REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A NEW WEIGHTING APPROACH TO SOLVE DATA SPARSITY PROBLEM IN COLLABORATIVE FILTERING

TASNIM ZAYET

MSc. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING

ADVISER
ASSOC. PROF. DR. M. ELİF KARSLIGİL

İSTANBUL, 2017

ACKNOWLEDGEMENTS

First and foremost, I want to thank the most important person in my life, my mother, the one who worked hard to rais me and the one who always support and encourage me to follow my dreams.

Second, I would like to thank my advisor, Dr. Mine Elif Karslıgil, for her precious advices, guidance and support. I would also like to thank her for the proofreading of this text.

Third, I want to thank the Turkish government that granted me the scholarship and gave me this precious opportionity to study in Turkey.

Finally, I would like to warmly thank all of my proffessors and teachers who teached me during the past years from the kindergarden untill now. The ones that without them I will not reach where I am now.

May, 2017

Tasnim ZAYET

TABLE OF CONTENTS

Page

LIST OF SYMBOLS

u_a	The active user
$Sim(u_a,u)$	The similarity between user u_a and user u
$R_{u,j}$	The rank of user u to item j
$\overline{R_u}$	The mean of user u's ranks
U	The set of all users
I	The set of all items
C	The set of all categories
I_n	The set of items that have been watched by user u_n
V_n	The set of vectors of the item-category matrix of user u_n
V	The prediction part of the weight
Sum-diff	The similarity part of the weight
U_s	The set of similar users
CW	The set of the categories' weight

LIST OF ABBREVIATIONS

CF Collaborative Filtering
UCW User-Category matrix

LIST OF FIGURES

Page

LIST OF TABLES

ASTRACT

A NEW WEIGHTING APPROACH TO SOLVE DATA SPARSITY PROBLEM IN COLLABORATIVE FILTERING

Tasnim ZAYET

Department of Computer Engineering

MSc. Thesis

Adviser: Assoc. Prof. Dr. Mine Elif KARSLIGİL

Similarity measure is an important part of user-based collaborative filtering, where the prediction of the rank of the item depends on the similarity between users. In this thesis, a novel weighting technique for the similarity measure is proposed in order to increase the accuracy of collaborative filtering. In this technique, the user weight of each category is calculated using the ranks of items that belong to these categories. This process converts the user-item model to user-category model. The sum of the differences between two users' models forms the first part of the weight and it can be used as a similarity measure. The second part of the weight is generated depending on the categories of the item that its rank will be predicted, the active user's weights and the similar users' weights of these categories. Using this weight as a similarity measure then multiplied it with the cosine similarity has improved the collaborative filtering accuracy almost between 2% and 0.6% by changing the number of similar users for MovieLens 100K dataset between 4 and 20 by adding 4 users each time.

Data sparsity problem is one of the biggest problems that faces the collaborative filtering algorithm. It means missing data and it almost happens when users do not rank the items they watched or bought. In order to solve this problem or at least to decrease its effects on collaborative filtering accuracy, using the percentages of the items that belong to each category as a weight of each category in the user-category matrix, a value has been generated to replace the missing ranks. Using this value, the mean absolute error of the sparse data has been decreased almost to the half in cases of high percentages of missing ranks and it improve the recommendation accuracy to a good extent in cases of low percentages of sparse data.

Key words:Collaborative filtering, recommendation systems, data sparsity

ÖZET

İŞBİRLİKÇİ FİLTRELEME YÖNTEMİ İLE SEYREK VERİ PROBLEMİNİ ÇÖZMEK İÇİN YENİ BİR AĞIRLIK BENZETİM YAKLAŞIMI

Tasnim ZAYET

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Doç. Dr. Mine Elif KARSLIGİL

Kullanıcı tabanlı işbirlikçi filtreleme yöntemlerinde, bir ögenin tahmini puanının belirlenmesi kullanıcıların benzerliğine bağlı olduğundan ötürü, benzerlik ölçümünün nasıl yapıldığı önemli bir etmendir. Bu tez çalışmasında, işbirlikçi filtrelerin doğruluğunu arttırmak için, benzerlik ölçümünde kullanılacak yeni bir ağırlıklandırma yöntemi önerilmiştir. Bu yöntemde,her kategorinin kullanıcı ağırlığı, bu kategorilere ait ögelerin puanları kullanılarak hesaplanır. Bu işlem, kullanıcı-öge modelini kullanıcı kategori modeline dönüştürür. İki kullanıcının modelleri arasındaki farkların toplamı, ağırlığın ilk kısmını oluşturur ve bir benzerlik ölçüsü olarak kullanılabilir. Ağırlığın ikinci kısmı, puanı tahmin edileceki ögenin kategorilerine, aktif kullanıcının ağırlığına ve bu kategorilerin benzer kullanıcılarının ağırlıklarına bağlı olarak üretilir.Bu ağırlığı bir benzerlik ölçütü olarak kullanıp kosinüs benzerliği sonucu ile çarparak sistem başarısı ölçülmüştür., Her seferinde 4 kullanıcı ekleyerek MovieLens 100K veri seti için benzer kullanıcı sayısını 4-20 arasında değiştirerek yapılan testlerde, işbirlikçi filtrelemenin doğruluk değerinin % 2-a% 0,6 arasında değiştiği gözlenmiştir.

Kullanıcılar tarafından puanlanan ögelerin sayısının, toplam öge sayısına oranla çok küçük değerlerde olmasına seyrek veriproblem denir. Seyrek veri konusu, işbirlikçi filtreleme yönteminde karşılaşılan en önemli sorunlardan biridir.. Bu tez çalışmasında, seyrek very sorununu çözmek veya en azından işbirlikçi filtreleme yönteminin başarısı üzerindeki etkisini azaltmak amacıyla,her kategoriye ait ögelerin yüzdelerini, kullanıcı kategorisi matrisindeki her bir kategorinin ağırlığı olarak kullanarak, eksik puanlarıdeğiştirmek için bir değer oluşturulmuştur. Bu değer kullanılarak, eksik puanların

çoğunluğu oluşturduğu durumlarda seyrek verilerin ortalama mutlak hatası yarıya düşürülmüş ve öneri doğruluğu iyileştirilmiştir.

Anahtar Kelimeler:İŞBİRLİĞİ FİLTRELEME, Öneri sistemleri,Veri seyrekliği

CHAPTER 1

INTRODUCTION

1.1 Literature Review

Many researches were done in the recommendation systems field. Adomavicius and Tuzhilin wrote a review paper [1] of recommendation systems, this paper presents the types of recommendation systems: content based, collaborative based and hybrid methods. Also, it shows the work that has been done on them until that time, the problems that could face the recommendation systems and other further work that can be done. [6] is another review paper of recommendation systems. Content based recommendation systems depend on the contents of the items or the users' features to recommend items, [9] is a review paper that shows the methods that have been used in content-based recommendation systems, the advantages and disadvantages of it. In the hybrid method, the content based method is used along with the collaborative based in order to increase its accuracy and solve the data sparsity and cold-start problems. Dhanashree and Shital [4] present a survey paper on hybrid recommendation systems. It shows the needs of this kind of recommendation systems by introducing the issues in the existing ones, also, it shows the advantages of hybrid systems and the challenges that could face them.

Although the above methods are the most common, but other methods were used in recommendation systems such as the association rules mining [8]. Usually, association rules mining is used to find the items that mostly bought/watched together. Chen, Miller and Dagher [3] proposed a recommendation system for the small online retailers using the association rules mining, this system was efficient and scalable but it did not result in high accuracy. Other researches [10] [15] combine the association rules mining with the

common methods of recommendation algorithms: collaborative filtering and content based filtering in order to increase the accuracy of the recommendations or to solve the recommendation systems problems such as: data sparsity and scalability.

Collaborative filtering is the most common recommendation algorithm, it can be user-based, item-based or model-based. User-based and item-based are called memory-based collaborative filtering as they are depending on the history of the users. Item-based collaborative filtering [13] depends on the ranks of the items to find the relationship between them and then, uses this relationship to generate the recommendations while user-based depends on the ranks to find the relationship between users. As for the model based collaborative filtering, it uses one of the machine learning or data mining algorithms to build a model. This method is fast as it generates multiple recommendations using the pre-build model. Clustering [7] and Bayesian models are ones of the most common in this field.

