Implementation and Evaluation of Movie Recommender Systems Using Collaborative Filtering

Salam Salloum and Dananjaya Rajamanthri
Department of Computer Science, California State Polytechnic University, Pomona, CA, USA
Email: ssalloum@cpp.edu

***Abstract*—Recommender systems have been utilized in several e-commerce applications. There are three types of recommender systems: content based filtering, collaborative filtering, and hybrid recommender systems. In this paper, two types of collaborative filtering techniques are evaluated using the Movielens dataset, which contains 1 million ratings. These two types are matrix factorization and user based collaborative filtering with cosine similarity function. The evaluation of the two types is based on the Root Mean Square Error (RMSE) of the complete dataset and different partitions of the complete dataset. The partitions are determined by age, genre, or date of rating. For both types, the results show that the RMSE of the complete dataset is less than that of each partition. Also, in this thesis, we introduce a new hybrid technique which integrates age, genre, and date into the definition of cosine similarity function. The new technique is evaluated using two Movielens datasets of different sizes: 100,000 ratings and 1 million ratings. For both datasets, the evaluation results show that the RMSE of the new hybrid technique is less than that of the user based collaborative filtering with traditional cosine function. For the dataset containing 100,000 ratings, the evaluation results show that the RMSE of the new technique is lower than that of matrix factorization for small training sets and higher for large training sets.**

***Index Terms*—content based filtering, collaborative filtering, hybrid recommender systems, cosine similarity**

I. Introduction

Data mining can be defined as discovering models and patterns in large datasets [1], [2]. Recommender systems are one of the data mining applications that are used to predict user responses to options [2]. Recommender systems are classified as follows:

- Content based filtering: Recommending items with similar content to the items the user preferred.
- Collaborative filtering: Recommending items preferred by similar users.
- Hybrid approaches: combining content-based and collaborative-filtering methods in several different ways [3]-[8].

Manuscript received November 30, 2020; revised April 4, 2021.

A. Utility Matrix

In recommender systems, users' preferences for items are called ratings. Usually a rating is an integer. The entity which maps users to ratings is called the utility matrix [2]. All the recommender systems are based on the utility matrix. In a utility matrix, the rows represent the users and the columns represent the items.

There are two ways of populating a utility matrix. The first way is asking users to rate items. For example, Netflix asks users to rate movies. Almost all the applications of recommender systems utilize this approach. Another approach is deriving the ratings based on user behavior. These derived ratings are typically Boolean values. For example, in a movie application, if a user watches a movie, we can assume that the user likes that movie. But we don't know to what extent the user likes the movie. That's why the predicted ratings based on user behavior are usually given in Boolean values.

The goal of a recommender system is predicting the missing elements or blanks of the utility matrix. We know that a utility matrix is populated by asking ratings from users. It is clear that the user is not going to rate all the items. So, each user has more blanks than filled elements in the utility matrix. The goal of a recommender system is predicting those blanks. In most cases, predicting blanks which would be rated highly by users is sufficient.

B. Content Based Filtering (CBF)

In content-based filtering, the utility u(c,s) of items for user c is estimated based on the utilities u(c,si) assigned by user c to items si Є S that are “similar” to s [9]-[11]. In another words, content based filtering is recommending similar items to items preferred by a user. Content based filtering incorporates three entities: utility matrix, item profile, and user profile. Previously, we discussed about utility matrix. Next, let's take a look at item profile and user profile.

Finally, in this section, we'll discuss the algorithm of content based filtering.

Item Profile: An item profile is a record of all the features of items. An item profile is populated by the application developers. Application developers have to analyze the items and try to extract the important characteristics of them. In content based filtering, this

doi: 10.12720/jait.12.3.189-196

process is critical because the performance of the system depends on how well the item profile is defined. For example, movies can have the following features:

- Genre
- Stars
- Director
- Year
- Language

This information can be easily extracted from the description of the movies. Genre is a vague concept. The best place to find the movie genre is the Internet Movie Database (IMDB). IMDB assigns a genre to every movie. Similarly, books can have the following features:

- Authors
- Subject
- Year
- Language
- Publisher

These features can also be extracted from the descriptions of the books.

User Profile: A user profile is a record of features a user prefers and how much the user prefers each feature. All of the features used in the item profile have to be used in the user profile too. We can also consider each row of the item profile and the user profile is a vector of which the components are the features. The user profile is an aggregation of the utility matrix and the item profile. After generating the user profile, recommendations are generated based on the user profile and the item profile.

