Fall 2015

RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING

Yunkyoung Lee
San Jose State University

RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING

A Thesis

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Yunkyoung Lee

December 2015

The Designated Project Committee Approves the Project Titled

Recommendation System Using Collaborative Filtering

By

Yunkyoung Lee

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE NIVERSITY

December 2015

Dr. T Y Lin	Department of Computer Science
Dr. H. Chris Tseng	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science

ABSTRACT

Recommendation System Using Collaborative Filtering

by Yunkyoung Lee

Collaborative filtering is one of the well known and most extensive techniques in recommendation system its basic idea is to predict which items a user would be interested in based on their preferences. Recommendation systems using collaborative filtering are able to provide an accurate prediction when enough data is provided, because this technique is based on the user's preference. User-based collaborative filtering has been very successful in the past to predict the customer's behavior as the most important part of the recommendation system. However, their widespread use has revealed some real challenges, such as data sparsity and data scalability, with gradually increasing the number of users and items.

To improve the execution time and accuracy of the prediction problem, this paper proposed item-based collaborative filtering applying dimension reduction in a recommendation system. It demonstrates that the proposed approach can achieve better performance and execution time for the recommendation system in terms of existing challenges, according to evaluation metrics using Mean Absolute Error (MAE).

ACKNOWLEDGEMENT

I am very thankful to my advisor Dr. Tsau Young Lin for his continuous guidance and support throughout the project and having firm believe in me. Also, I would like to thank the Committee members Dr. H. Chris Tseng and Dr. Thomas Austin for monitoring the progress of the project and their valuable time.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

E-commerce markets have been restructured into new markets revolving around mobile commerce since the advent of smart devices. User has more opportunity to access diverse information and the amount of information that can be collected has exponentially increased. The immense growth of the World Wide Web has led to an information overload problem. It is difficult for users to quickly obtain what they want from massive information. In recent years, each customer can actively share their review and get a discount based on customer participation such as in social surveys on E-commerce sites. It has become essential for E-commerce markets to effectively take advantage of these data by evolving new marketing strategy based on such data.

Besides, E-commerce markets have actively introduced an automated personalization service to analyze the customer's behavior and patterns as purchase factors. E-commerce sites try to collect various users' interests, such as purchase history, product information in the cart, product ratings, and product reviews in order to recommend new relevant products to customers. Collaborative filtering is the most commonly used algorithm to build personalized recommendations on the website including Amazon, CDNOW, Ebay, Moviefinder, and Netflix beyond academic interest [1, 14].

Collaborative filtering is a technology to recommend items based on similarity. There are two types of collaborative filtering: User-based collaborative filtering and Item-based collaborative filtering [8]. User-based collaborative filtering algorithm is an effective way of recommending useful contents to users by exploiting the intuition that a user will likely prefer the items preferred by similar users. Therefore, at first, the algorithm tries to find the user's neighbors based on user similarities and then combines the neighbor user's rating score by using supervised learning like k-nearest neighbors algorithm and Bayesian network or unsupervised learning like k-means algorithm [8, 9].

Item-based collaborative filtering algorithm fundamentally has the same scheme with user-based collaborative filtering in terms of using user's rating score. Instead of the nearest neighbors, it looks into a set of items; the target user has already rated items and this algorithm computes how similar items are to the target item under recommendation [8, 9]. After that it also combines the customer's previous preferences based on these item similarities.

Collaborative Filtering has been effective in several domains, but their widespread use has revealed some potential challenges, such as rating data sparsity, cold-start, and data scalability [2, 6, 8, 9]. Therefore, to solve the problems of sparsity and scalability in the collaborative filtering, in this paper, I proposed collaborative filtering applying dimension reduction.

The rest of this paper is organized as follows: Chapter 2 summaries the related work and their capabilities and limitation. The proposed approach is

described in Chapter 3. Chapter 4 describes the experimental configuration and evaluation metrics. Experimental results are given in Chapter 5. Finally, Chapter 6 concludes this paper and provides directions for future work.

CHAPTER 2

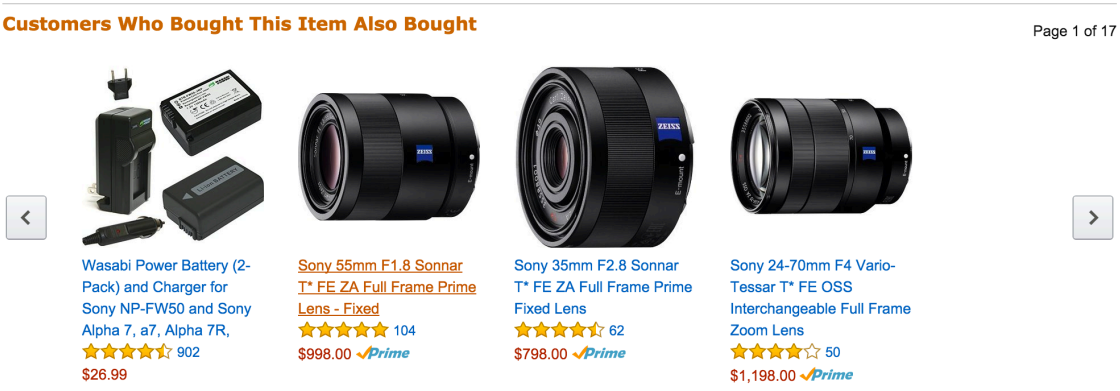
RELATED WORK

Since the advent of the information age, the immense growth of the World Wide Web gives rise to the difficulty for users to quickly find what they want given a variety of applications. Recommendation systems have rigorously been used in various applications as a way to suggest items that a customer would likely be interested in by predicting customer preference. The most popular applications using recommendation systems are movies, music, news, grocery shopping, travel guides, online dating, books, restaurants, E-commerce sites and so forth.

Recommendation systems can be broadly categorized as contents-based filtering, collaborative filtering, and hybrid approach [3]. Contents-based filtering systems are used to recommend items based on a description of items the user used to like before, or corresponding with pre-defined attributes of the user, such a system having its roots in information retrieval techniques. Collaborative filtering systems recommend items to user based on the past preferences of items rated by all users. Hybrid techniques combine both these approaches. In this paper, I will deal mainly with collaborative filtering (CF).

2.1 Collaborative Filtering (CF)

Recommendation systems in various applications have tried to provide users with an accurate recommendation to meet the needs of the user and to bring higher benefits to companies. Collaborative filtering is an effective and well-known technology in recommendation systems. Many web sites, particularly E-commerce sites, have used collaborative filtering technology in their recommendation systems to personalize the browsing experience for each user as seen [Figure 1]. As successful use cases of collaborative filtering, Amazon increased sales by 29% [11], Netflix increased movie rentals by 60% [12], and Google news increased click-through rates by 30.9% [13].

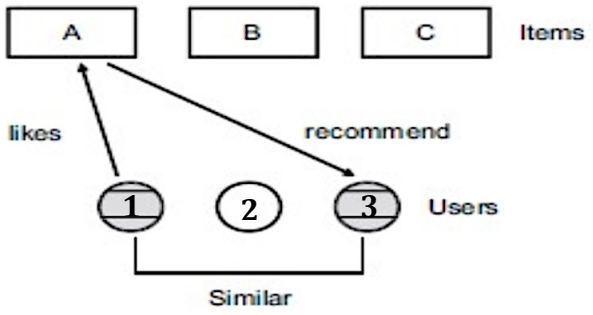


[Figure 1] Shopping cart recommendation at Amazon

Collaborative filtering (CF) can be categorized into two main methods as user-based collaborative filtering (memory-based) and item-based collaborative filtering (model-based) [8].

