

Effects of Binary Vectors Similarities on the Accuracy of Multi-Criteria Collaborative Filtering

 Burcu Demirelli Okkaloğlu¹

¹Yalova University, Faculty of Engineering, Computer Engineering Department;
bdokkalioglu@yalova.edu.tr

Received 16 June 2021; Revised 13 September 2021; Accepted 19 September 2021; Published online: 31 December 2021

Abstract

Recommender systems offer tailored recommendations by employing various algorithms, and collaborative filtering is one of the well-known and commonly used of those. A traditional collaborative filtering system allows users to rate on a single criterion. However, a single criterion may be insufficient to indicate preferences in domains such as restaurants, movies, or tourism. Multi-criteria collaborative filtering provides a multi-dimensional rating option. In similarity-based multi-criteria collaborative filtering schemes, existing similarity methods utilize co-users or co-items regardless of how many there are. However, a high correlation with a few co-ratings does not always provide a reliable neighborhood. Therefore, it is very common, in both single- and multi-criteria collaborative filtering, to weight similarities with functions utilizing the number of co-ratings. Since multi-criteria collaborative filtering is yet growing, it lacks a comprehensive view of the effects of similarity weighting. This work studies multi-criteria collaborative filtering and the literature of binary vector similarities, which are frequently used for weighting, by giving a related taxonomy and conducts extensive experiments to analyze the effects of weighting similarities on item- and user-based multi-criteria collaborative filtering. Experimental findings suggest that prediction accuracy of item-based multi-criteria collaborative filtering can be boosted by especially binary vector similarity measures which do not consider mutual absences.

Keywords: multi-criteria, collaborative filtering, similarity-weighting, binary vector similarity

1. Introduction

The number of people who have access to the Internet has been rapidly increasing. In parallel, e-commerce companies are increasing both in number and size. At this point, online companies like Amazon and Netflix utilize personalized recommendations to offer the right products to their customers. Recommender systems help companies understand their customers' tastes. By doing so, a customer might be interested in items otherwise he is not aware of. The first example of such a system was Tapestry [1], which was an experimental email filtering system at Xerox. Usually, recommender systems can be broadly categorized into collaborative filtering (CF), content-based filtering (CB), and hybrid approaches.

As the name suggests, CF systems require collaboration from their users [1], [2] to perform their tasks. Clearly, CF systems require user feedback. The feedback can be explicit or implicit. Explicit feedback is the ones that the users supply by casting their ratings. On the other hand, the CF system might utilize some implicit feedback such as browsing, click, or purchase history [3]. CF systems contain two approaches which can be either model or memory-based. Model-based approaches build a model and produce recommendations or predictions based on the model. On the other hand, memory-based approaches re-run the whole procedure to produce an output for the recommendation. The general idea in CF is that users who have similar preferences in the past will probably have similar preferences in the future as well. Therefore, memory-based CF approaches determine a set of neighbors for an active user. An active user is the one who is looking for a prediction or recommendation. The k nearest neighbor algorithm is the most dominant and widely used algorithm to select neighbors in user-based and item-based CF [4]. Although CF systems are widespread, they might suffer from data sparsity and cold start problems [5]. A cold start problem occurs when a new item or user is introduced into the system. Since there are not enough ratings for them, the CF system cannot produce a recommendation. CB utilizes contextual information of the items. Unlike CF, CB does not rely on different user feedbacks. Instead,

it essentially analyzes the content of the items rated by the active user and recommends similar items based on item contents. Such content can be described as a synopsis or actors for a movie-related domain. CB systems can be useful to avoid cold start problems for new items; however, the disadvantage is that it often produces obvious recommendations [6]. Hybrid approaches combine CF and CB to take advantage of them.

CF systems usually operate on $n \times m$ matrix, where n is the number of users while m is the number of items. Users rate items on a binary, interval, ordinal, or continuous scale [6]. In traditional CF systems, users express their preferences based on a single criterion. For example, a movie recommendation system collects the general preferences of the users about the movie. However, a movie evaluation might include several dimensions such as cast, story, direction, or action. Therefore, integrating multiple criteria might be useful to characterize items better. Adomavicius and Kwon [7] propose recommendation techniques for multi-criteria CF (MCCF) systems. Their seminal paper was the first to introduce the concept of MCCF and to propose two methods called similarity-based and aggregation function-based. In the similarity-based approach, the similarity values between users or items are calculated for each criterion using any existing similarity measures and then the overall similarity is calculated by averaging or selecting the smallest one. In the second method, the aggregation function-based approach, a relationship between the overall rating and sub-criteria ratings is extracted by utilizing an aggregation function. Since it is important to find an appropriate aggregation function, domain expertise, statistical techniques, and machine learning methods can be used.

Herlocker et al. [8] study the effectiveness of significance or similarity weighting in single criterion CF. The idea is that an active user might be highly correlated with some users on a small number of co-rated items. For example, two users might be perfectly correlated based on a single item and such correlation is stronger than a correlation of 0.9 over hundred co-rated items. Therefore, the authors [8] argue that the correlation between users can be adjusted by introducing significant weighting. In their original method, they use a threshold-based significance weighting factor. Although single criteria CF schemes have long been utilizing similarity weighting [9]-[12], two recent studies [13], [14] employ similarity weighting in MCCF. Scholars [13] improve the prediction performance of item-based MMCF. Yalcin and Bilge [14] propose a binary MCCF method and they apply binary similarity weighting to calculate the correlation between users. It is clear that MCCF literature lacks a systematic view of the effects of similarity weighting on the prediction performance.

In this study, we present a comprehensive study of the effects of binary vector similarity weighting on both item- and user-based MCCF. Binary vector similarity is used as a weighting factor in addition to the existing similarity measures to improve the prediction performance of MCCF. Considering the structure of the existing similarity measures in similarity-based approaches, similarities are only calculated based on co-rated users or items. Then, nearest neighbors are selected based on these calculated similarity values. On the other hand, higher similarity values may not always indicate a better correlation between users or items because these metrics do not consider how many co-rated items exist between two users or vice versa. In item- or user-based schemes, it is sufficient to have a single common item/user to calculate a similarity score between two users or items. In a user-based perspective, a user vector consists of item ratings while, in an item-based perspective, an item vector consists of users' ratings on the related item. At this point, binary vector representation of the rating vectors can reveal which items/users are rated by encoding rated items by 1 and unrated items by 0. Then, binary vector representations of two users' or items' vectors might be exploited in the similarity calculations since they inform the mutual and non-mutual presences as well as mutual and non-mutual absences of the ratings. Utilizing binary vectors similarities as weighting factors in addition to the existing methods is a solution to overcome the shortcoming of the limited number of co-ratings. This study presents extensive literature on binary vector similarity measures and their effect on similarity weighting in MCCF schemes. Besides, this work can serve as a reference point for future researchers to understand the contribution of binary vector similarity weighting on MCCF prediction accuracy.