Algorithm of Content Based Filtering: Pseudo code for algorithm of content based filtering is as follows:

```
//calculate user profile
for each user u
  userProfile[u] = r(u,i) * itemProfile[i] + userProfile[u]
    for each item i
            userProfile[u] = (r(u,i) * itemProfile[i] +
userProfile[u]) /2

//generate recommendations
for each user profile u
  for each item profile i of which item was
  not rated by u
    compute a similarity s between u and i  retain the top
items ranked by similarity
return top items.
```

In the above algorithm, userProfile[i] is a vector which represents the i^{th} row of the user profile. itemProfile[i] is a vector which represents the i^{th} row of the item profile. r(u,i) is the rating of user u for item i. First, the algorithm calculates the user profile aggregating the utility matrix and the item profile. Then, it generates recommendations based on the user profile and the item profile. For measuring similarity, distance measure such as cosine distance can be used.

C. Collaborative Filtering (CF)

In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple data sources. There are two main types of collaborative filtering techniques: user based collaborative filtering and matrix factorization.

User Based Collaborative Filtering (UBCF): In user based collaborative filtering, the utility r(u, i) of item i for user u is estimated based on the utilities r(uj, i) assigned to item i by those users uj ϵ U who are "similar" to user u [9]. The following is the algorithm of user based collaborative filtering for recommending items for user u:

```
for every other user w
  compute a similarity s between u and w,
retain the top users, ranked by similarity as the
neighborhood n,
for every item i that some user in n has a preference for,
but that u has no preference for yet
for every other user v in n that has a preference for i
incorporate v's preference for i into a running average
return the top items, ranked by weighted average [12].
```

According to the above algorithm, the first for loop calculates a set of similar users n which is called the neighborhood. The second for loop predicts ratings of the items which user u hasn't rated yet. To measure the similarity between two users, a distance measure such as cosine distance can be used.

There are two types of neighborhoods: fixed size neighborhood and threshold based neighborhood [11]. Fixed size neighborhood consists of n most similar users. Usually, when n increases, the accuracy of recommendation increases simply because there are more similar users. Threshold based neighborhood consists of similar users whose similarity measure is greater than value t where $-1 < t < 1$. Usually, when the threshold increases, the accuracy of the recommendation decreases simply because too few similar users end up in the neighborhood.

Matrix Factorization (MF): In matrix factorization, both users and items are mapped into a latent factor space such that user-item interactions are modeled as inner products in that space [13]. Each item i is associated with a vector $q_i \in R$ and each user u is associated with a vector $p_u \in R$. The rating of the user u rating for item i is denoted by r_{ui} which is estimated as follows:

$$r_{ui} = q_i{}^T p_u$$

For a given item *i*, the elements of q_i measures the extent to which the item possesses those factors. For a given user u, the elements of p_u measures the extent of interest a user has in factors. The dot product $q_i{}^T p_u$ measures the user u's interest in item *i*. In another words, the dot product $q_i{}^T p_u$ estimates the rating of user u rating for item *i*. The challenge is calculating the mapping of each item and user to the factor vectors p_u and q_i.

One popular way to find p_u and q_i by using gradient descent. In gradient descent, p_u and q_i is initialized with some value. Then, regularized squared error is calculated between the p_u q_i product and known rating r_{ui} . Then, p_u and q_i are adjusted such that the regularized squared error is minimum. This is done iteratively. Regularized squared error is defined as follows:

$$e_{iu}{}^2 = \sum_{(u,i)\in k} (r_{ui} - q_i{}^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

p_u and q_i can be adjusted as follows:

$$q_i = q_i + \gamma \frac{\partial\, e_{iu}{}^2}{\partial q_i}$$

$$p_u = p_u + \gamma \frac{\partial\, e_{iu}{}^2}{\partial p_u}$$

Or

$$q_i = q_i + \gamma(e_{ui} * p_u - \lambda * q_i)$$

$$p_u = p_u + \gamma(e_{ui} * q_i - \lambda * p_u)$$

Hybrid Recommender Systems: Hybrid recommender systems combine two or more recommender systems to minimize the drawbacks of any individual recommender system [3]-[8]. Based on combination techniques, hybrid recommender systems can be categorized into five main categories: weighted, switching, mixed, feature combination, and cascade.