Multiple researches have been conducted in user-based collaborative filtering field in order to increase its results accuracy and solve its problems like: data sparsity and cold-start. Qin, Cao and Peng [11] proposed a collaborative filtering algorithm depending on dimensional reduction to decrease the effect of the data sparsity. The user-item high dimensional traditional model was converted into user-category low dimensional model. In order to create recommendations for new users, the user attributes were used, such as: age, gender and occupation. Paper [16] is another algorithm that tried to solve the data sparsity problem by dimension reduction, it converted the user-item traditional model into user cloud model, where the similarity was measured between the cloud models in place of users. Singular value decomposition (SVD) [12] is a method makes a reduction of the dimensions of the user rating model. This method was effective in some cases but it was not better than collaborative filtering in the case of high data sparsity.

As collaborative filtering algorithm depends on similarity measures to generate the recommendations and the most common measures, such as: cosine similarity and Pearson

correlation have their drawbacks, some researchers tried to find new similarity measure. [5] is a research that proposed a new similarity algorithm based on calculating the similarity between the target item and the common items between the active user's history and other user's history. [14] is proposing another similarity measure that depends on the users' habits when giving the ranks to items, it is a combination of cosine similarity, Jaccard and Mean Measure of divergence. These new measures have performed better than the popular ones in case of the existence of the sparse data.

This research is proposing a new weighted similarity algorithm depending on the user-category model, the users' ranks and similar users' ranks. Al-Shwal and Al-Shamri [2] proposed a weighting similarity algorithm using a fuzzy variable as a weight. This variable is generated depending on the ratings or the ratings deviation.

1.2 Objective of the Thesis

As mentioned in section 1.1 recommendation systems is an important part of any e-commerce system and collaborative filtering is one of the most common recommendation algorithms, so many researches were done in order to increase its accuracy as this one. In this research, a new weighting similarity algorithm is introduced in order to increase the collaborative filtering accuracy.

In addition, this research is trying to solve one of the biggest problems that facing the collaborative filtering algorithm. This problem is data sparsity or in other words "missing ranks", which means the users did not rank the items they watched or bought. This research tried to solve this problem by generating a value which depends on users' interest to replace the missing ranks, so that the effect of the data sparsity problem on collaborative filtering results will be decreased.

1.3 Hypothesis

Collaborative filtering is predicting the rank that could the active user give to the target item through finding the most similar users to the active user. This happens through taking

into consideration the ranks of the common items between the active user's history and other users' history. Depending only on the ranks of the common items thought to be not enough to decide the similarity between users, users' general taste and interest should be taking into consideration, too. It can be said if two users like the same genres/categories then mostly they will like to watch the same items regardless of their ranks. So, it is thought that using the users' general taste along with the used similarity measures will increase the accuracy of collaborative filtering.

Moreover, in case of the data sparsity problem when the data contains missing ranks and the ranks will not be any more dependable, depending on the users' general taste and interests will be more reasonable. So, to say, using the users' general taste to generate a value to replace the missing ranks could be effective in decreasing the error resulted from their existence.

CHAPTER 2

GENERAL INFORMATION

Nowadays, the humans are intended more to use technologies to save their money, time and effort. E-commerce is one of these technologies. It is a common web technology that offers a shopping service for people, so they are able to buy anything they want, at any time and in anywhere they are. With the increasing number of e-commerce systems and e-commerce users, the need for recommendation systems increased. The recommendation systems are responsible on recommending items to the users that they could like. In this way, both sides: the users and the companies will gain benefits. The users will be able to save their time and effort and companies will be able to increase their profits.

Mainly, recommendation systems have three types [6], see figure 1.1:

- Content based systems
- Collaborative based systems
- Hybrid systems

Content based systems is one of the old methods that has been used in recommendation systems. It depends on the contents to compare and match items or information to the items in the user's history or it could depend on the users' features. The problem with this method is that it can be spammed easily as the contents could be changed easily, too. As for collaborative filtering (CF), it depends on evaluating all the users' history and it has two main types: memory-based CF and model-based CF. Memory-based CF also can be one of these two types: user-based CF and Item-based CF. Model-based CF uses the ranks and one of the machine learning or data mining techniques to build a model then uses this pre-computed model to make recommendations. Hybrid systems merge the two previous

types: content-based and collaborative-based. Usually, this is done to overcome some of the CF problems, such as: cold-start and data sparsity.

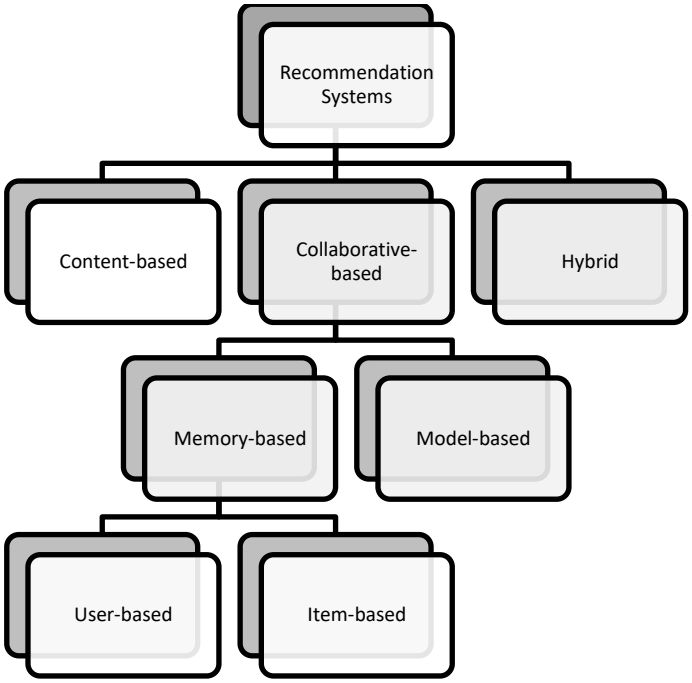

Figure 2.1The Types of Recommendation Systems

This research, is working on user-based CF systems and data sparsity problem. This chapter will show a general view of collaborative filtering, the goal of this research and the hypothesis that this research was based on. In the next chapters, the literature review that have been made before starting this algorithm, the details of the algorithm, the experimental part, the results and the future work will be presented.

2.1 Collaborative Filtering (CF)

Collaborative filtering algorithm is the most common algorithm in recommendation systems. It predicts the rank that could be given to an item from the active user to decide whether to recommend it to the active user or not. It has two types as memory-based and model-based collaborative filtering. The memory-based collaborative filtering can be user-based or item-based. The user-based collaborative filtering depends on finding the most similar users to the active users then used their co-rated items for prediction while the item-based collaborative filtering depends on the similarity between items. User-

based CF is the type that will be used in this research. Equation (1) shows the general formula of the collaborative filtering.

$$P_{u_a}(I_j) = \bar{R}_{u_a} + \frac{\sum_{u \in U} sim(u_a, u) \times (R_{u,j} - \bar{R}_u)}{\sum_{u \in U} |sim(u_a, u)|} \quad (1)$$

Where $P_i(I_j)$ is the prediction of the active user u_a rank to Item I_j , $\bar{R}_{u_a}$ is the mean rank of active useru_a, $sim(u_a, u)$ is the similarity between the users u_a and u ,$R_{u,j}$ is the rank of user u to item j and $\bar{R}_u$ is the mean rank of user u. As can be noticed from equation (1) the similarity measure is a main part of the collaborative filtering algorithm.