2.1.1 User-based Collaborative Filtering (UBCF)

User-based collaborative filtering approach is to predict items to the target user that are already items of interest for other users who are similar to the target user. For example, as seen [Figure 2] [15], let User 1 and User 3 have very similar preference behavior. If User 1 likes Item A, UBCF can recommend Item A to User 3. UBCF needs the explicit rating scores of items rated by users [8] to calculate similarities between users and exploits k-nearest neighbor algorithms to find the nearest neighbors based on user similarities. And then, it generates prediction in terms of items by combining the neighbor user's rating scores based on similarity weighted averaging [9].

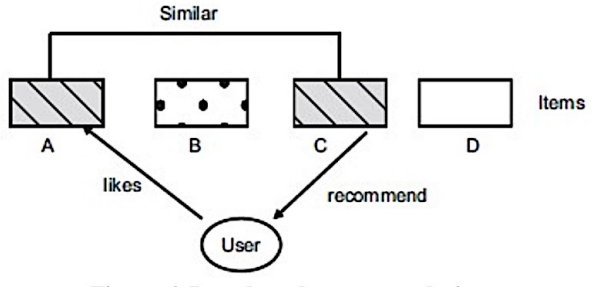


[Figure 2] User-based collaborative filtering

2.1.2 Item-based Collaborative Filtering (IBCF)

Item-based collaborative filtering approach is to predict items by inquiring into similarities between the items and other items that are already associated

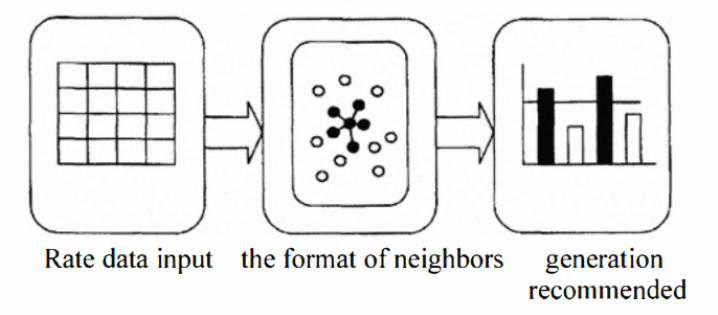
with the user. For example, as seen in [Figure 3] [15], let's say Item A and Item C are very similar. If a User likes Item A, IBCF can recommend Item C to the User. IBCF needs a set of items that the target user has already rated to calculate similarities between items and a target item. And then, it generates prediction in terms of the target item by combining the target user's previous preferences based on these item similarities [9]. In IBCF, users' preference data can be collected in two ways. One is that user explicitly gives rating score to item within a certain numerical scale. The other is that it implicitly analyzes user's purchase records or click-through rate [8].



[Figure 3] Item-based collaborative filtering

2.2 Collaborative Filtering Process

In a fundamental scenario, collaborative filtering (CF) processing can be mainly divided into three steps; Step 1) collecting user ratings data matrix, Step 2) selecting similar neighbors by measuring the rating similarity, and then Step 3) generating prediction as seen diagram [Figure 1] [4, 6, 7, 8, 9].



[Figure 4] The Collaborative filtering process

2.2.1 User Rating Score Data Input

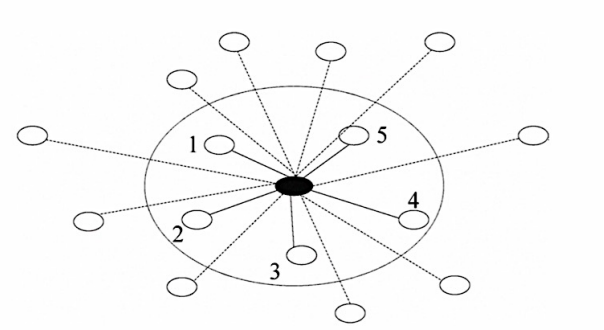
Generally, input data in recommendation system based on the CF technology consists of user, item, and user opinions on observed items as a matrix $m \times n$ as shown in [Table 1]. Symbol *m* symbolizes the total number of users and *n* symbolizes the total number of items. $R_{m,n}$ is the score of item *In* rated by user *Um*.

Item / User	*I1*	*I2*	*I3*	…	*In*
U1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$	…	$R_{1,n}$
U2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$	…	$R_{2,n}$
U3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$	…	$R_{3,n}$
…	…	…	…	…	…
Um	$R_{m,1}$	$R_{m,2}$	$R_{m,3}$	…	$R_{m,n}$

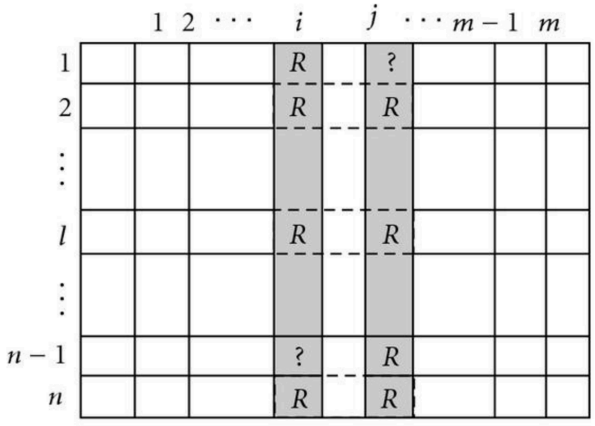
[Table 1] User-Item ratings matrix

2.2.2 The Formation of Neighbors

The CF approaches use statistical techniques to analyze the similarity between users and to form a set of users called neighbors. A set of similarity measures is a metric of relevance between two vectors [9]. User-based similarity is to compute the relevance between users as the values of two vectors. In UBCF, after the similarity is calculated, it is used in building neighborhoods of the current target user. For example, as seen in [Figure 5] [4], the distance between the target node (black node) and every other node is calculated by a similarity measure. And then, 5 users in the center are selected by k-nearest neighbor algorithm (k = 5).



[Figure 5] The neighborhood formation process



[Figure 6] Item based similarity computation

In contrast, it should be noted that IBCF does not form neighborhoods after the similarity is calculated. This is because IBCF already begins computing the similarity between co-rated items only as the value of two vectors [9,14]. For example, as seen in [Figure 6] [2], this item-based similarity is calculated by looking into Item *i* and Item *j* rated by User *2, l,* and *n*. Each of these pairs are given by different users. This is a similar process to the formation of neighbors in UBCF.

Since the similarity measure plays a significant role in improving accuracy in prediction algorithms, it can be effectively used to balance the ratings significance [9]. There are a couple of popular similarity algorithms that have been used in the CF recommendation algorithms [8]. In this paper, I present four similarity methods; Cosine vector similarity, Pearson correlation, Euclidean distance similarity, and Tanimoto coefficient [14, 16].

2.2.2.1 Cosine Vector Similarity

Cosine vector similarity is one of the popular metrics in statistics. Since it notionally considers only the angle of two vectors without the magnitude, it is a very useful measurement with data missing preference information as long as it can count the number of times that term appears in the data [17].

In the following formula, the cosine vector similarity looks into the angle between two vectors (the target Item *i* and the other Item *j*) of ratings in n-dimensional item space. $R_{k,i}$ is the rating of the target Item *i* by User *k*. $R_{k,j}$ is the rating of the other Item *j* by user *k*. *n* is the total number of all rating users to Item *i* and Item *j*.