Weighted hybrid recommender systems estimate ratings based on the ratings estimated by multiple recommender systems [4]. For example, ratings estimated by collaborative filtering and content based filtering can be linearly combined to produce a hybrid recommender. Weighted hybrid recommender systems bring the power of multiple recommender systems together in a simple manner. One recommender system can be added or removed easily from the system.

Switching hybrid recommender systems alternate between two or more recommender systems based on some criterion [4]. Unlike weighted hybrid recommender systems, switching hybrid recommender systems do not combine the results of multiple recommender systems to produce the final result. For example, content based filtering and collaborative filtering can be combined to produce a switching hybrid recommender system which will then generate recommendations with content based filtering and switch to collaborative filtering if it fails. Switching recommender systems are more complex due to criterion based switching.

Mix hybrid recommender systems combine recommendations generated from multiple recommender systems [4]-[8]. This kind of recommender systems are used when large numbers of recommendations are presented to the user. For example, we can combine recommendations from the content based filtering and collaborative filtering to produce a mix hybrid system. In combining, some kind of aggregation technique has to be used.

The feature combination hybrid recommender systems consider user ratings as additional feature information and run content-based filtering on the augmented data set [5]. The cascade hybrid recommender systems generate recommendations using a recommendation technique and use another recommendation technique to rank the recommendations of the first one. Thus, this technique consists of two stages.

Similarity Distance Function: User based collaborative filtering uses a similarity function to measure similarity between two users. Next, we'll investigate similarity functions.

Pearson correlation-Based Similarity: The Pearson correlation measures the tendency of two series of numbers, paired up one-to-one, to move together [11]. Pearson correlation can be calculated as follows:

$$Pearson\ sim(x,y) = \frac{\sum_{c\in L_{xy}}(R_{x,c} - \overline{R_x}) \times (R_{y,c} - \overline{R_y})}{\sqrt{\sum_{c\in L_x}(R_{x,c} - \overline{R_x})} \times \sqrt{\sum_{c\in L_y}(R_{y,c} - \overline{R_y})}}$$

where $R_{x,c}$ refers to the rating of user x for item c, L_{xy} denates the set of items that rated by both user x and y, $\overline{R_x}$ and $\overline{R_y}$ are the average rating of user x and y respectively. Pearson similarity is a number between -1 and 1.

Euclidean Similarity: Euclidean similarity measures how close two points are in Euclidean n-space.

Euclidean similarity is defined as follows:

$$Euclidean\ sim\ (x,y) = \frac{1}{1 + Euclidean\ distance\ (x,y)}$$

$$Euclidean\ distance\ (x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where *x* and *y* refers to two points. Euclidean similarity is a value between 0 and 1.

Cosine Similarity: Cosine similarity measures how close two vectors are. Cosine similarity can be defined as follows:

$$Cosine\ sim(x,y) = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i{}^2} * \sqrt{\sum_{i=1}^{n} y_i{}^2}}$$

where *x* and *y* are two vectors. Cosine similarity is a number between -1 and 1.

Spearman Correlation Similarity: Spearman correlation is a variant of Pearson correlation. Instead of using the actual preferences, in Spearman correlation, ranked preferences are used to calculate the correlation [11]. Spearman correlation is useful when the intervals between preferences are problematic.

Tanimoto coefficient: Tanimoto coefficient is used to measure the similarity between two sets when the elements of each set are binary. Tanimoto coefficiennt is also called Jaccard coefficient. Tanimoto coefficient is defined as the size of intersection divided by the size of the union [11] and can be represented by the following equation.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

where both *A* and *B* are sets with binary elements. When *A* and *B* are both empty, *J*(*A*,*B*) = 1. Thus,

$$0 \le J(A,B) \le 1$$

II. Evaluating User Based Collaborative Filtering with Cosine Function

The goal of this research is to improve user based collaborative filtering with cosine similarity. Our first attempt was to partition a dataset and check whether each partition would produce better recommendations. For evaluation, we used the Movielens dataset which contains one million ratings. We used the following procedure to evaluate a recommender algorithm.