2.1.1 Similarity Measures

The similarity measure forms an important part of the collaborative filtering algorithm. Different similarity measures can be used in calculating the predictions. In following the most common similarity measures:

- Pearson Correlation

$$sim(u_a, u) = \frac{\sum_{j \in I_{u_a,u}} (R_{u_a j} - \bar{R}_{u_a})(R_{uj} - \bar{R}_u)}{\sqrt{\sum_{j \in I_{u_a,u}} (R_{u_a j} - \bar{R}_{u_a})^2} \sqrt{\sum_{j \in I_{u_a,u}} (R_{uj} - \bar{R}_u)^2}} \quad (2)$$

- Cosine Similarity

$$sim(u_a, u) = \frac{\overrightarrow{R_{u_a}} \cdot \overrightarrow{R_u}}{\left|\overrightarrow{R_{u_a}}\right| \left|\overrightarrow{R_u}\right|} \quad (3)$$

- Adjusted Cosine Similarity

$$sim(u_a, u) = \frac{\sum_{j \in I_{u_a,u}} (R_{u_a j} - \bar{R}_{u_a})(R_{uj} - \bar{R}_u)}{\sqrt{\sum_{j \in I_{u_a}} (R_{u_a j} - \bar{R}_{u_a})^2} \sqrt{\sum_{j \in I_u} (R_{uj} - \bar{R}_u)^2}} \quad (4)$$

In which, $R_{u_a j}$ and R_{uj} represent the ranks of users u_a and u, respectively, to item j, $\bar{R}_u$ represents the average rank of user u, $\bar{R}_{u_a}$ represents the average rank of user i and $\vec{R}_{u_a}$ and $\vec{R}_u$ represent the user-item model of user u_a and u respectevily. All of the above similarity measures are depending entirely on the user-item traditional high dimensional model to calculate the similarity between users.

2.1.2 The problems that face collaborative filtering

Different problems could face any CF systems, but the most common problems are:

- Data Sparsity: CF algorithms mostly depends on ranks to make recommendation, so data sparsity is one of the biggest problems that facing them as it means the existence of missing ranks. Actually, in the real life, most of the users do not rank the items they watched or bought, so, most of the data on the web is sparse data. [11] claims that only 1% of the data on the internet is ranked which means almost 99% of the data is sparse data which make it a huge problem.
- Cold-Start: cold-start problem happens when new user or new item is added to the system. In this case user-based CF cannot give any recommendation to this new user as he/she does not have any history yet and item-based CF cannot give any recommendation according to this new item as no one rank it yet.

CHAPTER 3

A NEW WEIGHTING APPROACH TO SOLVE DATA SPARSITY PROBLEM IN COLLABORATIVE FILTERING

This thesis presents a novel weighting method that can be used along with the similarity measure or as a similarity measure by its own to increase the collaborative filtering accuracy. This chapter will show how to generate this weight in details using MovieLens 100K dataset.

The chapter begins the model conversion process where the traditional user-item high dimensional model will be converted to the user-category low dimensional model, after that, it shows the algorithm of generating the weight in details and at last, it will show how to deal with the data sparsity problem.

3.1 Model Conversion: User-Item traditional model to User-Category model

This section will show the method of converting the user-item traditional high dimensional model to user-category low dimensional model [11].

3.1.1 User-Item traditional model

User-item matrix is the traditional model that has been used in collaborative filtering based recommendation systems in its both types user-based and item-based. This matrix contains the users and their given ranks to the corresponding items. Let the set of all users in the system be as follows:

U= {u1, u2, u3, …,un}

Where n is the total number of users in the system. Let the set of all items (movies) in the system be represented as follows:

I= { i1, i2, i3 , …, im }

Where m is the total number of items in the system. Then the user-item matrix will be as represented in Table 3.1:

Table 3.1 User-Item matrix

User/Item	i1	i2	i3	…	im
u1					
u2					
u3					
…					
un					

This matrix is a high dimensional matrix specially when the system is used by a large number of users and offers a large number of items.

3.1.2 Item-Category Model

Item-Category model is a matrix constructed for each user. This matrix contains the items that have been watched by the user and the categories. The values in this matrix will indicate whether the item belongs to the category or not. Let the set of all categories be represented as follows:

C= {c1, c2, c3,…, cj}

And the set of all items that have been watched by the user as follows:

In= {i1, i2, i3,…, ib}

Where j is the total number of categories in the system that any item belongs to at least one of them, In is the set of all the items that have been watched by user un and b is the total number of items in set In. The values of this matrix will be filled depending on the following rule:

$$Category_j^b = \begin{cases} 1, & Item_b \in Category_j \\ 0, & Item_b \notin Category_j \end{cases}$$

Then the item-category matrix can be represented as in Table3.2.

Table 3.2 Item-Category matrix

Item/Category	c1	c2	c3	…	Cj
i1					
i2					
i3					
…					
Ib					

This matrix will be used in getting the user-category-weight matrix.

3.1.3 User-Category Model

This model is represented with a matrix that contains the users and the categories. The values in this matrix indicate the weight of each user to the corresponding category. Let the set of the category-weight be represented as follows:

CW= {cw1,cw2,cw3,…,cwj}

Then the matrix will be as shown in Table3.3.

Table 3.3 User-Category-Weight matrix

Item/Category	cw1	cw2	cw3	…	cwj
u1					
u2					
u3					
…					
un					

The dimensions of this matrix will be n x j while the dimensions of user-item matrix will be n x m. As the number of items (m) can be increased every day and the number of categories (j) is a limited number, the dimensions of the user-category matrix are much lower than the dimensions of the user-item matrix.

3.1.4 Conversion Method

This section will present the steps to convert the user-item matrix to the user-category matrix. The general steps are illustrated in Figure 3.1.

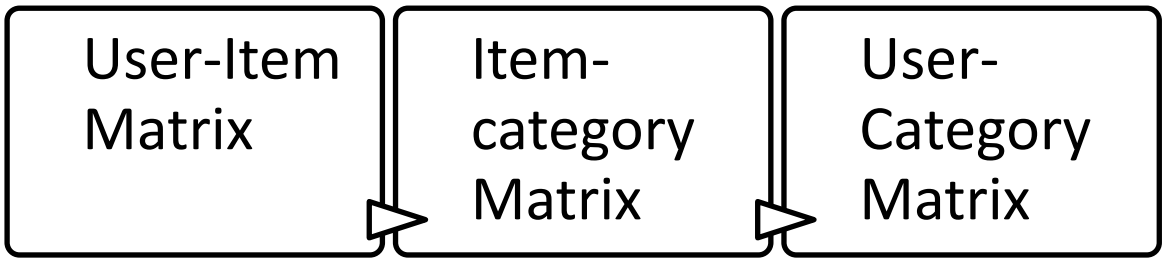

Figure 3.1 General steps of user-item matrix to user-category matrix conversion.

The steps in details are as following:

Step 1: User-Item matrix to Item-Category Matrix.

To convert the user-item matrix to item-category matrix, for each user un:

Find the items that have been watched by user un.

Construct the matrix where the rows represent the items and the columns represent the categories.

For each item, check whether it belongs to any of the categories by taking into consideration that the item can belong to multiple categories. If the item belongs to the corresponding category, then the cell value will be 1, if not, then the cell value will be 0. Table 3.2 in section 3.1.2 represent this matrix.

Step 2: Item-Category Matrix to User-Category Matrix

To convert the item-category matrices to user-category matrix, for each item-category matrix of user un:

Multiply each row with un rank to its corresponding item. Let the set of vectors of the item-category matrix be as follows:

Vn= {V1,n, V2,n , V3,n,…, Vb,n }

Where Vn is the set of vectors of the item-category matrix of user un, b is the number of items that have been watched by user un (section3.1.2) and Vb,n is the category vector (row) of item b in user un item-category matrix. This vector contains zeroes and ones values which indicate whether the item belongs to the corresponding category or not.

Apply a vertical summation over the matrix so that the result will be, for each user, one value for each category.

For each user, divide each value of each category by the number of items that the user has watched from this category. This division is just for normalization. The following equation illustrates the step 2 process:

$$UCM_{u_n} = \sum R_{b,n} \times V_{b,n} \tag{5}$$

Where UCM_{u_n} is the resulted matrix, the user-category matrix which is represented in Table3 in the previous section. $R_{b,n}$ is the rank of user n to item b and Vb,n is the category vector of item b in user un item-category matrix.