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \times \vec{j}}{\|\vec{i}\|^2 \times \|\vec{j}\|^2} = \frac{\sum_{k=1}^{n} R_{k,i} R_{k,j}}{\sqrt{\sum_{k=1}^{n} {R_{k,i}}^2 \sum_{k=1}^{n} {R_{k,j}}^2}}$$

When the angle between two vectors is near 0 degree (they are in the same direction), Cosine similarity value, *sim(i,j)*, is 1, meaning very similar. When the angle between two vectors is near 90 degree, *sim(i,j)* is 0, meaning irrelevant. When the angle between two vectors is near 180 degree (they are in the opposite direction), *sim(i,j)* is -1, meaning very dissimilar. In case of information retrieval using CF, *sim(i,j)* ranges from 0 to 1. This is because the angle between two term frequency vectors cannot be greater than 90 degrees [17].

2.2.2.2 Pearson Correlation Coefficient

Pearson correlation coefficient is one of the popularly used methods in CF to measure how larger a number in one series is, relative to the corresponding number. As following formula shows, it is used to measure the linear correlation between two vectors (Item *i* and Item *j*).

$$sim(i,j) = \frac{\langle R_{k,i} - A_i,\ R_{k,j} - A_j \rangle}{\left\| R_{k,i} - A_i \right\| \left\| R_{k,j} - A_j \right\|} = \frac{\sum_{k=1}^{n} (R_{k,i} - A_i)\,(R_{k,j} - A_j)}{\sqrt{\sum_{k=1}^{n} (R_{k,i} - A_i)^2 \times \sum_{k=1}^{n} (R_{k,j} - A_j)^2}}$$

It measures the tendency of two series of numbers, paired up one-to-one, to move together [14]. When two vectors have a high tendency, the correlation, $sim(i,j)$, is close to 1. When two vectors have a low tendency, $sim(i,j)$ is close to 0. When two vectors have opposite tendency, $sim(i,j)$ is close to -1. As mentioned above in [Figure 6], item-based similarity is computed with the co-rated items where users rated both (Item *i* and Item *j*).

$R_{k,i}$ is the rating of the target Item *i* given by User *k*. $R_{k,j}$ is the rating of the other Item j given by User *i*. A_i is the average rating of the target Item *i* for all the co-rated users, and A_j is the average rating of the other Item *j* for all the co-rated users. *n* is the total number of ratings users gave to Item *i* and Item *j*.

2.2.2.3 Euclidean Distance Similarity

Euclidean distance method is based on the distance between items. It forms coordinates to put preference values between items and measures Euclidean distance between each point. When distance value between two

points, $sim(i,j)$, is large, it means the two points are not similar. When $sim(i,j)$ is small, it means two points are similar. This is Euclidean distance formula is given below.

$$sim(i,j) = \sqrt{\sum_{k=1}^{n} (R_{k,i} - R_{k,i})^2}$$

$R_{k,i}$ is the ratings of the target Item *i* given by User *k*. $R_{k,j}$ is the ratings of the other Item j given by User *k*. *n* is the total number of rating users to Item *i* and Item *j*.

2.2.2.4 Tanimoto Coefficient

Tanimoto coefficient is known as the Jaccard similarity coefficient. It does not take into account preference values of an item rated by a user. It only considers if users express a preference. Tanimoto coefficient is the ratio of the size of the intersection, or overlap, in two users' preferred items, to the union of users' preferred items [14]. When two items are completely overlapped, Tanimoto coefficient, $sim(i,j)$, is 1. When two items are not completely overlapped, $sim(i,j)$ is 0.

As shown in the following formula, it examines the overlapped degree between two sets to compare the similarity and diversity of two sets. f_i is a set of Item *i* for which users express preference. f_j is a set of Item *j* for which users

express preference. $f_i \cap f_j$ is intersection of Item *i* and Item *j* for items where preference is expressed by users.

$$sim(i,j) = \frac{\left|f_i \cap f_j\right|}{|f_i| + \left|f_j\right| - \left|f_i \cap f_j\right|}$$

In fact, many users tend not to rate items or recommendation system might not have enough users' information. This metric would be helpful to compute similarity as long as at least preference information as Boolean type is available.

2.2.3 Prediction generation

Once CF computes the similarity between users (in UBCF) or items (in IBCF) and then finds the set of most similar user or similar items, it generates prediction of the target user's interest as the most significant step in CF.

2.2.3.1 Prediction Computation of UBCF

Since UBCF gets the neighborhood of user, UBCF can calculate the predictive rating for the target User *u* on the target Item *i.* It is scaled by the weighted average of all neighbors' ratings on the target Item *i* as following [2, 4]:

$$P_{u,i} = A_u + \frac{\sum_{w=1}^{n}\left(R_{w,i} - A_w\right) \times sim(u,w)}{\sum_{w=1}^{n} sim(u,w)}$$

A_u is the average ratings of the target User *u* to all other rated items and A_w is the average ratings of the neighbor User *w* to all other rated items. $R_{w,i}$ is

the rating of the neighbor User *w* to the target item *i*. $sim(u,w)$ is the similarity of the target User *u* and the neighbor User *w*. And *n* is the total number of neighbors.

2.2.3.2 Prediction Computation of IBCF

Since IBCF has got the neighborhood of items, IBCF tries to make sure how the target user rates similar items. To check if the prediction is in the predefined range [8], the predictive rating for the target User *u* on the target Item *i* is scaled by the weighted average of all neighbor items' ratings given by the target User *u* according to the following formula [8, 10].

$$P_{u,i} = \frac{\sum_{j=1}^{n} R_{u,j} \times sim(i,j)}{\sum_{j=1}^{n} sim(i,j)}$$

$R_{u,i}$ is the rating of the target User *u* to the target Item *i*. $sim(i,j)$ is the weighted similarity of the target Item *i* and the neighbor Item *j*, *n* is the total number of neighbor items.

2.3 Existing Limitations of Collaborative Filtering

Since the number of users and items in each application has steadily increased at the same time as the growth of World Wide Web, collected input data has been a big problem in producing an accurate prediction and in running

recommendation system using collaborative filtering. There are two main challenges in user-based collaborative filtering [8, 9, 10, 15].

2.3.1 Data Sparsity

User-based collaborative filtering depends on explicit feedback, such as ratings given by user to item. User-item input data matrix could have a few rating scores of the total number of items available, even though users are very active. In addition, because users tend not to rate actively, calculating similarity over co-rated set of items could be a challenge. These problems give rise to inaccurate performance of the recommendation system.

Even the cold-start problem is caused by the data sparsity. Collaborative Filtering predicts items based on user's previous preference behavior. That is, it could not predict recommendable items to new users unless new users rate many items. Also, new items could be considered for recommendation, because they have less rating scores by a sufficient number of users.

2.3.2 Data Scalability

For over millions of users and millions of items in user-item input data matrix, the nearest neighbor algorithm is required for high scalability of computation between users as the values of two vectors. Also, recommendation systems could not quickly react to online requirements and immediately make recommendations as it was a time-consuming job.

CHAPTER 3

Item Based Collaborative Filtering Applying Dimension Reduction

UBCF is easy to implement and good to scale correlated items [2]. However, as stated previously above, it comes up against a couple of problems: data sparsity and data scalability. Data sparsity problem could lead to a skewed prediction and low reliability of predictions. Besides, data scalability requires low operation time and high memory feature to scale with all users and items in the database.