The rest of the paper is organized as follows. In Section 2, the related work is given briefly. Section 3 represents detailed information about CF and MCCF. In Section 4, we give a review of binary vector similarity literature and related taxonomy. Then, we discuss how to apply binary vector similarities to

weight MCCF similarities. We experimentally show in Section 5 how binary similarity weighting affects the general prediction performance of user-based and item-based MCCF. Section 6 gives the discussion. The final section represents our conclusions.

2. Related Work

The implementation of multi-criteria recommendation systems dates to the early 2000s. First, Plantié, Montmain, and Dray [15] suggest using a decision support system that combines text mining techniques with multi-criteria analysis techniques to develop movie-based recommendation systems. Adomavicius and Kwon [7] state that using multi-criteria recommendation systems instead of traditional recommendation systems increase the likelihood that the correlations calculated between users. Researchers argue that rating items with more criteria, such as visual effects or scenario in a movie dataset, rather than giving a single rating about the items may also increase the accuracy of the recommendation. They show how traditional memory-based CF algorithms can be adapted to MCCF systems. First, researchers produce recommendations using traditional similarity-based approaches. While calculating the similarities between users, the system calculates the similarity value separately for each criterion as in the traditional similarity-based approach. Then, the average or smallest value of all criteria can be selected as a final similarity. They also show that similarity calculations can be made using multidimensional distance metrics. With this approach, it has been shown that only the traditional similarity calculation process has changed, the rest of the process will remain the same as in traditional CF algorithms to provide predictions. Researchers also mention that all other methods used in single-criteria or traditional recommendation systems can also be adapted to multi-criteria systems beside memory-based collaborative filtering algorithms. Bilge and Kaleli [16] propose a framework that adapts the work proposed by Adomavicius and Kwon [7] to multi-criteria item-based CF systems. The study shows that the performance of item-based multi-criteria systems is better than traditional item-based CF systems.

In memory-based CF algorithms, the number of items commonly rated among users is an important factor. Considering the nature of recommender systems, most items are not rated, and this situation causes similarity calculations to be made on a very small number of co-rated items when calculating a similarity value between two users. The similarity value calculated over a few common items may not reflect the actual correlation. Therefore, Herlocker et al. [8] propose to use a weighting method in traditional recommender systems while calculating the similarity between users. In the proposed method called significance-weighting, a threshold value is defined, and if the number of items voted commonly by two users is less than this value, the similarity is multiplied by the calculated weighting, thus the similarity between the two users is reduced to a certain extent. The goal is to obtain a more reliable similarity between two users. The more commonly rated items between two users, the more trust these two users have. Ma, King, and Lyu [9] modify the work proposed by Herlocker et al. [8] and apply the modified algorithm when calculating similarity both between users and items. Polatidis and Georgiadis [10] propose a new similarity calculation process inspired by the work [8] and divide the similarity process into multiple levels. If the conditions specified at each level are fulfilled, a value determined by the researchers is added to the calculated similarity. The aim of the researchers is to increase the similarity values of these users as more items are rated in common. In another study of the researchers, a dynamic multi-level CF system is also proposed [11]. If the criteria specified in the study are met, the correlation is affected positively by adding different numbers to the similarity calculation for different levels. On the other hand, if no criteria are met, the similarity is multiplied by the specified value and the similarity is devalued. Candillier, Meyer, and Fessant [12] suggest using the Jaccard similarity as a weighting method by multiplying it with the existing similarity methods for CF systems. The study shows that the proposed method gives better results than traditional methods. Shambour and Lu [17] propose to use the Jaccard similarity as a weighting method in item-based MCCF. First, adjusted cosine similarity (ACS) is used to calculate the similarity between items for each criterion. Then, the final similarity is calculated using the weighted average method as in the traditional multi-criteria algorithms. Researchers propose a new similarity calculation that multiplies the final calculated similarity value with Jaccard. Shambour, Hourani, and Fraihat [18] propose to apply the Dice metric as a weighting

method for multi-criteria item-based CF. Dice metric calculates a similarity between two items based on common users, such as Jaccard similarity. Euclidean distance is used while calculating the similarities between items for each criterion. Since the distance metric is selected to calculate similarity values, the similarity with the lowest value is chosen as the final similarity. Researchers propose a new similarity calculation process that multiplies the Dice metric with the final similarity in item-based MCCF systems. In a similar study to the last two, it is proposed for user-based MCCF systems [19]. The researchers integrate the Dice metric into ACS as a weighting method as in previous studies. Sadikoglu and Demirelli Okkalioglu [13] propose to apply Jaccard similarity and significance-weighting as weighting methods to improve the existing similarity calculation process in similarity-based approaches in item-based MCCF. The proposed two weighting methods are multiplied by Pearson Correlation Coefficient (PCC) and ACS for each criterion. The final similarity value is obtained by either the weighted average method or the minimum method. Researchers show that the proposed methods improve the accuracy and coverage of predictions compared to the traditional methods. Yalcin and Bilge [14] propose a binary MCCF method and they apply binary similarity weighting to calculate the correlation between users. Table 1 displays list of MCF work that utilize similarity weighting for comparison purposes.

Table 1 Comparison of the existing work

References	Data Type	Scheme	Similarity Weighting
Shambour and Lu [17]	Numeric	Item-based	Jaccard
Shambour, Hourani, and Fraihat [18]	Numeric	Item-based	Dice
Shambour [19]	Numeric	User-based	Dice
Sadikoglu and Demirelli Okkalioglu [13]	Numeric	Item-based	Jaccard and Significance-weighting
Yalcin and Bilge [14]	Binary	User-based, Item-based	Jaccard, Czekanowski, Simpson, Kulczynski, and Johnson

These studies state that high correlation values may not always ensure the best neighbors in similarity-based approaches. As the number of co-users or co-items increases, the correlation calculated between them is more reliable. Therefore, different weighting methods are proposed to change the neighbor selection process and to improve the prediction performance. Unlike the works presented here, we investigate the binary vector similarity literature and utilize a variety of binary similarity measures as weighting factors in the MCCF similarity calculation process. Thus, our work presents an enhanced view of the effects of binary vector similarities on the prediction accuracy of MCF schemes.