3.2 Weight Generation

This section will show the detailed steps for generating the proposed weight. This weight forms the core of the new weighting method which is proposed in this thesis. The generation of the weight will depend mainly on the user-category matrix that has been generated in the previous section. The weight is a product of two parts: similarity and prediction as in the following equation.

$$\text{weight} = sum\text{D}iff(u_a, u_n) \times \text{V} \tag{6}$$

Where $sum\text{D}iff(u_a, u_n)$ represents the similarity part of the weight while V represents the prediction part of it. The following sections show the method of generating each part of the weight.

3.2.1 Generating the similarity part of the weight

$sumDiff(u_a, u_n)$ is the part of the weight that can be used as a similarity measure. This similarity measure calculates the similarity between two users' general interests or taste by calculating the summation of the distance between each user's weight to each category. Let ua be the active user, who will see the recommended items, and un be any user in the system except ua, then $sumDiff(u_a, u_n)$ is the horizontal summation over the differences between the vector of the user-category matrix corresponds to ua (UCW_{ca})and the vector corresponds to un(UCW_{cn}). This is illustrated in the following equation:

$$sumDiff(u_a, u_n) = \sum_{c \in C} |UCW_{ca} - UCW_{cn}| \quad (7)$$

Where c is a category in the set of categories C. The example in Figure 3.2 shows clearly the process of generating $sumDiff(u_a, u_n)$.

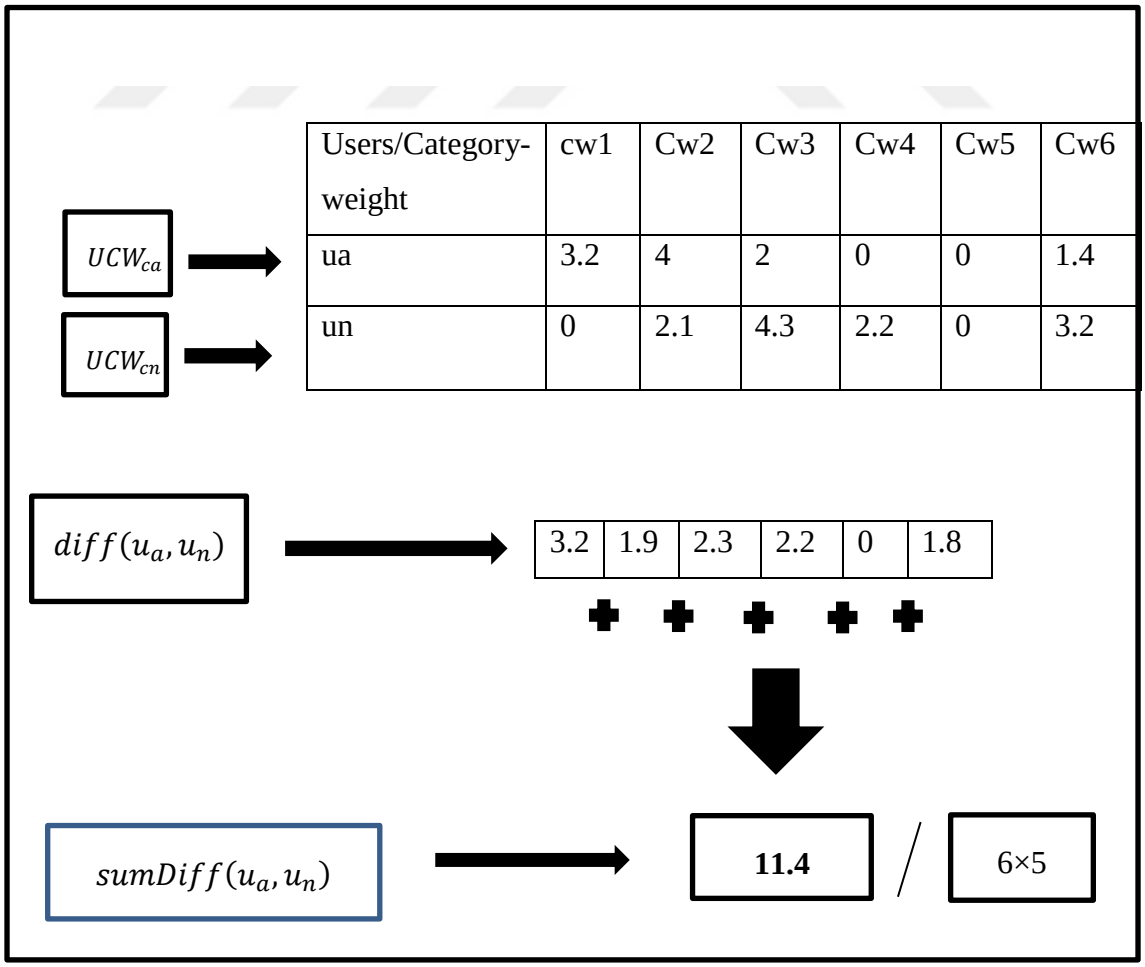

Figure 3.2 An example of calculating the similarity part of the weight

For normalization, $sumDiff(u_a, u_n)$ is divided by the number of categories multiplied by the highest possible rank.

3.2.2 Generating the prediction part of the weight

V is the summation of three values. The generating of these values depends on the categories that the target item belongs to. Using these categories, the values will depend on the similar users' weights of these categories, the active user weight of them and the average rank of active user items that belong to them. The resulted value of the summation of these three values (V) can be used as a prediction by its own. Actually, the similarity part of the weight can be concerned as a weight for this prediction. Let ua be the active user and un is any user in the system except ua, then V will be as follows:

$$V = v_1(u_a) + v_2(u_a, u_n) + v_3(u_a) \quad (8)$$

Let the set of all items that their ranks will be predicted for ua be represented as follows:

I_p= {i1,i2,i3,…,ip}

Where p is the total number of the items which their ranks will be predicted for ua. Then these three values are calculated as follows:

- Calculating $v_1(u_a)$

For each item i in Ip apply the following:

1. From ua history, extract the set of items that belong to the same category as item i. Let call it T.
2. Add the ranks of the items in T together.
3. For normalization, divide the number resulted in 2 by the number of items in T.

These steps can be illustrated in the following equation:

$$v_1(u_a) = \frac{1}{N_T} \sum_{t \in T, c \in C} R_{at} \quad (9)$$

Where R_{at} is the rank of user ua to item t, t is any item in set T, NT is the number of items in T and c is one of the categories that item i belongs to (set C).

- Calculating $v_2(u_a, u_n)$.

This value is depending on extracting the similar users to ua. The similar users are found using sum-diff where the lower value means a higher similarity. Let the set of all similar users to ua be represented as follows:

Us= {u1,u2,u3,…,us}

Where s is the total number of the similar users to ua. Then the value will be calculated as following:

1. From the users' history in Us, extract the items that belong to the same category as item i. Let it be represented as Ts.
2. Add the ranks of the items in Ts together.
3. For normalization, divide the number resulted in 2 by the number of items in Ts.

The following equation illustrate the above steps:

$$v_2(u_a, u_n) = \frac{1}{N_{T_s}} \sum_{u \in U_s} \sum_{t \in T_s, c \in C} R_{ut} \qquad (10)$$

Where R_{ut} is the rank of user u to item t and N_{T_s} is the total number of items in Ts.

- Calculating $v_3(u_a)$

This value is calculating depending on the user-category matrix. Let the vector of user-category matrix that belongs to ua be represented as UCWa. For each item i in Ip apply the following:

1. Construct the category set C of i which contains all the categories that item i belongs to.
2. Extract the weights of the categories in C from UCWa and add them together.
3. For normalization, divide the result in 2 by the number of categories in C.

This can be illustrated in the following equation:

$$v_3(u_a) = \frac{1}{N_C} \sum_{c \in C} UCW_{ac} \tag{11}$$

Where N_C is the total number of categories that item i belongs to and UCW_{ac} is the category weight of user ua to the category c.

3.3 Data Sparsity Problem

Data sparsity means missing data and in this case missing ranks. It is one of the main problems that face the recommendation systems because in the real life most users do not rank most of the items that they watched or bought. This problem affect the accuracy of the recommendation system and in some cases make the system unable to give recommendations. This section will propose a new method to lower the effect of the missing ranks by replacing them with a value.