To address these issues in UBCF, this paper proposes IBCF approach applying dimension reduction [6].

3.1 IBCF Applying Dimension Reduction

Enormous users and products have been added at E-commerce domains. A typical example is Amazon. Amazon added 30 million new customers in 2013 and had had over 244 million active customers as Geekwire reported in 2014 [18]. Also, Amazon had sold over 200 million products as ReportX reported in 2013 [19]. Currently in 2015, it is expected that Amazon would have more than these numbers of users and products. If the recommendation system using UBCF at

Amazon should look into all datasets similar to a 244 million × 200 million matrix, it will encounter data scalability and data sparsity issues. In UBCF, more the number of users and items increase, more the number of matrix dimensions increase and runtime takes long to find nearest neighbor of users. Therefore, it is assumed that using denser data having much more preference information given by users with IBCF effectively addresses data scalability and data sparsity problems. To focus on active items assuming that they have many ratings given by users, matrix is required to reduce dimension in IBCF without regard to passive items.

Item / User	*I1*	*I2*	*I3*	*I4*	*I5*
U1	2.0	4.0	3.0	3.0	3.0
U2			1.0	3.0	
U3	5.0	1.0	5.0	5.0	
U4	4.0		3.0		

[Table 2] User-Item matrix before dimension reduction

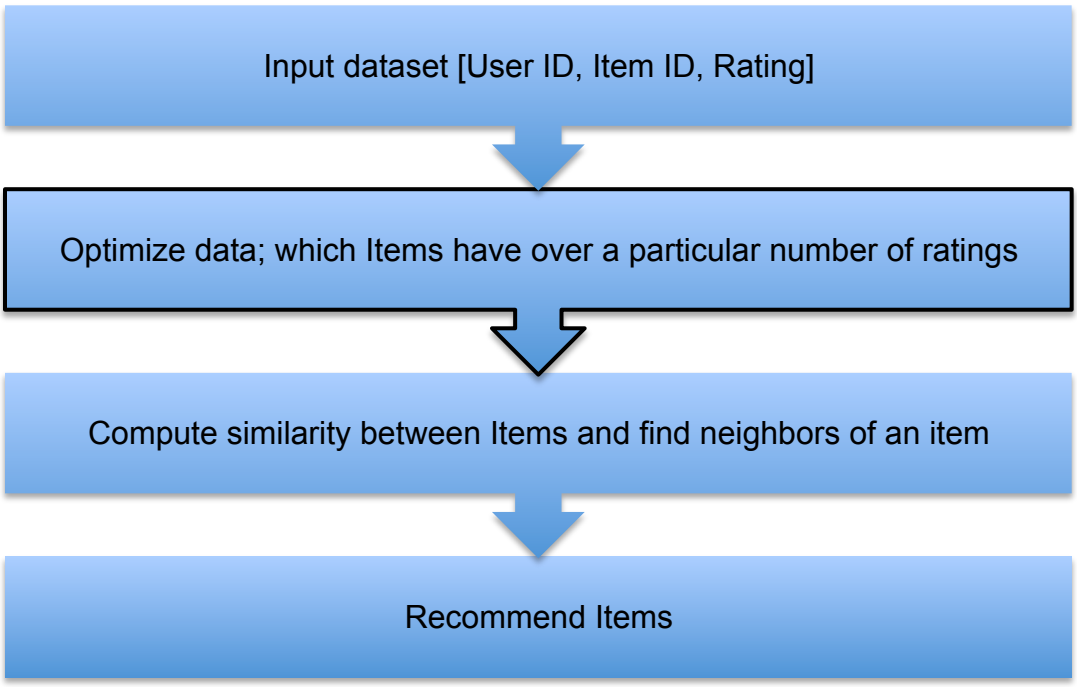
Item / User	*I1*	*I3*	*I4*
U1	2.0	3.0	3.0
U2		1.0	3.0
U3	5.0	5.0	5.0
U4	4.0	3.0	

[Table 3] User-Item matrix after dimension reduction

For instance, as seen in [Table 2], each item can get up to a maximum of 4 ratings by users. Item I2 has 2 ratings and Item I5 has 1 rating, which means the number of ratings for Item I2 and Item I5 is not bigger than half of the total number of ratings. We can assume that Item I2 and Item I5 do not carry much weight with this matrix. Hence, when matrix has impactful items like Item I1, Item I3, and Item I4 as seen in [Table 3], running time of the recommendation system in computing similarity between items and to provide more accurate prediction is expected to reduce.

3.2 Architecture of IBCF Applying Dimension Reduction

Here is a scenario of IBCF applying dimension reduction as seen in [Figure 7]. This is mainly divided into four steps. This approach is based on general collaborative filtering algorithm. To compute similarity between items, the algorithm uses an optimized data by reducing dimention of items that have the number of ratings less than a specific value. For example, if it needs to consider items that have over 20 ratings from users, it extracts data in terms of items having over 20 ratings. In other words, such items are rated by over 20 users.



[Figure 7] Diagram of IBCF applying dimension reduction

CHAPTER 4

Experiments And Evaluation Metrics

In this section, I describe dataset, evaluation metrics, and methodology to optimize data by reducing dimension based on Apache Mahout.

4.1 Experiments Dataset

The data used in this experiment is the MovieLens 1m datasets by GroupLens Research [20]. It contains 1,000,000 ratings by 6040 users on 3952 movies. Each user has rated at least 20 movies [21]. The range of ratings is from 1 (less interesting) to 5 (very interesting) as integer type.

These are a few parts of MovieLen dataset. It consists of user ID, Item ID and Rating as seen in [Table 4]. I consider it as the User-Item matrix as seen in [Table 5].

User ID	*Item ID*	*Rating*
1	1035	5
1	1287	5
1	3408	4
6	1035	5
6	1380	5
6	3408	5
10	1035	5
10	1380	5
10	1287	3
10	3408	4
10	1201	2
26	1035	2
26	1380	4
26	3408	2
26	1201	2
…	…	…

[Table 4] Raw dataset of MovieLens

Item / *User*	*1035*	*1380*	*1287*	*3408*	*1201*	*…*
1	5		5	4		…
6	5	5		5		…
10	5	5	3	4	2	…
26	2	4		2	2	…
…	…	…	…	…	…	…

[Table 5] User-Item Matrix by raw dataset

4.2 Performance Evaluation Criteria

To evaluate the accuracy of a recommendation system, I use statistical accuracy metrics. Mean Absolute Error (MAE) is a widely used metric in the

recommendation system using collaborative filtering to measure the deviation of recommendations from their true user actual ratings.

In the following formula, *N* is the total number of actual ratings in an item set. p_i is the prediction of user's ratings. q_i is corresponding real ratings data set of users.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

It computes the average of the absolute difference between N pairs; prediction scores of users' ratings and actual user ratings for the user-item pairs in the test dataset [2]. Lower the MAE value, better is the recommendation system's accuracy of prediction of user ratings.

4.3 Experiment Environment

- Processor: 2.6 GHz Intel Core i5
- Memory: 8 GB 1600 MHz DDR3
- Operation System: OS X El Capitan Version 10.11.1
- Language: Java
- Platform: Apache Mahout with a pseudo-distributed mode in Apache Hadoop

4.4 Architecture of Apache Mahout

To implement this experiment, I used Apache Mahout, widely used in recommendation system using collaborative filtering. Apache Mahout is one of the most powerful open source platforms in supporting scalable machine learning and distributing processing of a large dataset cluster of computers using the Apache Hadoop system required for recommender development [15]. Developer can effectively customize recommendation system with a rich set of modules abstractions provided by Apache Mahout. There are five key abstractions to define the Mahout interface shown in [Figure 8] [15].