- Split the dataset in to training set and probe set.
- Use recommender algorithm and predict the probe set using the training set.
- Use root mean square error (RMSE) to calculate the error between the real probe set and predicted probe set.

$$\text{Root mean square error (RMSE)} = \sqrt{\frac{\sum_{i=1}^{n}(R_i - r_i)^2}{n}}$$

In the above equation, R_i refers to the real rating, r_i refers to the calculated rating which corresponds to R_i, and n refers to the total number of real ratings.

A. Partitions Based on Age

We partitioned ratings based on age and calculated the error of user based collaborative filtering with cosine function while changing the percentage of the training dataset. Table I and Fig. 1 show the results.

TABLE I. RMSEs of UBCF With Cosine on Complete Set of 1 Million Ratings and Partitions Based on Age

training percentage	All	1	18	25	35	45	50	56
70	0.990715	1.355665	1.048646	0.987586	1.011545	1.078242	1.088309	1.253861
80	0.984574	1.302618	1.04177	0.977757	0.989276	1.040835	1.051507	1.198311
90	0.980789	1.269022	1.028613	0.973787	0.985838	1.033054	1.027331	1.164181

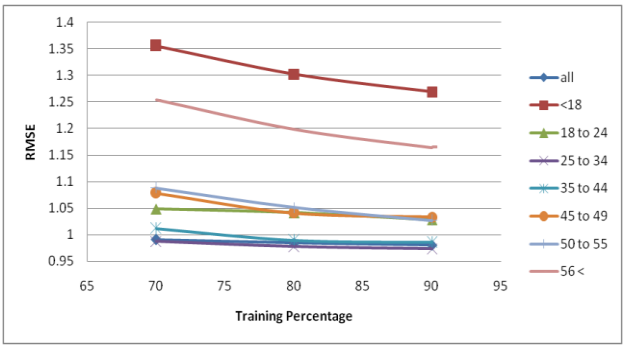

Figure 1. Plot of RMSEs of UBCF with cosine on complete set of 1 million ratings and partitions based on age.

From Table I and Fig. 1, we can see that error of user based collaborative filtering with cosine function on complete dataset is lower than the error of that on partitions.

B. Partitions Based on Genre

We partitioned ratings based on genre and calculated the error of user based collaborative filtering with cosine function while changing the percentage of the training dataset. Table II and Fig. 2 show the results.

TABLE II. RMSEs of UBCF with Cosine on Complete Set of 1 Million Ratings and Partitions Based on Genre

training percentage	All	genre 1	genre 2	genre 3	genre 4	genre 5
70	0.990715	0.987934	1.012781	1.148127	1.119669	1.046123
80	0.984574	0.987934	0.998815	1.148127	1.084554	1.03291
90	0.980789	0.981574	0.998669	1.148127	1.073493	1.020698

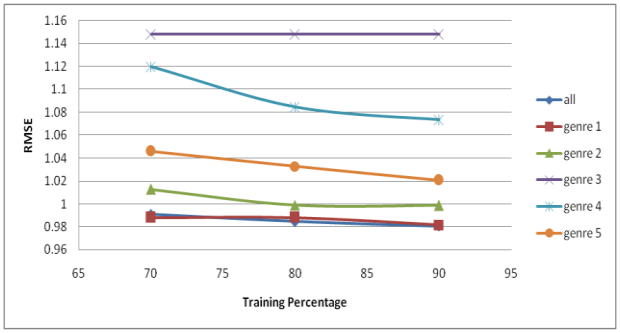

Figure 2. Plot of RMSEs of UBCF with cosine on complete set of 1 million ratings and partitions based on genre.

From Table II and Fig. 2, we can see that error of user base collaborative filtering with cosine function on complete dataset is lower than the error of that on partitions.

C. Partitions Based on Date

We partitioned ratings based on time of rating and calculated the error of user based collaborative filtering with cosine while changing the percentage of the training dataset. Table III and Fig. 3 show the results.