3.3.1 Models Conversion: User-item Model to User-Category Model

In the case of missing ranks, we can not depend on the ranks in model conversion process especially when there are a huge percentage of the missing ranks in the data set, so rather than depending on the ranks it will depend totally on the user interest. This is constructed on the assumption that if a user like a category the most then the movies that he is watching will mostly belong to that category.

In this case, the user-category matrix will be generated as follows:

For each user in the user set Un, for each category c in C, extract the items that belong to c and let it be represented as Inc.

The weight of each category will be the percentage of the items that belong to it from the total of the items that the user watched. The following equation illustrate this:

$$category - weight = \frac{N_{I_{nc}}}{N_n} \tag{12}$$

Where $N_{I_{nc}}$ is the number of items that user n watched and belong to the category c and N_n is the number of items that have been watched by user un.

3.3.2 Generating the replacement value of the missing rank

The replacement value which will be extracted from the user-category matrix, will replace the missing rank in order to increase the collaborative filtering accuracy and try to decrease the effect of the missing ranks on the collaborative filtering. To generate this value both the user-category matrix which has been generated in the previous section and the ranks of both the active user ua and the similar users, if exist, will be used. This value is a compilation of multiple values and the following formula illustrates it:

$$R_{v=}\left(cat\_weight(u_a)\times 5 + avr\_sim\_cat\_weight\times 5\right)/2 + ((1 - cat\_weight(u_a)) + (1 - avr\_sim\_cat\_weight) \tag{13}$$

Where $cat\_weight(u_a)$ is the user ua category weight of the categories of item I and $avr\_sim\_cat\_weight(u_a, u_n)$ is the similar users' category weight of the categories of item i.

Finding the similar users: The similar users here also will be found according to $sum\mathrm{D}iff(u_a, u_n)$, the only change is the way of calculating the user-category matrix as shown in the previous sections.

Calculating $\boldsymbol{cat\_weight(u_a)}$:This value is calculated in the same way of calculating $v_3(u_a)$ in section 3.2.2, the only deference is the way of calculating the user-category matrix.

Calculating $\boldsymbol{sim\_cat\_weight(u_a, u_n)}$: To calculate this value apply the following:

For the active user ua, extract the similar users and let it be represented as Us.

1. Let the set of all the categories of item i be represented as set C.
2. Make vertical summation over the weights of the categories in C in the user-category matrix that corresponds to the users in Us.
3. Sum the results of 3.

For normalization, divide the result of 4 by the number of the categories in set C.

CHAPTER 4

RESULTS AND DISCUSSION

The previous chapter presents the algorithm of generating the weight and a way to deal with the data sparsity problem. In order to show the effect of the weight on the collaborative filtering accuracy, this chapter, will show the experimental part.

This chapter begins with describing the scenarios of the experiment and their conditions, then it goes through the results of each scenario, finally, it presents a comparison between the results of this research and other researches.

4.1 The Scenarios

The experimental part can be divided into two parts: First part is to measure the effectiveness of the weight on the collaborative filtering accuracy using the original dataset and the second part is to measure the effectiveness of the weight in case of the existence of the missing ranks with replacing them with a value. Each part of these has its own scenarios. The scenarios for the first part of the experiment were as follows:

1. Using part of the weight as a similarity measure. This part is $sumDiff(u_a, u_n)$ (section 3.2.1).
2. Using the weight with the similarity measure.
3. Changing the number of similar users that can be taking into consideration when calculating the predictions. The number of users that have been used are: 4,8,12,16,20 for 100K-MovieLens dataset and 10,20,30,40,50 for 1M-MovieLens dataset.

While the scenarios of the second part of the experiment were as follows:

1. Changing the percentage of the missing ranks. The percentages that have been used are: 10%, 30%, 60% and 90%.
2. For each percent in step 1, repeat the scenarios of the first part of the experiment before and after replacing the missing ranks with a value. This value is generated according to the algorithm in section 3.3.

4.1.1 The Dataset

The data set that have been used to apply this experiment is MovieLens 100K dataset. It contains 100,000 ranks from 943 users to 1682 movies. Each user has ranked at least 20 movies, also, each movie can belong to one or more of the 19 categories in this dataset. To test the algorithm using this dataset, 5-fold cross validation method is used using the already divided datasets on the MovieLens website. When constructing the user-item matrix of this dataset, lots of sparse data will exist as there 1682 but the users have ranked at least 20 movies. The sparse rate can be calculated as following:

$$1 - \frac{100{,}000}{943 \times 1682} = 0.9369$$

In addition, MovieLens 1M dataset is used to test our proposed algorithm in order to compare the result of this algorithm with other algorithms which used this dataset. This dataset contains 1000209 ratings from 6040 users to approximately 3900 movies. To test the MovieLens 1M dataset, first, the dataset is divided into three subsets []: First subset contains the users who have less than 101 ranks, the second one contains the users who have number of ranks more than 100 and less than 501 and the third one contains the users who have more than 500 ranks. Then a subset of 1000 users is created by taking 50% of the users from the first dataset, 40% of the users from the second one and the rest from the third one. After that, 5-fold cross validation is applied by dividing the dataset into test and train datasets using 2:8 division rate.

4.1.2 The similarity measure

There are multiple similarity measures that can be used in collaborative filtering algorithm as mentioned in chapter 1. The most common similarity measures are Pearson correlation and cosine similarity. The one that has been chosen to be used in this experiment is the cosine similarity. The reasons behind that are, cosine similarity is faster than Pearson correlation and in some cases, even if two users have similar ranks to the same items, Pearson correlation will give a low similarity and vice versa.

4.1.3 Calculating the error

In order to calculate the error and the accuracy of the algorithm mean absolute error (MAE) is used. MAE make summation over the difference between the predicted rank and the real rank. Assume the predicted rank of the item I as pi and the real rank as ri, then:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|p_i - r_i| \tag{14}$$

Where N is the number of ranks.

4.2 The Results

This section will show the results of each part of the experiment.

4.2.1 The results of using the weight with the original dataset

Three ways were used to test the effectiveness of the weight on collaborative filtering algorithm. First (Test A): using the cosine similarity to select the most similar users then multiplying it with their weights, the predicted rank formula will be as follows:

$$P_{u_a}(I_j) = \bar{R}_{u_a} + \frac{\sum_{u_n \in U} w \times sim(u_a, u_n) \times (R_{u_n,j} - \bar{R}_{u_n})}{\sum_{u_n \in U} |sim(u_a, u_n)| \times sumDiff(u_a, u_n)} \quad (15)$$

The second way (Test B): using the first part of the weight $sumDiff(u_a, u_n)$ as a similarity measure to select the most similar users then multiplied it with the cosine similarity and the second part of the weight V.

$$P_{u_a}(I_j) = \bar{R}_{u_a} + \frac{\sum_{u_n \in U} sumDiff(u_a, u_n) \times \cos\_V \times (R_{u_n,j} - \bar{R}_{u_n})}{\sum_{u \in U} |cos(u_a, u_n)| \times sumDiff(u_a, u_n)} \quad (16)$$

$$cos\_V = \cos(u_a, u_n) \times V \quad (17)$$

Where U is the set of similar users to the active useru_a, $sim(u_a, u_n)$ and $\cos(u_a, u_n)$ both represent the cosine similarity and w is the weight. The third one (Test C): using the part of the weight ($sumDiff(u_a, u_n)$) as the similarity measure in place of the cosine similarity.

$$P_{u_a}(I_j) = \bar{R}_{u_a} + \frac{\sum_{u_n \in U} sumDiff(u_a, u_n) \times (R_{u_n,j} - \bar{R}_{u_n})}{\sum_{u_n \in U} |sumDiff(u_a, u_n)|} \quad (18)$$

The results are shown in Table4.1 and Figure 4.1.