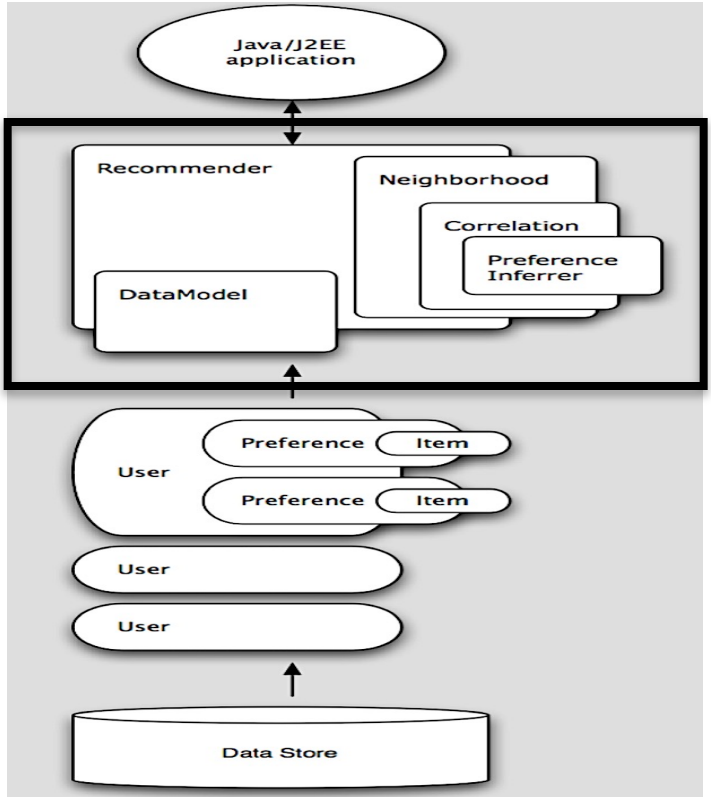
1) *DataModel* is the interface to repository about users and their associated preferences from any source. Users and items are identified solely by an ID value as numeric type [14, 15]. A *GernericPreference* object encapsulates the relation between an item and preference score. A *GenericUserPreference* object stores preferences for all users. A *PreferenceArray* object encapsulates the relation between item and preferred items by users.

2) *UserSimilarity* measures similarity between users used in UBCF.

3) *ItemSmilarity* measures similarity between items used in IBCF.

4) *UserNeighborhood* finds K-nearest neighborhood of similar users near a target user in UBCF. Since IBCF begins with a list of a user's preferred items, it does not need to find K-nearest neighborhood of items.

5) *Recommender* provides items with a target user given a *DataModel.*



[Figure 8] Apache Mahout architecture

4.5 Algorithm of IBCF Applying Dimension Reduction

Input: A: MovieLens dataset “UserID,ItemID,Rating,”

U: a target user U,

I: Item what user is interested,

M: Minimum ratings,

N: the number of items to recommend

1) Parse raw dataset and count the number of ratings per item

```
FastByIdMap C is a map of the total number of ratings per item.
ArrayList D is a list of all ratings.
For each line L in A Do
    Array P ← Convert L to [UserID, ItemID, Rating]
    Integer S = the number of ratings per item
    If C(P[ItemID]) exists Then
         C ← <P[ItemID], S+1 >
    Else
         C ← <P[ItemID], 1 >
    D ← P
```

2) Create optimized data structure by checking if each item's total number of ratings is bigger than M.

```
FastByIdMap O = <P[UserID], R>
For each P in D Do
    If (C(P[ItemID]) > M) Then
        ArrayList<GenericPreference> R
           ← GenericPreference (P[UserID], P[ItemID], P[Rating])
    O ← <P[UserID], R>
```

3) Convert users' ArrayList<GenericPreference> to GenericUserPreferenceArray

```
For each <P[UserID], R> in O Do
    FastByIdMap X ← <P[UserID], GenericUserPreferenceArray(R)>
```

4) Create W = GenericDataModel (X)

5) Compute Similarity between I and other items in W

6) Create GenericItemBasedRecommender

7) Select the most similar N items against I

Output: N recommended ItemIDs for U

4.6 Benchmark UBCF

To compare the performance of recommendation system using IBCF, I also implemented UBCF with Apache Mahout.

CHAPTER 5

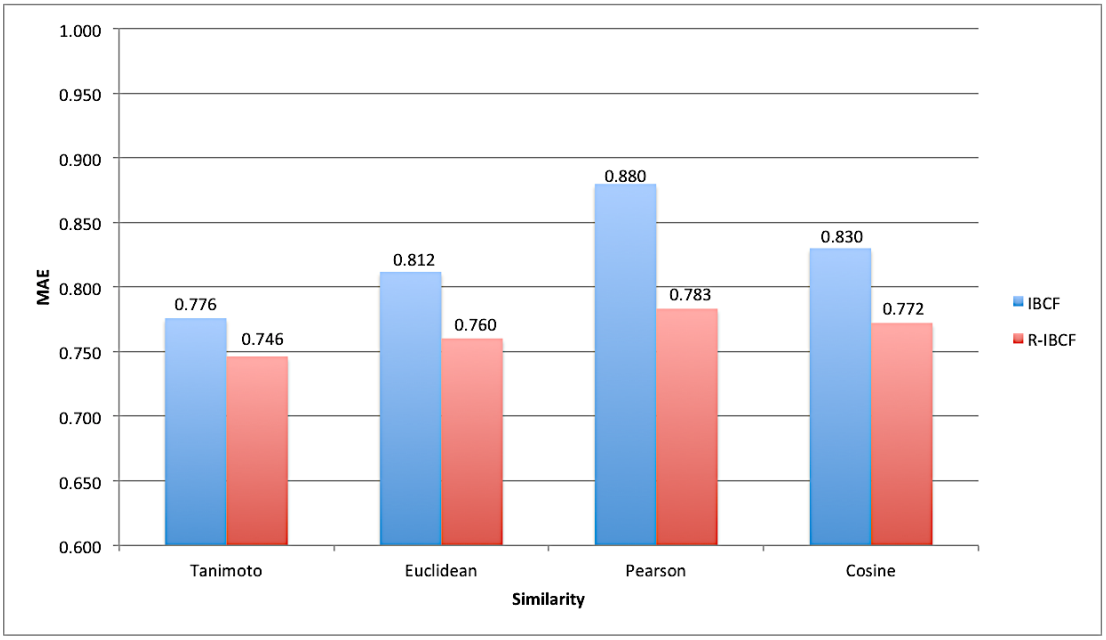
PERFORMACE RESULTS

In this chapter, I implemented item-based collaborative filtering applying dimension reduction (R-IBCF). The goal of the proposed R-IBCF is to provide better quality of prediction in terms of the MAE measure and to make faster execution time. I compared the R-IBCF algorithm to IBCF in order to find an optimal similarity algorithm and training/test ratio of the dataset. Also, I selected an optimal value of the number of ratings per item on R-IBCF as I varied the value of it.

In addition, I implemented UBCF as a benchmark to compare runtime of R-IBCF and IBCF to UBCF and the quality of prediction with optimal parameters.