TABLE III. RMSEs of UBCF with Cosine on Complete Set of 1 Million Ratings and Partitions Based on Date

training percentage	All	2000	2001	2002	2003
70	0.990715	1.002803	1.223215	1.450935	1.507215
80	0.984574	0.992762	1.171007	1.403151	1.510001
90	0.980789	0.986636	1.13869	1.394571	1.528393

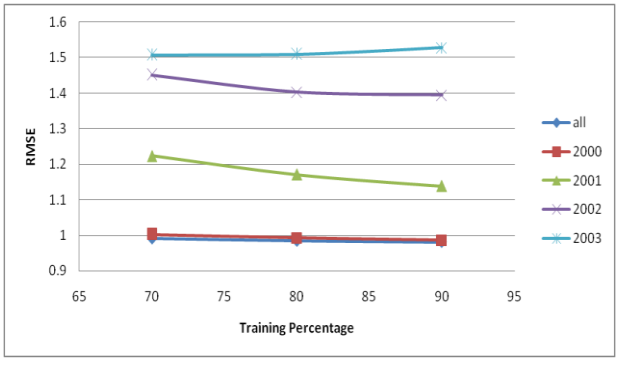

Figure 3. Plot of RMSEs of UBCF with cosine on complete set of 1 million ratings and partitions based on date.

From Table III and Fig. 3, we can see that the error of user based collaborative filtering with cosine function on complete dataset is lower than the error of that on partitions.

III. Evaluating User Based Collaborative Filtering with Modified Cosine Function

The goal of this research is to improve user based collaborative filtering where the similarity function used is the cosine function. As we discussed in Section II, we tried to achieve that by partitioning the dataset based on age, genre, or date, which was not successful. Next, we tried to achieve our goal by modifying the cosine similarity. In this chapter, we are going to discuss modified cosine similarity and compare RMSEs of user based collaborative filtering with the original and modified cosine functions. For evaluation, we used two Movielens datasets which contain 100,000 ratings and 1 million ratings.

A. Original Cosine Similarity

Original cosine similarity is defined as follows:

$$\frac{\sum_{i=1}^{n} rating_{ai} * rating_{bi}}{\sqrt{\sum_{i=1}^{n} rating_{ai}{}^2} * \sqrt{\sum_{i=1}^{n} rating_{bi}{}^2}}$$

where $rating_{ai}$ refers to the rating user a gave for movie i and $rating_{bi}$ refers to the rating user b gave for movie i.

B. Modified Cosine Similarity

We modified the cosine similarity with the following assumptions:

- The users with similar age are more similar than the rest of the users.
- The users who are interested in the same genre are more similar than the rest of the users.
- The ratings given during the same time period are more similar than the rest of the ratings.

Modified cosine similarity is as follows:

$$\frac{\sum_{i=1}^{n} w_t * rating_{ai} * rating_{bi}}{\sqrt{\sum_{i=1}^{n} rating_{ai}{}^2} * \sqrt{\sum_{i=1}^{n} rating_{bi}{}^2}} + w_a + w_g$$

where $rating_{ai}$ refers to the rating user a gave for movie i and $rating_{bi}$ refers to the rating user b gave for movie i.

w_a refers to the age weight and can be calculated as follows:

$$|age_a - age_b| \leq L_a => w_a = c_a$$

$$|age_a - age_b| > L_a => w_a = 0$$

$$L_a = 5, c_a = 0.5$$

w_t refers to the time weight and can be calculated as follows (assume time is in years):

$$|time_{ai} - time_{bi}| \leq L_t => w_t = c_t$$

$$|time_{ai} - time_{bi}| > L_t => w_t = 1$$

$L_t = 1, c_t = 2w_g$ refers to the genre weight and calculated as follows:

$$Genres\ a\ veiwed\ the\ most \cap Genres\ b\ veiwed\ the\ most \neq \emptyset => w_g = c_g$$

$$Genres\ a\ veiwed\ the\ most \cap Genres\ b\ veiwed\ the\ most = \emptyset => w_g = 0$$

$$c_g = 0.5$$

For implementation, we used Mahout recommender framework. Mahout is an open source machine learning library from apache. The current version of Mahout implements machine learning algorithms belonging to three categories: recommendation, classification, and clustering. It is aimed to be used by applications with data that is too large for a single computer to process. To achieve the scalability, it is built on top of Apache Hadoop which is a distributed computing system. It is implemented in Java.

Table IV and Fig. 4 compare the RMSEs of user based collaborative filtering with original and modified cosine functions on the dataset containing 100,000 ratings.