Table 4.1MAE results of using the weight in collaborative filtering for 100K-Movielens dataset

No. of sim users	Traditional CF	Test C	Test A	Test B
4	0.8308	0.81177	0.82998	0.81084
8	0.78961	0.77879	0.7888	0.77763
12	0.7755	0.76779	0.77469	0.76647
16	0.76853	0.7627	0.76770	0.76135
20	0.76384	0.75906	0.763	0.75765

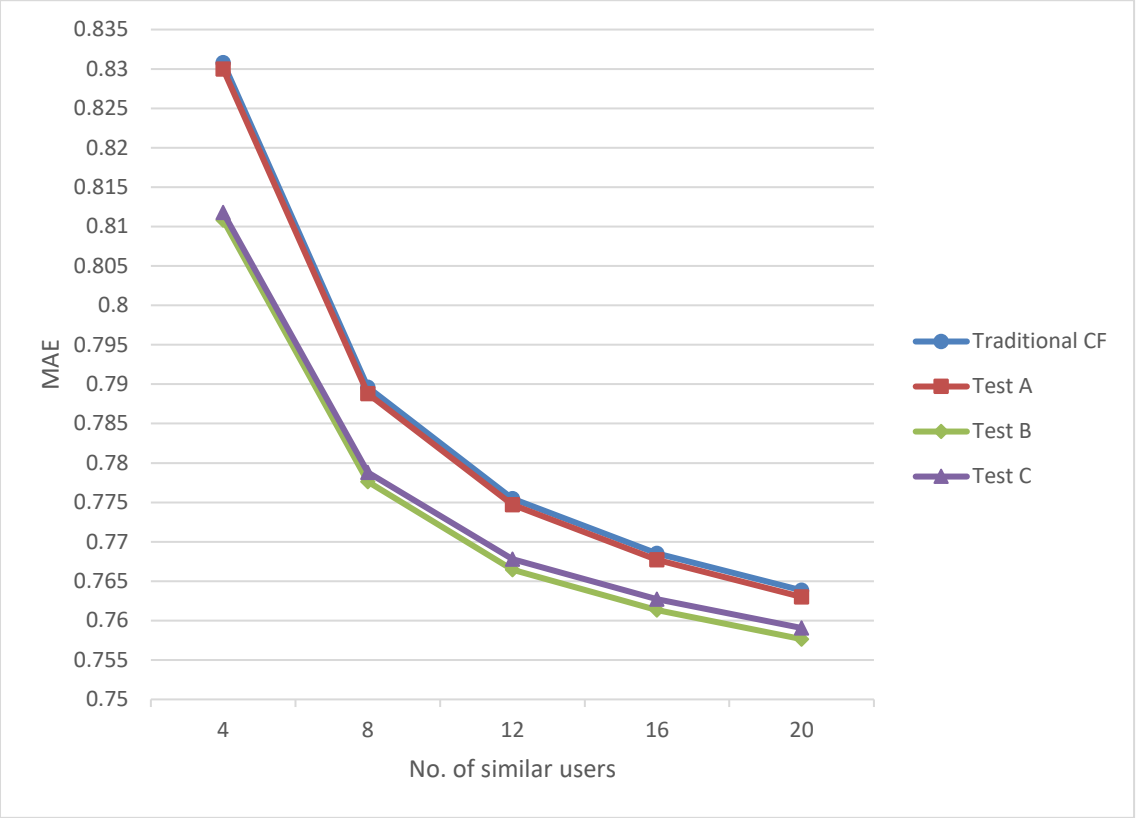

Figure 4.1MAE results of using the weight in collaborative filtering for 100K-movieLens dataset.

Table 4.2 MAE results of using the weight for MovieLens 1M dataset

Number of similar users	Test C	Test A	Test B
10	0.75354	0.76846	0.75325
20	0.74367	0.75136	0.74293
30	0.74243	0.74579	0.7416
40	0.74221	0.74339	0.74142
50	0.74278	0.74242	0.74202

4.2.2 The results of using the weight with missing ranks

To test the effectiveness of the weight in case of missing ranks (where the rank equal to 0), the percentage of the missing ranks in the dataset is changed and then, the scenarios that have been used in case of no missing ranks is repeated for each percent. In addition, to test the effectiveness of the value that replaces the missing ranks, these scenarios are repeated for each percent two times, one with the existence of the missing ranks and the other after replacing them with the value. The results of each percent before and after replacing the missing ranks is represented in tables 4.3-4.6.

- 10% missing ranks of the dataset

Table 4.3 MAE before replacing the missing ranks with the value in case of 10% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	1.01787	0.96882	1.01697	0.96939
8	0.94372	0.90142	0.9429	0.90128
12	0.91688	0.87745	0.91612	0.87717
16	0.90253	0.86436	0.90181	0.86397
20	0.89274	0.85768	0.89206	0.85717

Table 4.4 MAE after replacing the missing ranks with the value in case of 10% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	0.86344	0.84059	0.86258	0.8406
8	0.81248	0.79883	0.81164	0.79813
12	0.79494	0.78491	0.79413	0.78382
16	0.78616	0.77726	0.78532	0.77614
20	0.78059	0.77277	0.77974	0.77163

- 30% missing ranks of the dataset.

Table 4.5 MAE before replacing the missing ranks with the value in case of 30% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	1.46053	1.38111	1.46013	1.38742
8	1.38035	1.30394	1.38016	1.30947
12	1.3498	1.28136	1.34983	1.28556
16	1.33277	1.26872	1.33296	1.27265
20	1.32098	1.25960	1.32137	1.26329

Table 4.6 MAE after replacing the missing ranks with the value in case of 30% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	0.94484	0.91577	0.94434	0.97498
8	0.88303	0.86497	0.88252	0.86543
12	0.86149	0.84710	0.86103	0.84737
16	0.85058	0.8384	0.85012	0.83804
20	0.84392	0.83427	0.84348	0.83384

- 60% missing ranks of the dataset.

Table 4.7 MAE before replacing the missing ranks with the value in case of 60% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	2.25449	2.20688	2.25307	2.20063
8	2.21704	2.17004	2.21524	2.17828
12	2.20526	2.15745	2.2035	2.16642
16	2.19924	2.15118	2.19748	2.15955
20	2.19405	2.14701	2.19261	2.15609

Table 4.8 MAE after replacing the missing ranks with the value in case of 60% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	1.11381	1.08054	1.11329	1.23699
8	1.05518	1.03409	1.05472	1.03626
12	1.03665	1.01780	1.03620	1.02279
16	1.02620	1.0089	1.02576	1.01312
20	1.01968	1.00400	1.01926	1.00464

- 90% missing ranks of the dataset.

Table 4.9 MAE before replacing the missing ranks with the value in case of 90% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	3.08044	3.14355	3.03352	2.29027
8	3.07754	3.15519	3.02992	2.61648
12	3.07545	3.16122	3.02807	2.74067

16	3.07473	3.16477	3.02743	2.81015
20	3.07444	3.16963	3.02714	2.85318

Table 4.10 MAE after replacing the missing ranks with the value in case of 90% missing ranks.

No. of sim users	Traditional CF	Test C	Test A	Test B
4	1.41761	1.39495	1.41676	1.40199
8	1.39596	1.36978	1.3951	1.37646
12	1.38902	1.35951	1.38833	1.37042
16	1.38472	1.35355	1.38420	1.35948
20	1.38164	1.34886	1.38128	1.35169

4.2.3 Discussion

As can be noticed in the previous section, using the weight as a similarity measure to choose the most similar users then multiplying it with the cosine similarity gives the highest accuracy. In contrast, the traditional collaborative filtering with using the cosine similarity as a similarity measure without using the weight gives the lowest accuracy. The second highest accuracy was gotten by using the weight as a similarity measure without using the cosine similarity. It is thought that this is because of the weight which represent the users' interests. Giving the priority to the user interests over the ranks seems to give higher accuracy as it depends on the assumption that if two users have the same interests, then most likely they will like to watch the same movies.