5.1 Optimum Similarity Measurement

I implemented four different similarity measurements: Cosine vector similarity, Pearson correlation coefficient, Euclidean distance, and Tanimoto coefficient as described in Section 2.2.2. For each similarity algorithms, I measured MAE to find an optimal similarity on IBCF and R-IBCF for this dataset.



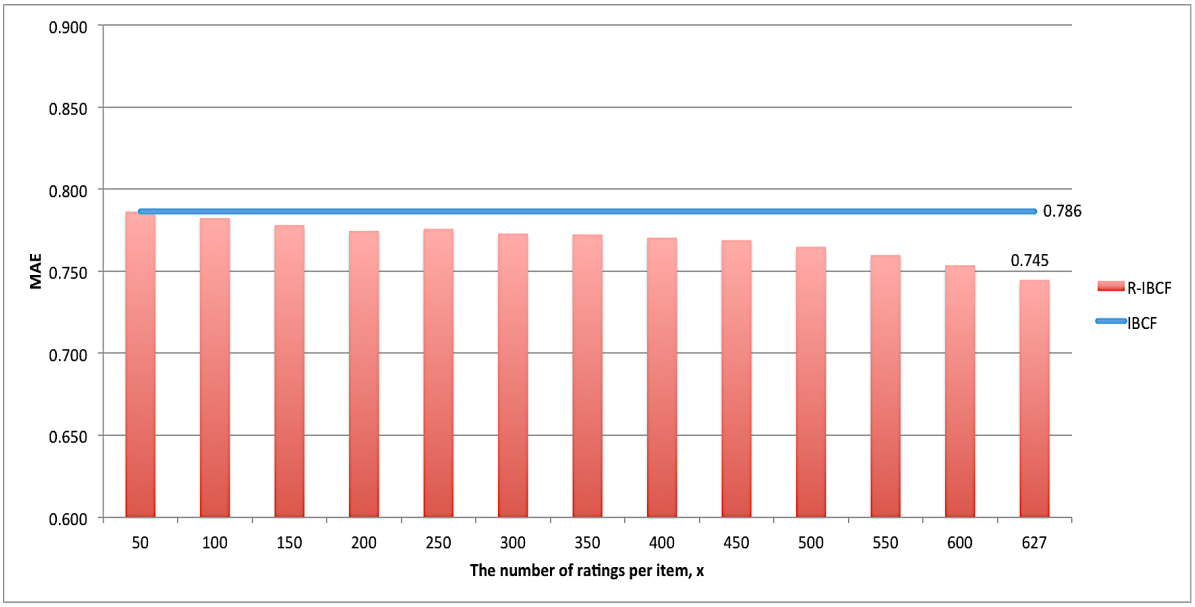
[Figure 9] The impact of the similarity computation on IBCF and R-IBCF

[Figure 9] shows the experiments results. I observed that IBCF applying dimension reduction generally produced better quality of predictions more than IBCF with four similarity measurements. In particular, Tanimoto coefficient has a clear advantage, as MAE is the lowest on IBCF and R-IBCF. Therefore, I select Tanimoto coefficient similarity for the rest of my experiments.

5.2 Optimum The Number of Ratings per Item

If a user does not rate at least one item, the system cannot recommend any items to the user. In this dataset, each item needs to have at least 627 ratings in order to recommend items to all users. That is, each item has to be rated by at least 627 users. For example, if I use the optimized data with items

having at least 628 ratings, after dimension reduction based on it, one person among 6040 users cannot get item recommendation. This is because that user did not give items any ratings at all. Therefore, to prevent non-recommendation from happening, I performed these experiments to find the optimal number of ratings per item ranging from 50 (similar to raw data) to 627 (smaller dataset) in mostly increments of 50, and computed MAE.



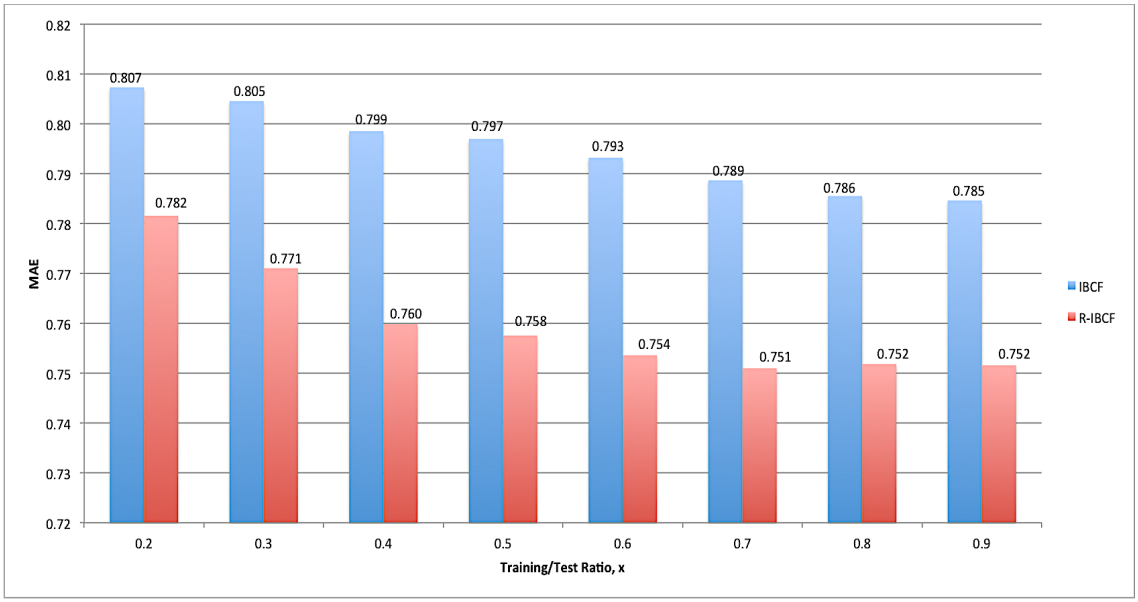
[Figure 10] Comparison of Impact of the number of ratings on R-IBCF to IBCF

My results are shown in [Figure 10]. I observed that the quality of prediction increases as I apply reducing dimensions on IBCF based on the number of ratings, x. When I reduce a few dimensions (x = 50), the quality of prediction is almost same with IBCF (MAE = 0.786). On the other hand, when I reduce lots of dimensions (x = 627), the quality of prediction is the best (MAE =

0.745). Therefore, I select 627 as an optimal value in terms of the number of ratings per item for the rest of my experiments.

5.3 Optimum Training/Test Ratio

To determine the sensitivity of density of the dataset, I ran these experiments where I varied the value of training/test ratio ranging from 0.2 to 0.9 in an increment of 0.1 and computed MAE. For instance, x is 0.2 means that my experiments run with 20% of dataset as training data and 80% of dataset as test data.