3. Multi-Criteria Collaborative Filtering

Before introducing MCF, brief information about a traditional CF is given. The best-known CF algorithm is the memory-based or neighborhood-based algorithm. The memory-based algorithms are also divided into two classes: user-based and item-based. The underlying approach of user-based is to find like-minded users when an active user (a) asks for a prediction for a target item (q). Equation 1 shows a formula how to calculate a prediction after users are selected as nearest neighbors for a .

$$pred(a, q) = \bar{r}_a + \frac{\sum_{u=1}^k (r_{u,q} - \bar{r}_u) sim(a, u)}{\sum_{u=1}^k sim(a, u)} \quad (1)$$

where k is k -nearest-neighbors of a . \bar{r}_a and \bar{r}_u are mean ratings for user a and user u , respectively. $r_{u,q}$ is the rating of user u on item q and $sim(a, u)$ illustrates the similarity value between user a and user u .

The similarity between users is crucial in CF. There are different methods used in the literature. One of the most popular similarity methods is Pearson Correlation Coefficient (PCC) [20]. PCC calculates similarities in the range [-1, 1], where 1 depicts the highest correlation whereas -1 illustrates the worst correlation between two users. Equation 2 shows how to apply PCC between two users, a and u :

$$sim(a, u) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{u,p} - \bar{r}_u)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{u,p} - \bar{r}_u)^2}} \quad (2)$$

where P demonstrates the set of items rated both an active user, a , and a user, u , $r_{a,p}$ and $r_{u,p}$ represent ratings for item p rated by a and u , respectively. \bar{r}_a and \bar{r}_u are the mean of P items' ratings of a and u .

Whereas user-based algorithms produce a prediction by utilizing user similarities, item-based algorithms generate a prediction based on similarities between items. In this case, an active user (a) asks for a prediction for a target item (q) as in user-based algorithms. The algorithm finds similarities between the target item and other items and k items are selected as the best neighbors. The prediction formula is given in Equation 3.

$$pred(a, q) = \frac{\sum_{i \in ratedItem(a)} sim(i, q) \times r_{a,i}}{\sum_{i \in ratedItem(a)} sim(i, q)} \quad (3)$$

Where $sim(i, q)$ shows the similarity value between item i and item q , $r_{a,i}$ represents the rating value given by user a to item i . Recall that among the selected neighboring items, the items rated by the user a are included in this calculation.

ACS is the common similarity method when calculating similarities between items [21]. The similarity value between two items in ACS is also in the range of $[-1, 1]$ as in PCC. ACS performs the similarity calculation between item i and q using Equation 4.

$$sim(i, q) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,q} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,q} - \bar{R}_u)^2}} \quad (4)$$

In Equation 4, U indicates common users who rated item i and item q . $R_{u,i}$ and $R_{u,q}$ show the rating given by user u to items i and item q , respectively. \bar{R}_u represents the average of ratings of user u .

In addition, PCC and ACS are the best-known similarity methods in CF. Therefore, these methods are widely used in both user-based and item-based algorithms. Equation 2 represents user-based similarity calculation; however, the formula can easily be converted into an item-based one. Likewise, Equation 4 shows item-based ACS, but it can be modified to user-based ACS. Since corresponding item- or user-based formulas are trivial, these formulas are briefly presented here.

A traditional CF system performs similarity calculations over a single criterion. Users send overall ratings about items and a prediction is produced using a single rating. On the other hand, researchers put forward an idea that multi-criteria systems could better reflect the characteristics of the users instead of a single rating-based system. Multi-criteria systems allow users to rate more than one criterion separately. Users reflect their personal preferences better due to the multi-criteria system. A rating function R in a multi-criteria CF is represented as follows:

$$R: users \times items \rightarrow R_0 \times R_1 \times \dots \times R_k \quad (5)$$

where R_0 shows overall ratings that users rate items and R_j represents that users rate items for j th criteria where $j = 1, 2, \dots, k$. k is the number of criteria.

In multi-criteria CF, Adomavicius and Known [7] propose similarity-based and aggregation function-based approaches to provide predictions. In this study, similarity-based approaches are applied. In this approach, similarity values between users/items are calculated using any existing similarity measures for each criterion separately as in traditional CF. Then, it is required to aggregate individual similarities to obtain an overall similarity. Adomavicius and Known [7] introduce two methods that aggregate individual similarities: average and worst-case similarity. In the average similarity approach as seen in Equation 6, the similarities of all individual criteria are averaged to get an overall similarity. In Equation 6, k defines the number of criteria, $sim_c(i, j)$ shows the similarity between item i and item j for criterion c .

$$sim_{avg}(i, j) = \frac{\sum_{c=0}^k sim_c(i, j)}{k + 1} \tag{6}$$

The second aggregation approach is called the worst-case or minimum similarity. The main purpose is to select minimum similarity as a final similarity among all individual criteria. The final similarity is estimated as seen in Equation 7.

$$sim_{min}(i, j) = \min_{c=0,1,\dots,k} sim_c(i, j) \tag{7}$$

Although Equation 6 and Equation 7 show how to calculate the overall similarity in item-based MCCF algorithms, user-based MCCF can be applied similarly [7].

4. Weighting Similarities in Multi-Criteria Collaborative Filtering

4.1 Binary Vector Representations and Similarities

In this section, our motivation is to present binary similarity measures to use them as weighting factors in the neighborhood selection. Since rating vectors are generally very sparse [22] in CF, similarity calculations can be boosted if weighting factors are utilized [23], [24]. PCC, ACS, or other similarity measures are applied over co-rated numeric items. Herlocker et al. [8] argue that the number of co-rated items can contribute to the similarity score as a weighting factor because the similarity between two vectors might have been calculated over few items. For example, the same similarity score, say 1.0 (perfect), may not indicate the same degree of similarity for vectors, which have few versus many co-rated elements. Therefore, weighting similarity scores is used in CF literature [8], [13], [14], [17]-[19], [23], [24] in the neighborhood selection.

A weighting factor can be calculated over binary representations of vectors. Assume that the rating range is a set of discrete numeric values, say $R = \{0, 1, 2, 3, 4, 5\}$, where 0 means unrated. For an item-based MCCF, an item vector for a criterion can be defined as $item_j = \{r_1, r_2, \dots, r_{n-1}, r_n\}$, where j is the item id and $r_i \in R, \forall i \in \{1, 2, \dots, n\}$. Such a vector can be also expressed in a binary form where each r_i is replaced either 1 if $r_i \neq 0$, or 0 otherwise. In binary vector format, 1s represent the presence while 0s represent the absence. For example, a rating vector of $item_1 = \{1, 2, 3, 0, 0, 4, 5, 0, 0, 2, 0\}$ can be transformed into a binary vector, $item_1 = \{1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0\}$. These binary vectors are also called binary basket data in the data mining literature [25].