TABLE IV. RMSEs of UBCF with Original and Modified Cosine on Complete Set of 100, 000 Ratings

training percentage	RMSE of Original UBCF	RMSE of Midified UBCF
70	1.097501426	0.992585309
80	1.062239766	0.986342565
90	1.056571337	0.990040666

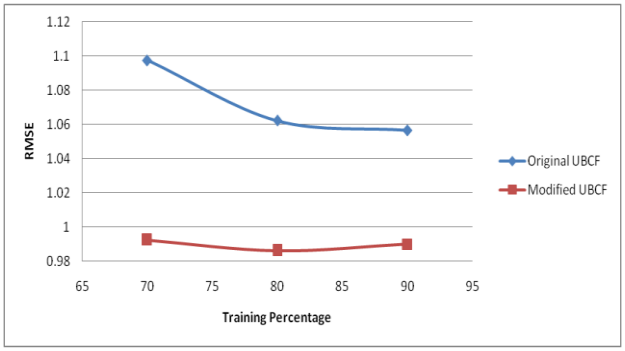

Figure 4. Plot of RMSEs of UBCF with original and modified cosine on complete set of 100,000 ratings.

From Table IV and Fig. 4, we can see that the RMSE of user based collaborative filtering with modified cosine function is lower than the original one. We also calculated RMSEs of user based collaborative filtering with original and modified cosine functions on Movielens dataset containing 1 million ratings for 90 percent training set. The results were consistent with the previous results.

IV. Evaluating Factorization Methods

We calculated RMSE of matrix factorization on age, genre, and date partitions of Movielens dataset containing one million ratings. The first three subsections of this section are devoted to present the results of that experiment. At the end of this section, we are going to compare RMSE of matrix factorization to RMSE of user based collaborative filtering with modified cosine on the Movielens dataset containing 100,000 ratings.

A. Partitions Based on Age

We partitioned ratings based on age and calculated error of matrix factorization while changing the percentage of the training dataset. Table V and Fig. 5 show the results.

TABLE V. RMSEs OF MF ON COMPLETE SET OF 1 MILLION RATINGS AND PARTITIONS BASED ON AGE

training percentage	All	1	18	25	35	45	50	56
70	0.866632	1.229243	0.960782	0.888546	0.910918	0.983122	0.991166	1.076821
80	0.861228	1.206244	0.937142	0.877081	0.901742	0.96026	0.977943	1.046889
90	0.854716	1.176732	0.927782	0.867745	0.88919	0.917808	0.95727	1.022675

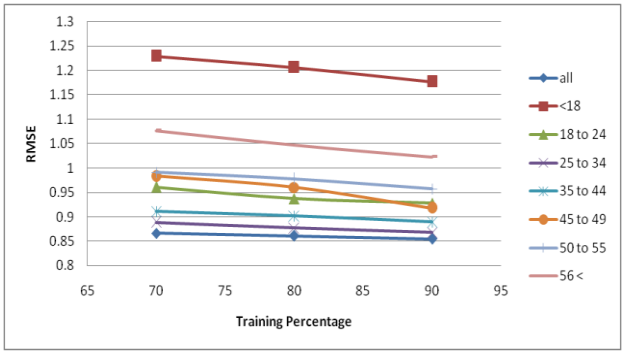

Figure 5. Plot of RMSEs of MF on complete set of 1 mllion ratings and partitions based on age.

From Table V and Fig. 5 we can see that the error of matrix factorization on complete dataset is lower than the error of that on partitions.

B. Partitions Based on Genre

We partitioned ratings based on genre and calculated the error of matrix factorization while changing the percentage of the training dataset. Table VI and Fig. 6 show the results.

TABLE VI. RMSEs OF MF ON COMPLETE SET OF 1 MILLION RATINGS AND PARTITIONS BASED ON GENRE

training percentage	All	genre 1	genre 2	genre 3	genre 4	genre 5
70	0.866632	0.880333	0.932697	0.967414	0.946097	0.910816
80	0.861228	0.873504	0.921348	0.939902	0.927442	0.903401
90	0.854716	0.857486	0.899124	0.936145	0.910057	0.892047

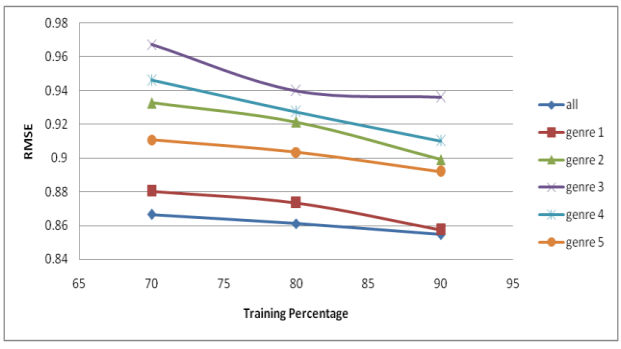

Figure 6. Plot of RMSEs of MF on complete set of 1 million ratings and partitions based on genre

From Table VI and Fig. 6, we can see that the error of matrix factorization on complete dataset is lower than the error of that on partitions.