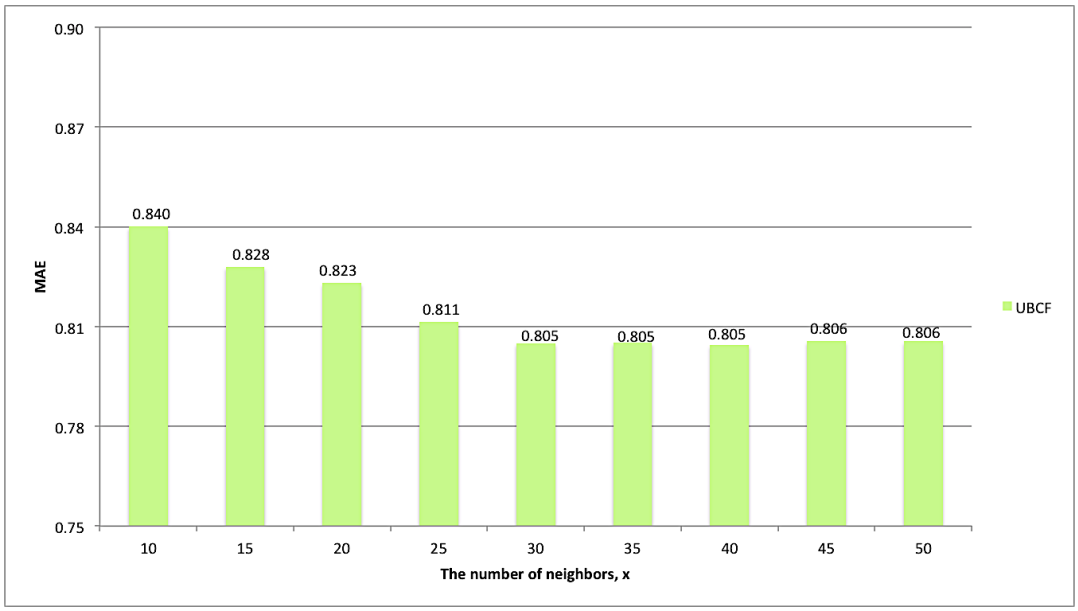


[Figure 11] Sensitivity of the parameter x in IBCF and R-IBCF

The results are shown in [Figure 11]. I observed that applying dimension on IBCF generally makes the quality of prediction better than IBCF. When the training ratio is 0.7, R-IBCF tends to be flat. Hence, I select 0.7 as optimal choice of training/test ratio for the rest of my experiments.

5.4 Optimum The Neighborhood Size of UBCF

The size of neighborhood on UBCF plays an important role in affecting the prediction quality [8]. To find the sensitivity of neighborhood size, I ran an experiment where I varied the number of neighbors ranging from 10 to 50 in an increment of 5 to be used, and measured MAE on UBCF.



[Figure 12] Sensitivity of neighborhood size in UBCF

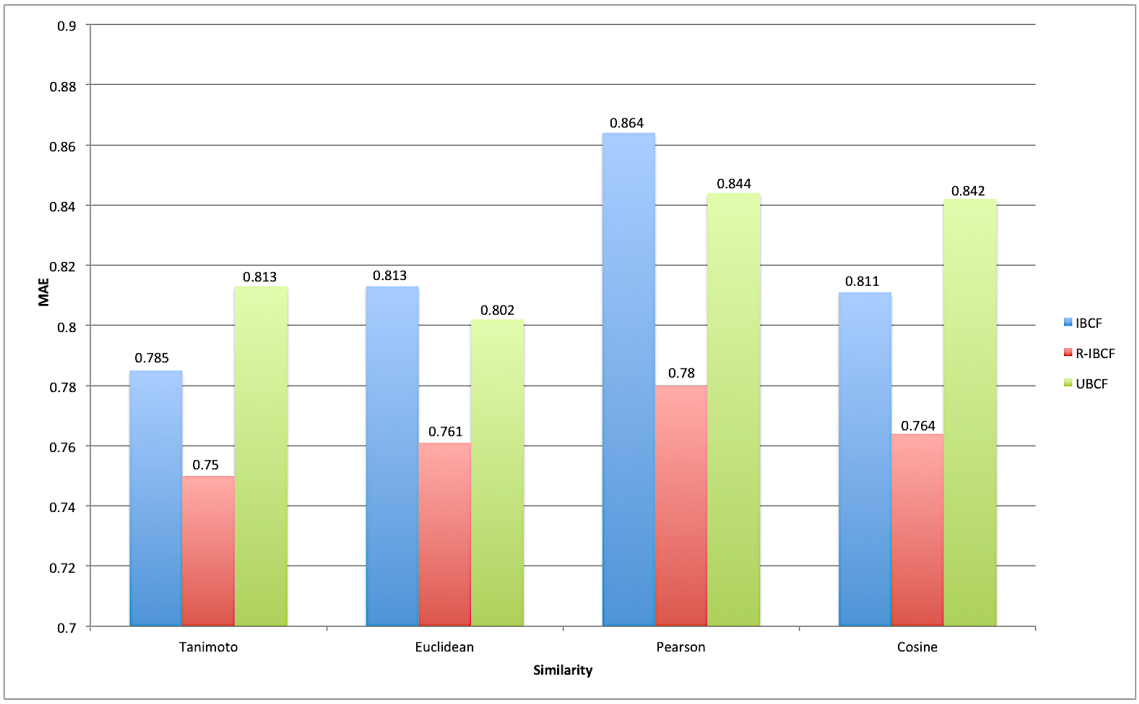
[Figure 12] shows the experimental results. I observed that neighborhood size has an effect on the quality of prediction. The quality of prediction gets better by increasing the number of neighbors and when the rate is 30, the quality of predictions tends to be flat. Therefore, I select 30 as an optimum value of the neighborhood size for subsequent experiments with UBCF.

5.5 Comparison of Prediction Quality with Benchmark

Once I obtained the optimal values of the parameters, I compared both IBCF and R-IBCF approaches with the benchmark UBCF. The purpose of this experiment was to determine how each similarity algorithm influences the quality of prediction accuracy of IBCF, R-IBCF, and UBCF. It is a critical step in collaborative filtering to compute the similarity between each item or each user in selecting the most similar neighbors of them.

I present the results in [Figure 13]. I performed them with selected values: 0.7 as the optimum training on three CFs, 627 as an optimal value of the number of ratings per item on R-IBCF, and 30 as an optimal size of a neighbor on UBCF. Overall, R-IBCF provides better quality of predictions than IBCF and UBCF. It can be observed that data sparsity and data scalability problems affect the quality of predictions in computing similarity between items. Reducing dimensions means that it does not take into account items having fewer ratings by users, but considers typical or representative items. Therefore, the results

show that reducing dimensions on IBCF contributes greatly to improve the quality of predictions in terms of data sparsity and data scalability.