While calculating the similarities of binary vectors, corresponding absence and presence become crucial. Table 2 displays how absence and presence value will be represented for two vectors, say v_1 and v_2 . a is $|v_1 \cdot v_2|_1$, which is the number of 1s after the dot product. In other words, a represents the number of common elements that both v_1 and v_2 have 1, which is the mutual presence. b is $|v_1 \cdot \overline{v_2}|_1$, which is the number of elements that is present in v_1 but absent in v_2 . c is $|\overline{v_1} \cdot v_2|_1$, which is the number of elements that is absent in v_1 but present in v_2 . $b + c$ can be called the non-mutual presence. d is $|\overline{v_1} \cdot \overline{v_2}|_1$ is the number of elements that are absent in both v_1 and v_2 , which is the mutual absence. Obviously, $|v_1 \cdot v_1|_1 = a + b$, $|v_2 \cdot v_2|_1 = a + c$, and $a + b + c + d$ is the number of features. In our case with item-item similarities, the number of features will be n , which is the number of users.

Table 2 Notation for binary vector similarities

v_1	v_2	
	Presence (1)	Absence (0)
Presence (1)	a	b
Absence (0)	c	d
	$n = a + b + c + d$	

Based on the notation in Table 2, many variations of binary vector similarities can be calculated. Table 3 lists 21 different measures including similarities and distances. The use of d is controversial [26], [27] while some measures do not include it some others include. Because the number of attributes that are absent in both vectors is very dominant, d might be omitted. This is true in CF, as well. These similarity and distance measures can be broadly categorized into two, ones ignoring d and ones considering d

[27], [28]. In addition, binary similarity functions that consider d in the calculation can be further classified as the ones treating both a and d equally or unequally [27].

Table 3 Binary similarity and distance measures

	Measure	Equation	Range	Ref
Similarity measures ignoring d	Jaccard	$\frac{a}{a+b+c}$	[0, 1]	[26], [27]
	Dice	$\frac{2 \times a}{2 \times a + b + c}$	[0, 1]	[26], [27]
	3W-Jaccard	$\frac{3 \times a}{3 \times a + b + c}$	[0, 1]	[26], [27]
	Sokal-Sneath-I	$\frac{a}{a + 2 \times b + 2 \times c}$	[0, 1]	[26], [27]
	Simpson	$\frac{a}{\min(a+b, a+c)}$	[0, 1]	[26], [27]
	Braun	$\frac{a}{\max(a+b, a+c)}$	[0, 1]	[26], [27]
	Johnson	$\frac{a}{a+b} + \frac{a}{a+c}$	[0, 2]	[26], [27]
	Kulczynski-I	$\frac{a}{b+c}$	[0, ∞)	[26], [27]
	Sorgenfrei	$\frac{(a \times a)}{(a+b) \times (a+c)}$	[0, 1]	[26], [27]
	Ochiai	$\frac{a}{\sqrt{(a+b)(a+c)}}$	[0, 1]	[26], [27]
Similarity measures considering d	Russel-Rao	$\frac{a}{a+b+c+d}$	[0, 1]	[26], [27]
	Sokal-Michener	$\frac{a+d}{a+b+c+d}$	[0, 1]	[26], [27]
	Sokal-Sneath-2	$\frac{2 \times (a+d)}{2 \times a + b + c + 2 \times d}$	[0, 1]	[26], [27]
	Roger-Tanimoto	$\frac{a+d}{a+2 \times (b+c)+d}$	[0, 1]	[26], [27]
	Gower-Legendre	$\frac{a+d}{a+0.5 \times (b+c)+d}$	[0, 1]	[26], [27]
	Faith	$\frac{a+0.5 \times d}{a+b+c+d}$	[0, 1]	[26], [27]
Distances ignoring d	Squared-Euclid	$\sqrt{b+c}^2$	[0, ∞)	[26], [27]
	Manhattan	$b+c$	[0, ∞)	[26], [27]
Distances considering d	Mean- Manhattan	$\frac{b+c}{a+b+c+d}$	[0, 1]	[26], [27]
	Size-Difference	$\frac{(b+c)^2}{(a+b+c+d)^2}$	[0, 1]	[26], [27]
	Shape-Difference	$\frac{n \times (b+c) - (b-c)^2}{(a+b+c+d)^2}$	[0, 1]	[26], [27]

Similarity measures that ignore d are generally useful for asymmetric binary vectors, where the presence is more important than absence, such as market basket data in the data mining literature. There might be many items that can go into a market basket; however, any two customers can have a very limited number of common items in their baskets. Therefore, calculating a similarity using d either in the numerator or denominator part of the equation devalues the mutual presence, which is a . The examples of such similarity measures that do not utilize d in Table 3 start from Jaccard until Ochiai. Jaccard is the ratio of dot products of two vectors in consideration over the union of presences (mutual presence plus non-mutual presence). The Jaccard measure does not prioritize mutual presence over non-mutual presences. However, Dice and 3w-Jaccard weigh the mutual-presence (a) by two and three, respectively. On the contrary, Sokal-Sneath-I imposes a penalty for non-mutual presence by a factor of two. Simpson and Braun measures are the ratio of mutual presence to the minimum and maximum non-mutual presence, respectively. Johnson, on the other hand, utilizes both. Kulczynski-I measure is the ratio of

mutual presence over non-mutual presence. In terms of set theory, it is the cardinality of the intersection over the sum of the cardinalities of relative differences. However, such a measure as Kulczynski range between 0 and ∞ . To avoid it, we use the updated Kulczynski formula [28]. Sorgenfrei and Ochiai measures are similar, where Ochiai is the extension of Cosine similarity when vectors are binary. Sorgenfrei, on the other hand, is the square of Ochiai measure.

Another group of measures utilizes d in a very similar nature. Russel-Rao is similar to Jaccard; however, it measures mutual presences over all elements. Sokal-Michener does not weigh any of a, b, c, d while Sokal-Sneath-2, Roger-Tanimoto, Faith, and Gower-Legendre utilize weighting on some of them. Distance measures need to be transformed into similarities. Squared-Euclid and Manhattan value can range between 0 and ∞ ; therefore, we converted them to $1/(1 + distance)$, where $distance$ is either Squared-Euclid or Manhattan score. Likewise, the rest of the distance metrics are converted into $(1 - distance)$ because their range is between 0 and 1.

