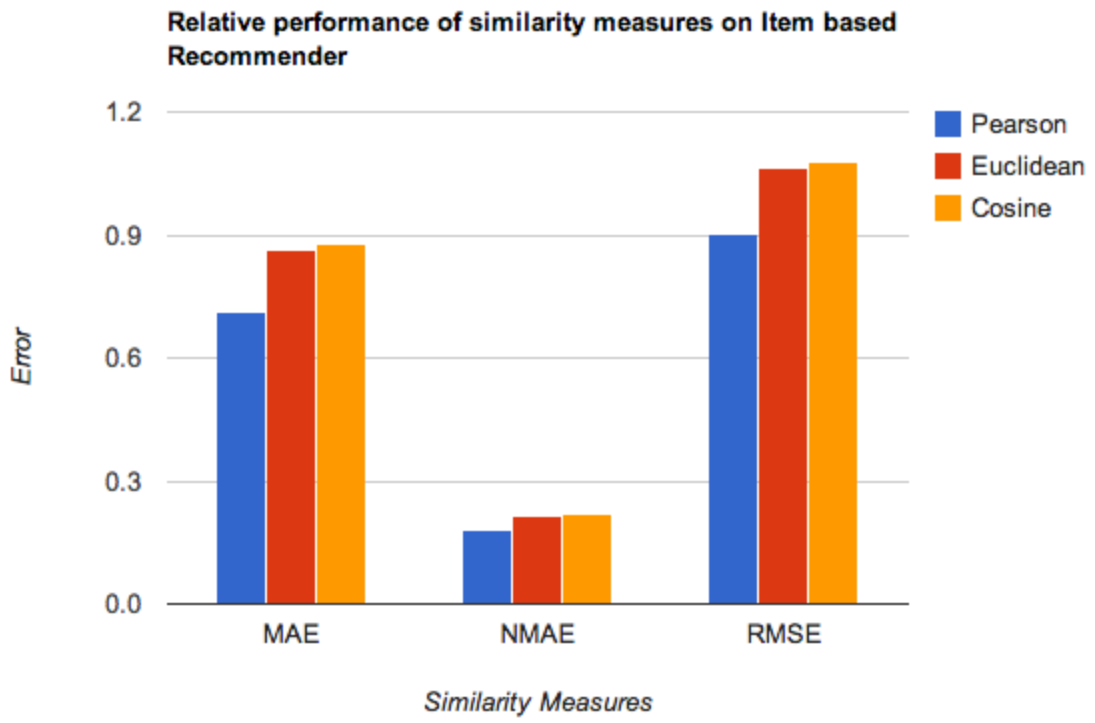
Performance of Similarity Algorithms:

Three different similarity algorithms, Pearson correlation, euclidean distances, and cosine distances was implemented and tested on data set with 70% as training set and 30% as test set. For each similarity algorithm I used K nearest neighbor to compute neighborhood with 0.5 minimal similarity and used weighted sum to predict rating. I ran these experiments on training data and used test data to compute Mean Absolute Error, Normalized Mean Absolute Error, and Root Mean Square Error. It can be observed from the results that Pearson has a clear advantage over euclidean and Cosine.