In case of the missing ranks, in general, replacing the missing rank with the value gives better results than testing the dataset with missing ranks. The results were acceptable for the small percentages of missing ranks but for the high percentages are not, even though replacing the missing ranks with the value was able to decrease the MAE to the half for these cases, but it still more than 100%. In addition, using the weight as a similarity measure- with or without the cosine similarity- give the best results before and after replacing the missing ranks with the value except for 90% which using the weight as a

similarity method gives the worse results. In most cases, the Test C and Test B methods give very close results where Test A and traditional collaborative filtering results were very close. This can be because the Test A method and the traditional collaborative filtering are giving the priority to the cosine similarity while the others two methods give the priority to the weight.

4.3 The Comparison with other researches

The basic idea of this research which is converting the user-item traditional model to user-category model was taken from paper [11], this paper generated MAE almost between 0.81 and 0.76 by using the 100K MovieLens dataset and changing the number of similar users from 4 to 20 as in this experiment while this research generated MAE almost between 0.81 and 0.757 using the same dataset and scenarios. This means this research improved the results by almost 0.3% when the most 20 similar users are taken into consideration.

As for other weighted similarity measure algorithms such as [2], this research achieved better results especially in cases of low similar users' number. In [2], a fuzzy weightings scheme where used to improve the accuracy of collaborative filtering. By changing the number of similar users from 10 to 50 with addition of 10 each time, a MAE between 2.7943 and 0.8731 was obtained when the fuzzy weighting depends on the rating deviation and a MAE between 3.0016 and 0.9425 when it depends on the ratings. Using the same dataset (MovieLens 1M) and scenarios, a MAE between 0.75 and 0.74 was obtained by weight-sim algorithm of this research.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter presents the conclusion of the methodologies in this research and their results also, it presents the future work that can be done on these methodologies or by using them in addition to other problems of the collaborative filtering that need to be solved.

5.1 Conclusion

Similarity measure is an important part of any user-based collaborative filtering algorithm where it represents the weight of the similar users' rank to the target item. Usually, this measure calculates the similarity between users depending on their ranks to the mutual items. Taking into consideration just the mutual items, it is thought to be not enough because it discards the general taste or interest of the users, although the users do not have mutual items or a big number of them but most of that they watched is from the same categories, so they are similar because they have similar general interests. This assumption is the base of this research.

This research tried to find a new weighting methodology depends on the general interests of the users to be used along with the existing similarity measures in order to improve the accuracy of collaborative filtering algorithm. This weight is formed of two parts: the first part represent a similarity measure by its own and it is generated by taking the summation of the differences between the user-category matrices, and the second part represent a prediction, this prediction is generated by taking the average rank of the similar users to the target item, the average ranks of the active user to the items that belongs to the same

categories as the target item and the average rank of the similar users to the items that belong to the same categories as the target item.

As mentioned above, one part of the weight can be used as a similarity measure, using this part $sum\mathrm{D}iff(u_a, u_n)$ (see section 3.2.1) as a similarity measure to choose the most similar users then multiplied the weight that corresponds to these users with the common similarity measure (cosine similarity) gives the highest accuracy and it improves the collaborative filtering accuracy to a good extent.

This research also presents a methodology to solve the data sparsity problem by replacing the missing rank with a value. This value is generated depending on the user-category matrix but in place of using the ranks to find the weights of the categories for each user, the percentages of the items that belong to each category are used for this goal. This method resulted acceptable results for the low and medium percentages of missing data but not for the high ones.

5.2 Future Work

This algorithm was tested mainly using 100k-MovieLens dataset and 1M-MovieLens dataset was used in the comparison with other weighted algorithms. 100-K dataset can be concerned as a small dataset so its results could be inaccurate. In order to get a more accurate and clear results, this algorithm should be tested on huger datasets such as 10M and 20M MovieLens datasets which need time and resources.

As for the data sparsity problem, even though this research finds a solution by replacing the missing rank with a value but it is still not enough and it needs more work on this part in order to get better results especially in case of huge sparsity. That is the researches proves that most of the data on the internet has a huge percent of the missing ranks as most of the users do not rank the items they watched or bought.

Other problem that face the recommendation systems and need a solution is the cold-start problem which happened when a new user enters the system. This user didn’t

watch/bought anything before, so it will be hard or even impossible to decide his/her similar users depending on the ranks or other users' history. This research algorithm can be used in case that there is a user who watched movies but he/she didn't rank any by depending on the categories of the items that he/she watched. This user in the traditional collaborative filtering, which depends only on the ranks to find the similar users, will be concerned as a new user. As for the new users with zero history, more work should be done.

REFERENCES

[1] Adomavicius, G. And Tuzhilin, A., (2005). "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", IEEE Transactions on Knowledge and Data Enginerring, 17(6):734–749.

[2] Al-Shamri, M.Y.H. And Al-Ashwal, N.H., (2014). "Fuzzy-weighted similarity measures for memory-based collaborative recommender system", Journal of Intelligent Learning System and Applications, 6(1):1-10.

[3] Chen, J., Miller, C. and Dagher, G., (2014). "Product Recommendation System For Small Online Retailers Using Association Rules Mining", International Conference On Innovation Design And Manufacturing.

[4] Dhanashree, P. and Shital, K., (2017). "Survey on Hybrid Recommendation System with Review Helpfulness Features". International Advanced Research Journal in Science, Engineering and Technology, 4(4):25-27.

[5] Huang, B.H. and Dai, B.R., (2015). "A Weighted Distance Similarity Model to Improve the Accuracy of Collaborative Recommender System", 16th IEEE International Conference on Mobile Data Management.

[6] Isinkaye, F.O., Folajimi, Y.O. and Ojokoh, B.A., (2015). "Recommendation systems: Principles, methods and evaluation", Egyptian Informatics Journal, 16(3):261-273.

[7] Joshi, R.C. and Paswan, R.S., (2015). "A survey paper on clustering-based collaborative filtering approach to generate recommendation", International Journal of Science and Research, 4(1):1395-1398.

[8] Kumbhare, T.A. and Chobe, S.V., (2014). "An Overview of Association Rule Mining Algorithms", International Journal of Computer Science and Information Technologies, 5(1):927-930.

[9] Mandave, D. and Pole, G.S., (2016). "A Review on Content Based Recommendation System", International Journal of Innovative Research in Computer and Communication Engineering, 4(11):20473-20479.

[10] Parvatikar, S. and Joshi, B., (2015). "Online Book Recommendation System by Using Collaborative Filtering and Association Mining", IEEE International Conference on Computational Intelligence and Computing Research.

[11] Qin, J., Cao, L. and Peng, H., (2016). "Collaborative Filtering Recommendation Algorithm Based on Weighted Item Category", 28th Chinese Control and Decision Conference, IEEE.

[12] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., (2000). “Application of Dimensionality Reduction in Recommender System - A Case Study”, Proceeding of the ACM WebKDD, workshop.

[13] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., (2001). “Item-based collaborative filtering recommendation algorithms”, Proceedings of the 10th international conference on World Wide Web, ACM.

[14] Suryakant and Mahara, T., (2016). “A New Similarity Measure Based on Mean Measure of Divergence for Collaborative Filtering in Sparse Environment”, Twelfth International Multi-Conference on Information Processing.

[15] Tewari, A.S., Kumar, A. and Barman, A.G., (2014). “Book Recommendation System Based On Combine Features of Content Based Filtering, Collaborative Filtering and Association Rule Mining”, IEEE.

[16] Zheng, G.W., Li, D.Y., et, (2007). “A collaborative filtering recommendation algorithm based on cloud model”, Journal of Software,18(10):2403-2411.

APPENDIX-A

MOVIELENS-100K DATASET README

SUMMARY & USAGE LICENSE

==========

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. Detailed descriptions of the data file can be found at the end of this file.

Neither the University of Minnesota nor any of the researchers involved can guarantee the correctness of the data, its suitability for any particular purpose, or the validity of results based on the use of the data set. The data set may be used for any research purposes under the following conditions:

* The user may not state or imply any endorsement from the University of Minnesota or the GroupLens Research Group.

* The user must acknowledge the use of the data set in publications resulting from the use of the data set (see below for citation information).

* The user may not redistribute the data without separate permission.

* The user may not use this information for any commercial or

revenue-bearing purposes without first obtaining permission from a faculty member of the GroupLens Research Project at the University of Minnesota.