4.2 Weighting Similarities in Multi-Criteria Collaborative Filtering by Binary Vector Similarity Measures

As introduced, PCC and ACS are the most prevalent techniques to calculate similarities in item- and user-based MCCF. Considering the item-based MCCF for brevity, both methods compute the item-item similarities based on users who rate the related items. Assume that we need to calculate $item_1$'s similarity to $item_2$ and $item_3$ and the number of common users who rate $item_1$ and $item_2$ is 1 while the number of users who both rate $item_1$ and $item_3$ is 100. The similarity score between $item_1$ and $item_2$, which is based on a single user, might be greater than the similarity between $item_1$ and $item_3$. Since PCC and ACS do not take the number of users who rate both items into account, the similarity scores may become controversial. According to Herlocker et al. [8], similarities become more credible if items are co-rated by many users. Therefore, similarities can be weighted by the number of users co-rating both items. For this purpose, we will utilize binary vector similarities to weigh item-item similarities as given in Equation 8, which gives the average aggregation, and Equation 9, which gives the minimum aggregation. Here, the item-item similarity for each criterion, sim_c , is multiplied by the binary similarity measure, which is $weightingFactor_c$, which can be adopted from Table 3.

$$sim'_{avg}(i, j) = \frac{\sum_{c=0}^k sim_c(i, j) \times weightingFactor_c(i, j)}{k + 1} \quad (8)$$

$$sim'_{min}(i, j) = \min_{c=0,1,\dots,k} sim_c(i, j) \times weightingFactor_c(i, j) \quad (9)$$

We briefly discuss item-based MCCF to weight similarities, the very similar idea can be applied to the user-based MCCF. In user-based settings, user-user similarities must be weighted by the binary vector similarities that are constructed for the co-rated items between users. Each calculated similarity between users for each criterion is multiplied by the corresponding binary vector similarity measure computed by the related user vectors. Again, these measures can be adopted from Table 3.

5. Experimental Evaluation

5.1 Data set


The most widely used dataset in MCCF, Yahoo!Movies, is used to test our proposed approaches. The dataset includes four sub-criteria and an overall rating for the movie domain. Users are asked to rate criteria such as direction, acting, story, and visuals. In addition, the average of four sub-criteria is calculated as the overall rating. Due to the extreme sparsity of the dataset, two subsets of Yahoo!Movies are created [29]. First, a subset of users and items with at least 10 ratings is created and called YM10. Likewise, YM20 dataset is also obtained with a subset of users and items with at least 20 ratings. Table 4 displays YM10 and YM20 data sets and their corresponding number of users, items and the ratings they have.

Table 4 Datasets

	YM10	YM20
# users	1293	202
# items	1164	247
# ratings	34846	8157

Yahoo!Movies includes 13 different ratings, and the rating range is from A+ to F. A+ indicates the highest rating, while F shows the lowest rating. Since CF systems usually use 5 star-scale, the ratings in Yahoo!Movies are converted into 5 star-scale. (A+, A, A-) is converted to 5, (B+, B, B-) is exchanged to 4, others are transformed to 3, 2, and 1 is assigned to the letter F [29]. A small example of the multi-criteria user-item matrix is given in Figure 1.

	<i>item</i> ₁	<i>item</i> ₂	<i>item</i> ₃	<i>item</i> ₄
<i>user</i> ₁	-	$B_{C^+,A^+,C^+,A}^+$	-	$B_{B^+,C,B,A}$
<i>user</i> ₂	$D_{C,F,D,D}$	-	-	$F_{F,F,F,D}^-$
<i>user</i> ₃	$A_{A^+,A,A,A}$	-	$C_{B,C,C,D}$	-
<i>user</i> ₄	-	$A_{A^-,B^+,A,A}^-$	$B_{C,A,A,C}$	-



	<i>item</i> ₁	<i>item</i> ₂	<i>item</i> ₃	<i>item</i> ₄
<i>user</i> ₁	-	4 _{3,5,3,5}	-	4 _{4,3,4,5}
<i>user</i> ₂	2 _{3,1,2,2}	-	-	1 _{1,1,1,2}
<i>user</i> ₃	5 _{5,5,5,5}	-	3 _{4,3,3,2}	-
<i>user</i> ₄	-	5 _{5,5,4,5}	4 _{3,5,5,3}	-

Figure 1 An Example of the multi-criteria user-item matrix

5.2 Methodology

Experiments are conducted with YM20 and YM10 data sets. 5-fold cross-validation is utilized where each experiment is split into four training and one test set. Each time a prediction is produced for an active user (test user in the test set). For an active user, all rated items are listed and one of them is removed and the original value is stored for future tests, a prediction is produced for the removed target item, q . After the prediction is calculated, it is compared with the original value to calculate the accuracy metric, which is given in 6.3. Then, the removed original value is copied back for the next test. This strategy is repeated and known as leave-one-out. The number of neighbors is set to 40 [13], [16].

5.3 Evaluation Criteria

Mean absolute error, MAE, is used as the evaluation criteria. MAE measures the prediction error in terms of absolute difference over all queries. Its formulation is given in Equation 10, where p_i is the prediction, o_i is the original rating, and R is the number of predictions. It basically tells how much the predictions deviate from the original value.

$$MAE = \frac{1}{R} \sum_{i=1}^R |p_i - o_i| \quad (10)$$

5.4 Experiments

5.4.1 Effects of binary vector similarity on item-based MMCF

In the first experiment, we analyze how different binary similarity measures affect MAE scores when item-based MMCF is employed. Recall that PCC and ACS item-item similarities are aggregated with average and minimum. Table 5 displays the related results for both YM20 and YM10 data sets. Notice that bold table cells show that corresponding binary vector similarity weighting outperforms the original MCCF scheme.

The traditional MCCF method performs around 0.661, 0.681, 0.590, and 0.602 when PCC and ACS are aggregated by average and minimum functions, respectively, for YM20 data set. When the binary vector similarity measures are introduced in the item-item similarity calculation, most of them outperform the traditional MCCF method as seen in Table 5. The most remarkable measures are Sorgenfrei, Kulczynski-I, for YM20 data set. Sorgenfrei, Kulczynski-I are always among the top three measures in terms of MAE improvement compared to the traditional MCCF. Soregnfrei achieves 8%, 6.7%, 6.3%, and 5.8%, and Kulczynski-I achieves 7.5%, 6.5%, 6.6%, and 4.7% improvements for PCC-AVG, PCC-

MIN, ACS-AVG, and ACS-MIN, respectively. Sokal-Sneath-I accompanies these two measures in top-three in terms of MAE improvement when item-item similarities are calculated by PCC-MIN, ACS-AVG, and ACS-MIN. On the other hand, the worst performing measure is the Shape-Difference which always falls behind the traditional MCCF. However, it is prominent in this experiment to note that none of the binary vector similarities perform worse than the traditional MCCF except Shape-Difference for YM20 data set. Each of them, at least, achieves the same results with the traditional MCCF. Another interesting fact about binary similarity measures is that binary similarity measures ignoring d always perform better than both traditional MCCF and the rest of the binary similarity measures except Russel-Rao for YM20 data set.