- Similarity Algorithms on **User** based Recommender

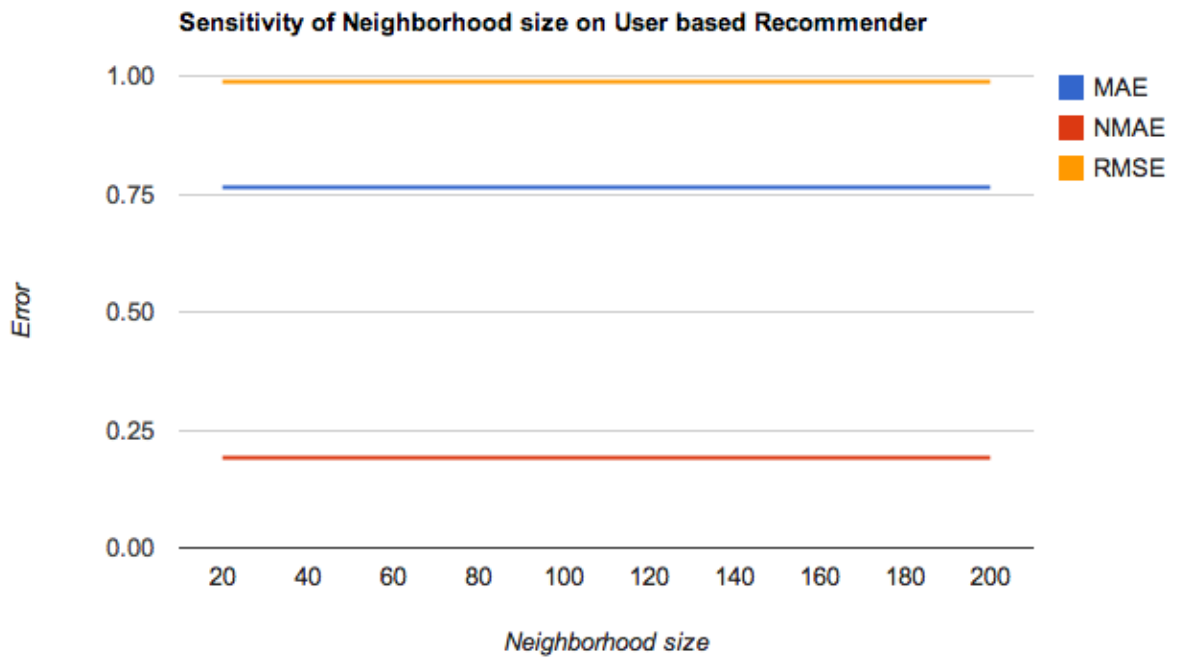


- Similarity Algorithm on **Item** Based Recommender



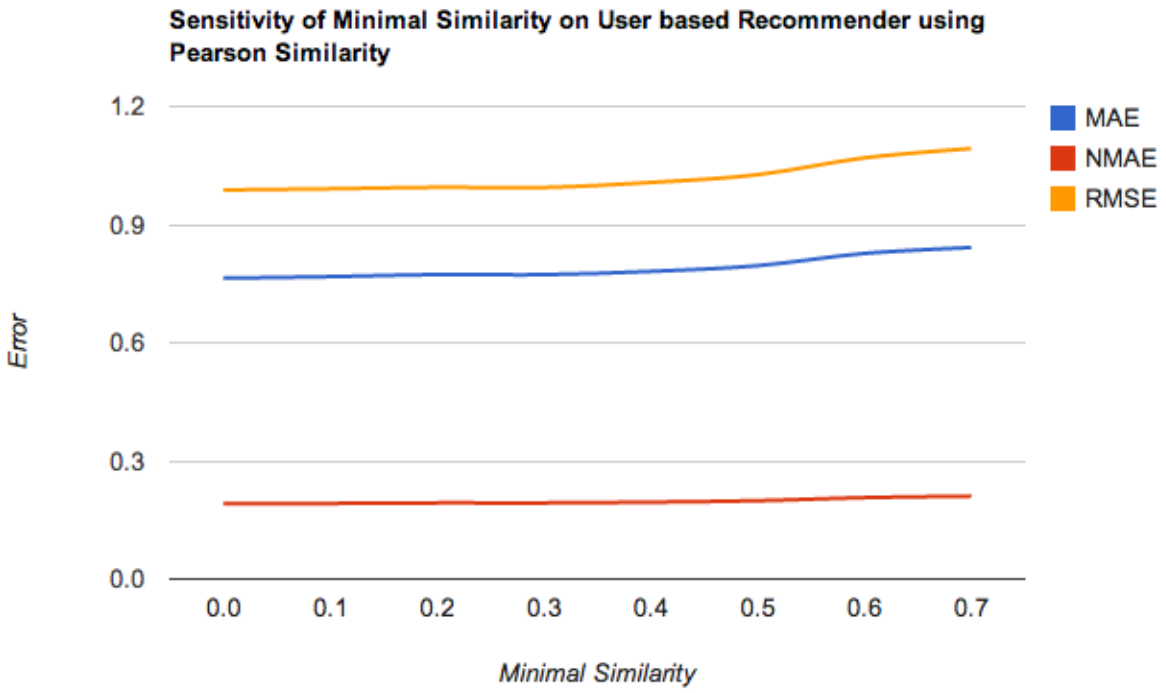
Effect of Neighborhood Size:

The size of the neighborhood has little significant effect on the prediction quality. To determine the sensitivity of this parameter, I performed an experiment where I varied the number of neighborhood to be used and computed MAE, NMAE and RMSE. We can observe that the size of neighborhood doesn't much affect the quality of the prediction but it does affect other metrics like precision and recall.



Sensitivity of Minimal Similarity between User:

I implement pearson algorithm to compute similarity between users and similarity are in range (-1,1). I performed and experiment where I varied the absolute value of similarity and computed MAE, NMAE and RMSE. All these experiments were performed on User based recommendation system with 70% training data and 30% testing set. Two users are neighbors if and only if their similarity value is greater than minimal similarity. Minimal similarity has significant effect on the prediction quality. Similarity is used to compute neighborhood and minimal similarity effect the neighborhood size. The less value of minimum similarity means more number of neighbors. It can be observed from the results that as we increase the minimal similarity, neighbor becomes small and this leads to increase in prediction error.



Conclusion:

It was a great learning experience for me. Recommender systems are a powerful new technology for extracting additional value for business from its user database. These systems help users find what they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Recommender systems are rapidly becoming the crucial tool in E-commerce and many other fields. Recommender systems are being stressed by the huge volume of user data in existing databases and will be stressed even more by the increasing volume of data available on the web.

In future, I would like to try some extensions to memory based algorithms (eg. default voting, inverse user frequency, case amplification, imputation-boosted CF algorithms, weighted majority prediction). I also want to try some model based CF techniques (eg. Bayesian Belief Net CF Algorithm, Clustering CF Algorithm, Regression Based , Latent Semantic CF Algorithms, etc). I'll try to continue working on this project. It would be interesting to compare both memory based and model based recommendation systems. I also want to perform some experiments using precision and recall as evaluation metrics.

References:

[1] Item Based Collaborative Filtering Recommendation Algorithms by Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
[2] A Survey of Collaborative Filtering Techniques by Xiaoyuan Su and Taghi M. Khoshgoftaar
[3] A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience by Matthew R. McLaughlin and Jonathan L. Herlocker
[4] A guide to practical data mining, collective intelligence, and building recommendation systems by Ron Zacharski.
[5] Crab Python Recommendation Framework by Muriçoca Labs
[6] Programming Collective Intelligence book by Toby Segaran
[7] MovieLens data from GroupLens datasets

Special Thanks
Prof. Jay Aslam

Thanks
Mandeep Singh
CS6240