If you have any further questions or comments, please contact GroupLens <grouplens-info@cs.umn.edu>.

CITATION

==========

To acknowledge use of the dataset in publications, please cite the following paper:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

DOI=http://dx.doi.org/10.1145/2827872

ACKNOWLEDGEMENTS

==========

Thanks to Al Borchers for cleaning up this data and writing the

accompanying scripts.

PUBLISHED WORK THAT HAS USED THIS DATASET

==========

Herlocker, J., Konstan, J., Borchers, A., Riedl, J.. An Algorithmic

Framework for Performing Collaborative Filtering. Proceedings of the

1999 Conference on Research and Development in Information

Retrieval. Aug. 1999.

FURTHER INFORMATION ABOUT THE GROUPLENS RESEARCH PROJECT

==========

The GroupLens Research Project is a research group in the Department

of Computer Science and Engineering at the University of Minnesota.

Members of the GroupLens Research Project are involved in many

research projects related to the fields of information filtering,

collaborative filtering, and recommender systems. The project is lead by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992, but is most well known for its world wide trial of an automated collaborative filtering system for Usenet news in 1996. The technology developed in the Usenet trial formed the base for the formation of Net Perceptions, Inc., which was founded by members of GroupLens Research. Since then the project has expanded its scope to research overall information filtering solutions, integrating in content-based methods as well as improving current collaborative filtering technology.

Further information on the GroupLens Research project, including research publications, can be found at the following web site:

http://www.grouplens.org/

GroupLens Research currently operates a movie recommender based on collaborative filtering:

http://www.movielens.org/

DETAILED DESCRIPTIONS OF DATA FILES

==========

Here are brief descriptions of the data.

ml-data.tar.gz -- Compressed tar file. To rebuild the u data files do this:

gunzip ml-data.tar.gz

tar xvf ml-data.tar

mku.sh

u.data -- The full u data set, 100000 ratings by 943 users on 1682 items.
Each user has rated at least 20 movies. Users and items are
numbered consecutively from 1. The data is randomly
ordered. This is a tab separated list of

user id | item id | rating | timestamp.

The time stamps are unix seconds since 1/1/1970 UTC

u.info -- The number of users, items, and ratings in the u data set.

u.item -- Information about the items (movies); this is a tab separated list of

movie id | movie title | release date | video release date |

IMDb URL | unknown | Action | Adventure | Animation |

Children's | Comedy | Crime | Documentary | Drama | Fantasy |

Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi |

Thriller | War | Western |

The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once.

The movie ids are the ones used in the u.data data set.

u.genre -- A list of the genres.

u.user -- Demographic information about the users; this is a tab separated list of

user id | age | gender | occupation | zip code

The user ids are the ones used in the u.data data set.

u.occupation -- A list of the occupations.

u1.base -- The data sets u1.base and u1.test through u5.base and u5.test

u1.test are 80%/20% splits of the u data into training and test data.

u2.base Each of u1, ..., u5 have disjoint test sets; this if for

u2.test 5 fold cross validation (where you repeat your experiment

u3.base with each training and test set and average the results).

u3.test These data sets can be generated from u.data by mku.sh.

u4.base

u4.test

u5.base

u5.test

ua.base -- The data sets ua.base, ua.test, ub.base, and ub.test

ua.test split the u data into a training set and a test set with

ub.base exactly 10 ratings per user in the test set. The sets

ub.test ua.test and ub.test are disjoint. These data sets can

be generated from u.data by mku.sh.

allbut.pl -- The script that generates training and test sets where

all but n of a users ratings are in the training data.

mku.sh -- A shell script to generate all the u data sets from u.data.

APPENDIX-B

MovieLens-1M dataset ReadMe File

SUMMARY

==========

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

USAGE LICENSE

==========

Neither the University of Minnesota nor any of the researchers involved can guarantee the correctness of the data, its suitability for any particular purpose, or the validity of results based on the use of the data set. The data set may be used for any research purposes under the following conditions:

* The user may not state or imply any endorsement from the
University of Minnesota or the GroupLens Research Group.

* The user must acknowledge the use of the data set in
publications resulting from the use of the data set
(see below for citation information).

* The user may not redistribute the data without separate
permission.

* The user may not use this information for any commercial or
revenue-bearing purposes without first obtaining permission
from a faculty member of the GroupLens Research Project at the
University of Minnesota.

If you have any further questions or comments, please contact GroupLens
<grouplens-info@cs.umn.edu>.

CITATION

==========

To acknowledge use of the dataset in publications, please cite the following paper:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

ACKNOWLEDGEMENTS

==========

Thanks to Shyong Lam and Jon Herlocker for cleaning up and generating the data set.

FURTHER INFORMATION ABOUT THE GROUPLENS RESEARCH PROJECT

==========

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering, collaborative filtering, and recommender systems. The project is lead by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992, but is most well known for its world wide trial of an automated collaborative filtering system for Usenet news in 1996. Since then the project has expanded its scope to research overall information filtering solutions, integrating in content-based methods as well as improving current collaborative filtering technology.

Further information on the GroupLens Research project, including research publications, can be found at the following web site:

http://www.grouplens.org/

GroupLens Research currently operates a movie recommender based on collaborative filtering:

http://www.movielens.org/

RATINGS FILE DESCRIPTION

==========

All ratings are contained in the file "ratings.dat" and are in the

following format:

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040

- MovieIDs range between 1 and 3952

- Ratings are made on a 5-star scale (whole-star ratings only)

- Timestamp is represented in seconds since the epoch as returned by time(2)

- Each user has at least 20 ratings

USERS FILE DESCRIPTION

==========

User information is in the file "users.dat" and is in the following

format:

UserID::Gender::Age::Occupation::Zip-code

All demographic information is provided voluntarily by the users and is

not checked for accuracy. Only users who have provided some demographic

information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female

- Age is chosen from the following ranges:

    * 1: "Under 18"

    * 18: "18-24"

    * 25: "25-34"

    * 35: "35-44"

    * 45: "45-49"

    * 50: "50-55"

    * 56: "56+"

- Occupation is chosen from the following choices:

    * 0: "other" or not specified
    * 1: "academic/educator"
    * 2: "artist"
    * 3: "clerical/admin"
    * 4: "college/grad student"
    * 5: "customer service"
    * 6: "doctor/health care"
    * 7: "executive/managerial"
    * 8: "farmer"
    * 9: "homemaker"
    * 10: "K-12 student"
    * 11: "lawyer"
    * 12: "programmer"
    * 13: "retired"
    * 14: "sales/marketing"
    * 15: "scientist"
    * 16: "self-employed"

* 17: "technician/engineer"

* 18: "tradesman/craftsman"

* 19: "unemployed"

* 20: "writer"

MOVIES FILE DESCRIPTION

==========

Movie information is in the file "movies.dat" and is in the following

format:

MovieID::Title::Genres

- Titles are identical to titles provided by the IMDB (including

year of release)

- Genres are pipe-separated and are selected from the following genres:

* Action

* Adventure

* Animation

* Children's

* Comedy

* Crime

* Documentary

* Drama

* Fantasy

* Film-Noir

* Horror

* Musical

* Mystery

* Romance

* Sci-Fi

* Thriller

* War

* Western

- Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries

- Movies are mostly entered by hand, so errors and inconsistencies may exist

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname : Tasnim ZAYET

Date of birth and place : 4.May.1991, Abu-Dhabi, U.A.E.

Foreign Languages :English, Turkish

E-mail :tasneem.zayet@gmail.com

EDUCATION

Degree	Department	University	Date of Graduation
Undergraduate	Computer Eng.	Palestine Technical University (Kadoorie)	Jan.2014
High School	Scientific stream	Al-Adawiah Sec.bGirls School	Jun.2009

PUBLISHMENTS

Conference Papers

1. **ACRS: Arabic Character Recognition System, 7th annual undergraduate research projects on applied computing, UAE, 2015.**

2. **A new weighting algorithm for collaborative filtering, 25th Signal processing and communications applications conference, Antalya, Turkey, 2017.**

AWARDS

1. **Best oral presentation, 7th annual undergraduate research projects on applied computing, 2015.**