C. Partitions Based on Date

We partitioned ratings based on date and calculated the error of matrix factorization while changing the percentage of the training dataset. Table VII and Fig. 7 show the results.

TABLE VII. RMSEs OF MF ON COMPLETE SET OF 1 MILLION RATINGS AND PARTITIONS BASED ON DATE

training percentage	All	2000	2001	2002	2003
70	0.866632	0.873139	1.024393	1.182956	1.489212
80	0.861228	0.86434	1.007889	1.162301	1.506717
90	0.854716	0.859452	0.984957	1.08671	1.467378

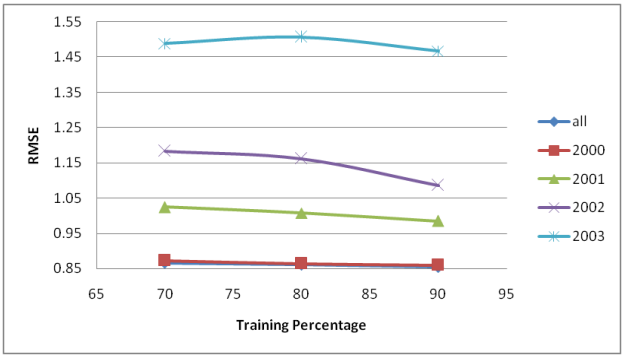

Figure 7. Plot of RMSEs of MF on complete set of 1 million ratings and partitions based on date.

From Table VII and Fig. 7, we can see that the error of matrix factorization on complete dataset is lower than the error of that on partitions.

These results are consistent with the results of the collaborative filtering in Section II.

D. Comparing RMSE of Matrix Factorization with RMSE of User Based Collaborative Filtering with Modified Cosine Function

In Section III, we measured the RMSE of modified user based collaborative filtering. We noticed that the RMSE of user based collaborative filtering with modified cosine function was lower than the RMSE of user based collaborative filtering with original cosine function. In this subsection, we are going to compare the RMSE of matrix factorization to that of user based collaborative filtering with modified cosine function. Table VIII and Fig. 8 show RMSEs for both methods.

TABLE VIII. RMSEs OF UBCF, UBCF WITH MODIFIED COSINE, AND MF ON COMPLETE SET OF 100,000 RATINGS

training percentage	RMSE of UBCF	RMSE of Midified UBCF	RMSE of MF
70	1.097501426	0.992585309	0.997306
80	1.062239766	0.986342565	0.978038
90	1.056571337	0.990040666	0.946081

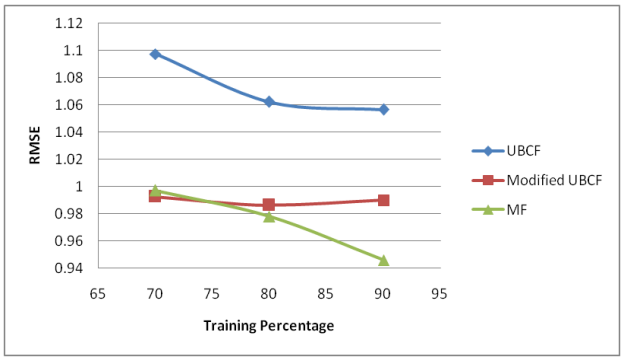

Figure 8. Plot of RMSEs of UBCF, UBCF with modified cosine, and MF on complete set of 100, 000 ratings.

From Table VIII and Fig. 8, we can see that the original user based collaborative filtering has the highest RMSE. The RMSE of user based collaborative filtering with modified cosine function is lower than that of matrix factorization for small training sets and greater for large training sets.