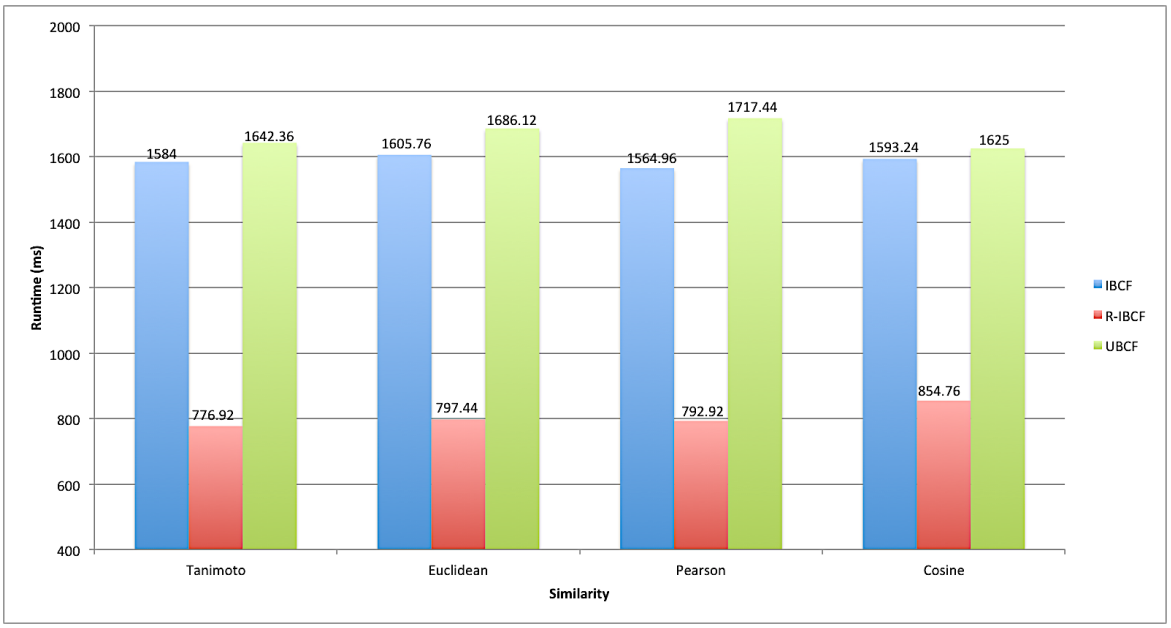
[Figure 13] Comparison of the prediction quality of IBCF, R-IBCF, and UBCF

5.6 Comparison of Runtime with Benchmark

Runtime of performance is also an important point in terms of data scalability. I implemented R-IBCF consuming memory. I ran each experiment with four similarity algorithms 30 times and got the average of their runtime excluding the first 5 times.

These results are shown in [Figure 14]. Even though it takes more time to filter data based on the number of ratings per item, I observed that it is faster

than computing similarity between all co-rated items or all users. Therefore, reduction of dimension on IBCF has considerable impact on runtime being fast in terms of data scalability. In addition, because IBCF and R-IBCF only consider co-rated items to compute similarity, they do not take finding the nearest neighbors step. Therefore, it generally influences on better runtime of IBCF and R-IBCF by comparison with UBCF.



[Figure 14] Comparison of runtime of IBCF, R-IBCF, and UBCF

CHAPTER 6

CONCLUSION AND FUTURE WORK

Recommendation systems have been an important in E-commerce on the web for the customer to suggest items what they would be interested. With the increasing number of users and items, recommendation systems encounter the main shortcoming: data sparsity and data scalability problems, which bring out the poor quality of prediction and the inefficient time consuming.

In this paper, I have proposed item-based collaborative filtering approach applying dimension reduction to improve the predictive accuracy and recommendation quality in overcoming the existing limitations. By reducing the noise of dimensional data, it focuses on typical and popular items to compute the similarity between them and to predict the most similar items to users. The experimental results show that this approach makes a considerable impact on providing better accuracy of prediction and much faster execution time in comparison with traditional UBCF and IBCF. It results in improving the quality of recommendation system using collaborative filtering.

The potential limitation would use this approach with dataset widely consisting of not enough ratings by users, expecting less accuracy. Therefore, to overcome this challenge, I propose an approach to mix both explicit and implicit ratings to alleviate the data sparsity problem further in this aspect.

REFERENCES

[1] Schafer, J. Ben, Joseph Konstan, and John Riedl. 1999. “Recommender Systems In E-Commerce.” In *1St ACM Conference On Electronic Commerce*, 158-166.

[2] Su, Xiaoyuan, and Taghi M. Khoshgoftaar. 2009. “A Survey Of Collaborative Filtering Techniques.” *Advances In Artificial Intelligence* 2009: 1-19. doi:10.1155/2009/421425.

[3] Melville, Prem, and Vikas Sindhwani. 2010. “Recommender Systems.” *Encyclopedia Of Machine Learning*.

[4] Xingyuan Li.2011 “Collaborative Filtering Recommendation Algorithm Based on Cluster”, *International Conference on Computer Science and network Technology(ICCSNT),* IEEE, 4: 2682-2685.

[5] Francesco Ricc, Lior Rokach, Bracha Shapira.2011. *Recommender Systems Handbook*. NY:Springer, 1-35

[6] Sarwar, George Kaypi, Joseph Konstan and John Riedl.2000. "Application of Dimensionality Reduction in Recommender Systems -- A 6 Study." In ACM WebKDD Workshop.

[7] Sarwar, George Kaypi, Joseph Konstan, John Riedl.2000. “Analysis of recommendation algorithms for E-commerce.” *In the Second ACMConference on Electronic Commerce*, 158–167

[8] Sarwar, George Kaypi, Joseph Konstan, John Riedl.2001. “Item-based Collaborative Filtering Recommendation Algorithms.” *In the 10th International World Wide Web Conference*, 285-295

[9] Gong, Songjie. 2010. “A Collaborative Filtering Recommendation Algorithm Based On User Clustering And Item Clustering.” *JSW* 5 (7). doi:10.4304/jsw.5.7.745-752.

[10] Yan Shi, Xiao, HongWu Ye, and SongJie Gong. 2008. “A Personalized Recommender Integrating Item-Based And User-Based Collaborative Filtering.” *ISBIM '08 International Seminar On Business And Information Management* 1 (2008): 264-267.

[11] Mangalindan, JP. 2012. “Amazon’S Recommendation Secret.” *Fortune*. http://fortune.com/2012/07/30/amazons-recommendation-secret/.

[12] Koren, Yehuda.2009. “Collaborative Filtering with Temporal Dynamics.” *15th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD 09), ACM (2009)*: 447-455.

[13] Liu, Jiahui, Dolan, Peter, Pedersen, Elin Rønby.2010. “ Personalized news recommendation based on click behavior”, In: Rich, et al. (eds.) *In the 14th Int. Conf. on Intelligent User Interfaces (IUI), ACM, (2010)*: 31–40.

[14] Owen, Sean, Anil, Robin, Dunning, Ted, Friedman, Ellen. 2011 . *Mahout in action.* Shelter Island NY: Manning. ()

[15] Walunj, Sachin, Sadafale, Kishor. 2013. “An online recommendation system for e-commerce based on apache mahout framework.” *Proceedings of the 2013 annual conference on Computers and people research, ACM (2013)*: 153–158.

[16] Verbert, Katrien, Drachsler, Hendrik, Manouselis, Nikos, Wolpers, Martin, Vuorikari, Vuorikari, Riina, Duval, Erik.2011. “Dataset-driven Research for Improving Recommender Systems for Learning.” *1st International Conference Learning Analytics & Knowledge, ACM (2011)*: 44-53.

[17] Wikipedia,. 2015. “Cosine Similarity.” https://en.wikipedia.org/wiki/Cosine_similarity.

[18] Duryee, Tricia. 2014. “Amazon Adds 30 Million Customers In The Past Year - Geekwire.” *Geekwire*. http://www.geekwire.com/2014/amazon-adds-30-million-customers-past-year/.

[19] Grey, Paul. 2013. “How Many Products Does Amazon Sell? | Exportx.” *Exportx*. https://export-x.com/2013/12/15/many-products-amazon-sell/.

[20] Resnick, Paul, Iacovou, Neophytos, Suchak, Mitesh, Bergstrom, Peter, Riedl, John.1994. “ GroupLens: an open architecture for collaborative filtering of netnews.” *CSCW conference, ACM (1994).*

[21] GroupLens,. 2013. “Movielens'. http://grouplens.org/datasets/movielens/.”