Table 5 MAE Results When Binary Vector Similarities Are Applied in Item-based MCCF

	YM20				YM10			
	PCC AVG	PCC MIN	ACS AVG	ACS MIN	PCC AVG	PCC MIN	ACS AVG	ACS MIN
MCCF Traditional	0.661	0.681	0.590	0.602	0.760	0.772	0.777	0.777
Jaccard	0.617	0.652	0.555	0.577	0.677	0.711	0.605	0.623
Dice	0.622	0.654	0.558	0.579	0.679	0.714	0.609	0.624
3W-Jaccard	0.627	0.658	0.560	0.581	0.682	0.717	0.610	0.625
Sokal-Sneath-I	0.615	0.646	0.553	0.576	0.676	0.709	0.604	0.622
Simpson	0.624	0.660	0.562	0.582	0.692	0.727	0.632	0.639
Braun	0.623	0.655	0.559	0.580	0.685	0.715	0.613	0.628
Johnson	0.622	0.657	0.559	0.579	0.684	0.720	0.615	0.627
Kulczynski-I	0.611	0.637	0.551	0.574	0.674	0.707	0.603	0.620
Sorgenfrei	0.608	0.635	0.553	0.567	0.665	0.700	0.603	0.617
Ochiai	0.623	0.657	0.558	0.580	0.681	0.715	0.609	0.622
Russel-Rao	0.613	0.648	0.560	0.579	0.680	0.714	0.620	0.625
Sokal-Michener	0.658	0.681	0.584	0.600	0.757	0.769	0.755	0.752
Sokal-Sneath-2	0.661	0.681	0.586	0.601	0.760	0.770	0.758	0.750
Roger-Tanimoto	0.656	0.681	0.584	0.598	0.755	0.770	0.754	0.750
Gower-Legendre	0.661	0.681	0.586	0.601	0.760	0.770	0.758	0.750
Faith	0.652	0.678	0.581	0.598	0.755	0.768	0.747	0.748
Squared-Euclid	0.647	0.676	0.579	0.594	0.736	0.756	0.715	0.727
Manhattan	0.647	0.676	0.579	0.594	0.736	0.756	0.715	0.727
Mean- Manhattan	0.658	0.681	0.584	0.600	0.757	0.769	0.755	0.752
Size-Difference	0.662	0.681	0.588	0.602	0.760	0.770	0.759	0.750
Shape-Difference	0.864	0.942	1.030	1.060	0.888	0.905	1.032	1.065

Table 5 also displays the results for YM10 data set. The traditional method for MCCF records 0.760, 0.772, 0.777, and 0.777 for PCC and ACS when aggregated with average and minimum function, respectively. In all tests, the top three improvement scores are achieved by measures that ignore d , Sorgenfrei, Kulczynski-I, and Sokal-Sneath-I. Sorgenfrei performs 12.5%, 9.3%, 22.4%, and 20.6% improvement for PCC-AVG, PCC-MIN, ACS-AVG, and ACS-MIN, respectively. Moreover, Sorgenfrei is always the best-performing measure. Kulczynski-I improves the traditional MCCF by 11.3%, 8.4%, 22.4%, and 20.2%, while Sokal-Sneath-I achieves an improvement of 11.1%, 8.2%, 22.3%, and 19.9% for PCC-AVG, PCC-MIN, ACS-AVG, and ACS-MIN, respectively. Beside these top three measures, the other binary vector similarity measures except Shape-Difference achieve an improvement over traditional MCCF.

Since Table 5 includes experimental results of 22 different binary similarity measures for four different MCCF methods and two different data sets, we also illustrate the best performing binary similarity measures in Figure 2 for readers' convenience.

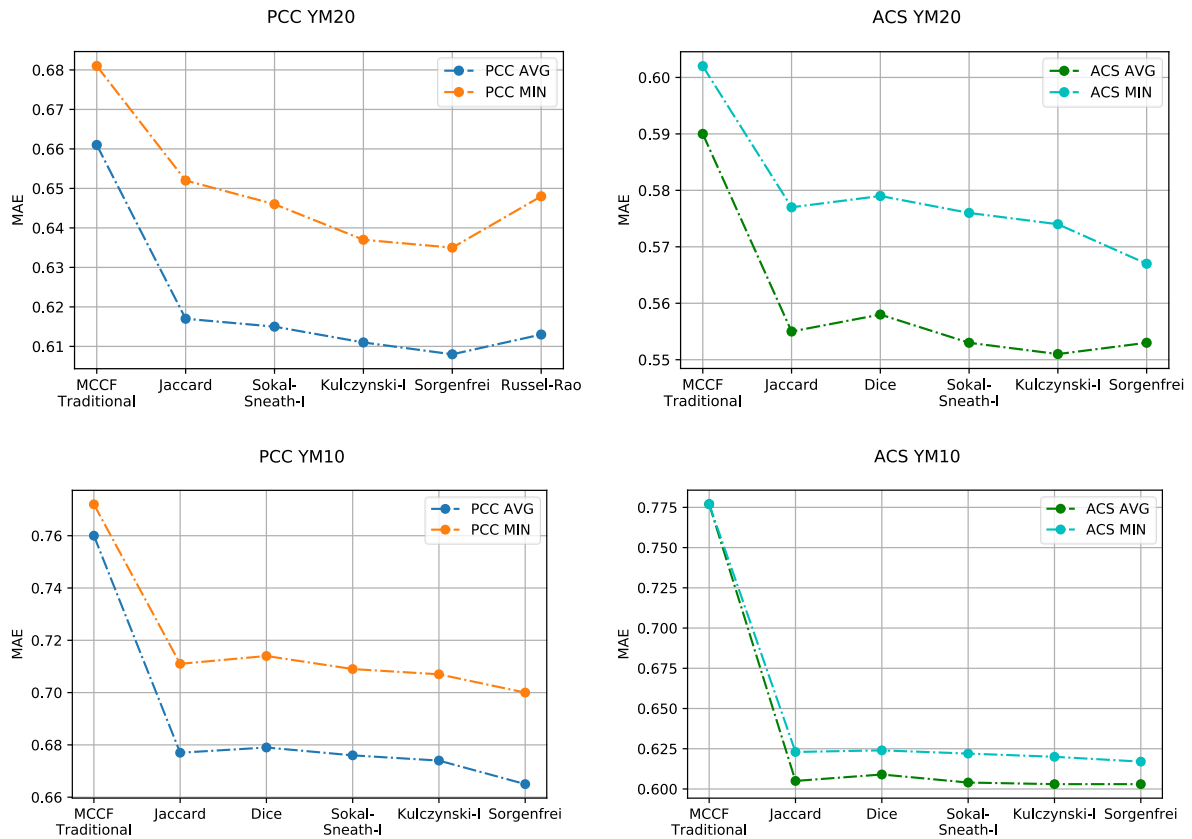


Figure 2 The best performing binary similarity measures against Traditional MCCF in item-based MCF

We analyzed how binary similarity measures affect the MAE results when item-based MCF is utilized. The experiment shows that weighting item-item similarities based on their binary vector representations can improve prediction accuracy in MCF. Based on the experimental findings, binary vector similarities that ignore d are relatively more successful than the measures that consider d . This phenomenon might occur due to the fact that mutual presences (occurrences of 1s) are more important than mutual absences (occurrences of 0s) in item-item similarity calculations. Because the data sets in CF are usually sparse; therefore, mutual absences do not reveal much. Besides, although dissimilarity measures usually contribute to the prediction accuracy in terms of MAE, Shape-Difference always falls behind the traditional MCF.