V. Conclusion

Recommender systems have been utilized in several e-commerce applications, such as Amazon and Netflix. There are three types of recommender systems: content based filtering, collaborative filtering, and hybrid techniques. In this thesis, two types of collaborative filtering techniques were evaluated using the Movielens dataset containing one million ratings. These two types were matrix factorization and user based collaborative filtering with cosine similarity function. The evaluation of the two types was based on RMSE of complete dataset and different partitions of complete dataset. The partitions were determined by age, genre, and time. For both techniques, the results showed that the RMSE of the complete set is less than that of each partition. The reason why we got more error in partitioning is that the attributes of movies in one partition is dependent on the attributes of the movies in other partitions. In partitioning, we maximize the contribution of one partition while ignoring the contribution of the other partitions. Thus, this method is subtractive. What we need is an additive method in which we maximize the contribution of each partition, while keeping the contribution of the other partitions.

Also in this paper, we introduced a new hybrid technique by integrating age, genre, and date in to the definition of cosine similarity function. The new technique was evaluated using the Movielens dataset containing 100,000 ratings. The evaluation results showed that the RMSE of the new technique is less than that of the user based collaborative filtering with traditional cosine function. We also calculated RMSEs of user based collaborative filtering with original and modified cosine functions on the Movielens dataset containing 1 million ratings for 90 percent training set. The results were consistent with the previous results. The reason for this is that the new technique maximizes the contribution of each partition without ignoring the contribution of the other partitions. Thus, this method is an additive method.

We also compared the RMSEs of matrix factorization to that of user based collaborative filtering with modified cosine functions on the Movielens dataset containing 100,000 ratings. The RMSE of the new technique is lower than that of matrix factorization for small training sets and higher for large training sets.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

This manuscript is a product of Dananjaya Rajamanthri's Master thesis research under the supervision and guidance of Dr. Salam Salloum.

References

[1] U. Panniello and M. Gorgoglione, "A contextual modeling approach to context-aware recommender systems," in *Proc. the 3rd Workshop on Context-Aware Recommender Systems*, October 2013.

[2] A. Rajaraman, J. Leskovec, and J. D. Ullman, "Recommendation systems," in *Mining of Massive Datasets*, Cambridge University Press, 2012, pp. 303-323.

[3] G. Badaro, H. Hajj, W. El-Hajj, and L. Nachman, "A hybrid approach with collaborative filtering for recommender systems," in *Proc. 9th International Wireless Communications and Mobile Computing Conference*, 2013, pp. 349-354.

[4] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331-370, November 2002.

[5] J. Fan, W. Pan, and L. Jiang, "An improved collaborative filtering algorithm combining content-based algorithm and user activity," in *Proc. International Conference on Big Data and Smart Computing*, 2014, pp. 88-91.

[6] F. Fouss and M. Saerens, "Evaluating performance of recommender systems: An experimental comparison," presented at the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008.

[7] H. Ji, J. Li, C. Ren, and M. He, "Hybrid collaborative filtering model for improved recommendation," in *Proc. IEEE International Conference on Service Operations and Logistics, and Informatics*, 2013, pp. 142-145.

[8] K. Jia and Z. Yi, "Context-aware recommender systems in mobile applications," presented at the 6th International Conference on Information Management, Innovation Management and Industrial Engineering, 2013.

[9] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, June 2005.

[10] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, "Context-Aware recommender systems," *AI Mag.*, vol. 32, pp. 67-80, 2011.

[11] G. Adomavicius and Y. Kwon, "New recommendation techniques for multimedia rating systems," *IEEE Intelligent Systems*, pp. 48-55, 2007.

[12] S. Owen, R. Anil, T. Dunning, and E. Friedman, "Recommendations," in *Mahout in Action*, Manning Publications Co, 2012, pp. 2-114.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30-37, 2009.

Salam S. Salloum received a Ph.D. in computer science from the University of Southern California (USC), Los Angeles U.S.A in June 1979. He also recieved an M.S. in computer Science, an M.S. in applied mathematics and an M.S. in electrical engineering from University of Southern California (USC) in 1974, 1977, and 1980, respectively. Dr. Salloum's research spans the fields of algorithms, database and data mining, software testing, fault tolerant computing, and residue number system.

He was a faculty member at several univeristies in the Middle East and was the founding chair of the Computer Science Department in the Univeristy of Baghdad, Iraq. He is currently a Professor of computer science at California State Polytechnic University, Pomona U.S.A.

Dananjaya Rajamanthri received his M.S. in computer scinece from Califoronia State Polytechnic University, Pomona U.S.A. in 2014. He has been working as software enguneer in PSC Biotech Corporation located Pomona, U.S.A.