5.4.2 Effects of binary vector similarity on user-based MCF

This experiment analyzes the effect of binary vector similarities on the user-based MCF. Unlike the previous example, the binary similarity of two user vectors is multiplied by the user-user similarity. Table 6 displays the corresponding results. Recall that bold cells are better than the traditional MCF score.

When user-based MCF is applied on YM20 data set with binary vector similarity weighting, MAE scores are 0.552, 0.592, 0.497, and 0.623, for PCC-AVG, PCC-MIN, ACS-AVG, and ACS-MIN, respectively. Binary vector similarities ignoring d comparably achieve better results than the traditional MCF. In terms of measures considering d , they generally fall behind the traditional MCF with PCC-AVG and ACS-MIN; on the contrary, they are slightly better when item-item similarity calculation is based on PCC-MIN and ACS-AVG. The last group of binary vector similarities utilizes distance

functions, they perform similar to the group of measures that considers d . However, Shape-Difference, once more, is dramatically lower than the traditional MCCF.

When user-based MCCF is applied to YM10 data set which is sparser than YM20, weighting user-based similarities by binary vectors similarities, in general, does not introduce an improvement for PCC-AVG, PCC-MIN, and ACS-AVG. The only setting that an increase in MAE is obvious occurs when ACS-MIN is applied to weight user-user similarities. In this setting, the measures that consider d and dissimilarity functions (except Shape-Difference) record some improvement.

Table 6 MAE Results When Binary Vector Similarities Are Applied in User-based MCCF

	YM20				YM10			
	PCC AVG	PCC MIN	ACS AVG	ACS MIN	PCC AVG	PCC MIN	ACS AVG	ACS MIN
MCCF Traditional	0.552	0.592	0.497	0.623	0.584	0.633	0.407	0.513
Jaccard	0.548	0.585	0.495	0.596	0.606	0.645	0.528	0.538
Dice	0.548	0.588	0.496	0.602	0.606	0.646	0.526	0.536
3W-Jaccard	0.548	0.587	0.496	0.602	0.605	0.646	0.524	0.535
Sokal-Sneath-I	0.548	0.584	0.494	0.597	0.605	0.644	0.528	0.539
Simpson	0.551	0.586	0.501	0.593	0.609	0.640	0.522	0.532
Braun	0.549	0.584	0.495	0.605	0.612	0.649	0.532	0.537
Johnson	0.549	0.591	0.498	0.593	0.599	0.640	0.519	0.529
Kulczynski-I	0.546	0.579	0.493	0.598	0.605	0.643	0.529	0.538
Sorgenfrei	0.550	0.581	0.500	0.556	0.615	0.648	0.542	0.557
Ochiai	0.547	0.587	0.497	0.601	0.603	0.647	0.520	0.535
Russel-Rao	0.558	0.589	0.500	0.578	0.621	0.649	0.539	0.549
Sokal-Michener	0.552	0.593	0.495	0.634	0.584	0.636	0.427	0.488
Sokal-Sneath-2	0.552	0.590	0.496	0.629	0.583	0.634	0.425	0.486
Roger-Tanimoto	0.554	0.593	0.496	0.635	0.587	0.632	0.426	0.489
Gower-Legendre	0.552	0.590	0.496	0.629	0.583	0.634	0.425	0.486
Faith	0.551	0.590	0.493	0.626	0.584	0.634	0.418	0.485
Squared-Euclid	0.554	0.589	0.496	0.636	0.600	0.635	0.461	0.495
Manhattan	0.554	0.589	0.496	0.636	0.600	0.635	0.461	0.495
Mean- Manhattan	0.552	0.593	0.495	0.634	0.584	0.636	0.427	0.488
Size-Difference	0.551	0.589	0.496	0.629	0.583	0.633	0.431	0.485
Shape-Difference	0.881	0.694	0.768	0.891	0.969	0.938	0.861	0.917

Unlike the previous experiment where item-based MCCF is tested, the effect of introducing binary vector similarity to weight user-user similarities is relatively limited. These two experiments show that if the MCCF algorithm utilizes an item-based approach, binary vector similarities, especially the ones ignoring d can be integrated into the method for improved prediction accuracy. However, one should be informed about the fact that weighting might not help improve the prediction accuracy when user-based MCCF is applied if the data set is sparse like YM10.

Since Table 6 includes many experimental results and they are very close to each other five binary similarity measures are illustrated in Figure 3.

6. Discussion

MCCF approaches utilizing neighborhood methods need to calculate either user-user or item-item similarities. If the scheme is user-based, then the similarity calculation is performed on the co-rated

items of associated users. Otherwise, in item-based schemes, co-rating users of associated items are utilized in the similarity calculation. Similarities, on the other hand, are calculated based on co-ratings, regardless of how many there are. The number of co-ratings, whether a few or many, does not prevent the similarity calculation. Similarities calculated with very few co-ratings may be misleading as they are not built on an overwhelming mutual presence of ratings. Therefore, the lack of enough co-ratings may affect the neighbor selection and prediction performance negatively. Similarity weighting is utilized to alleviate such concerns. The general idea is to associate the similarity calculation with mutual presences and absences or non-mutual presences and absences of corresponding vectors. Such relations between different user or item vectors can be constructed by leveraging binary vector representations and similarities between binary vector representations can be employed as a weighting factor in neighbor selection while calculating similarities.

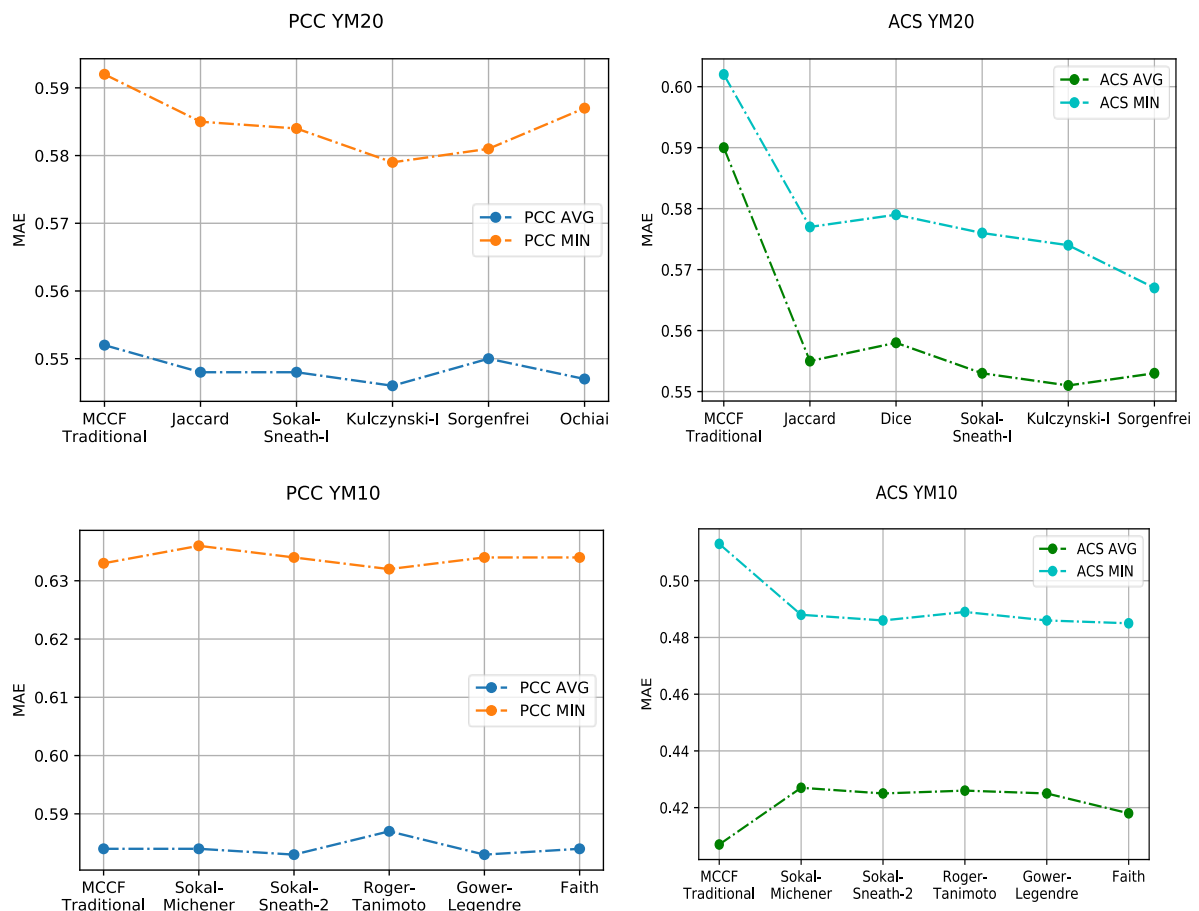


Figure 3 Successful binary similarity measures against Traditional MCCF in user-based MCCF

We, in this study, discuss item- and user-based MCCF schemes and the literature of binary vector similarity in detail. Then, we design and conduct extensive experiments to evaluate the effects of similarity weighting on the prediction accuracy of both item- and user-based multi-criteria collaborative filtering when item-item and user-user correlations are measured by PCC and ACS similarity, which are widely in use. The binary vector similarities measures are grouped under similarity- and distance-based methods. Those measures are further categorized as the ones considering mutual absences or not. Because collaborative filtering data is usually very sparse, the assumption is that mutual presence becomes more important than mutual absences. Experimental findings confirm the assumption: similarities weighted by binary vector similarity measures that ignore mutual absences achieve comparably better results (in terms of mean absolute error) than measures considering mutual absences as well as the traditional item-based MCCF where no weighting is applied. Among the binary vector similarity measures, Sorgenfrei, Kulczynski-I, and Sokal-Sneath-I are worth mentioning because they are generally the top three measures contributing the highest improvement percentages. The highest

improvement is achieved by Sorgenfrei and Kulczynski-I weightings by 22.4% for YM10 data set. Unlike item-based MCCF, in the user-based setting, the effect of binary vector similarity weighting on the prediction performance remains relatively limited. Therefore, one should consider the fact that similarity weighting may not improve the prediction accuracy in MCCF when user-based schemes are employed. Beyond the improvements achieved, one should be aware of the fact that employing Shape-Difference distance metric as a weighting factor never contributes to the prediction accuracy.

7. Conclusions

In this study, we perform a detailed analysis of the effects of binary vector similarities on the prediction performance of the multi-criteria collaborative filtering. Twenty-one different binary vector similarity measures have been discussed and they are used as weighting factor to scale item- and user-based correlations. Experimental findings suggest that measures not considering mutual absences contribute to the prediction accuracy. As a future work, we plan to scrutinize different aspects of binary multi-criteria collaborative filtering schemes in which the number of studies is limited.

Acknowledgments

The author has no conflict of interest to declare.

References

- [1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992, doi: 10.1145/138859.138867.
- [2] Z. Zhang, T. Peng, and K. Shen, "Overview of collaborative filtering recommendation algorithms," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 440, 2020.
- [3] S. Chen and Y. Peng, "Matrix factorization for recommendation with explicit and implicit feedback," *Knowledge-Based Systems*, vol. 158, pp. 109–117, Oct. 2018, doi: 10.1016/j.knosys.2018.05.040.
- [4] B. Demirelli Okkalioglu, "Improving Prediction Performance of Dynamic Neighbor Selection in User-Based Collaborative Filtering," *Sakarya University Journal of Computer and Information Sciences*, vol. 3, no. 2, pp. 73–87, Aug. 2020, doi: 10.35377/saucis.03.02.714969.
- [5] P. K. Singh, P. K. D. Pramanik, A. K. Dey, and P. Choudhury, "Recommender systems: an overview, research trends, and future directions," *Int. J. Bus. Syst. Res.*, vol. 15, no. 1, 2021.
- [6] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st ed. 2016. Cham: Springer International Publishing: Imprint: Springer, 2016.
- [7] G. Adomavicius and Y. Kwon, "New Recommendation Techniques for Multicriteria Rating Systems," *IEEE Intell. Syst.*, vol. 22, no. 3, pp. 48–55, May 2007, doi: 10.1109/MIS.2007.58.
- [8] J. Herlocker, J. A. Konstan, and J. Riedl, "An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms," *Information Retrieval*, vol. 5, no. 4, pp. 287–310, Oct. 2002, doi: 10.1023/A:1020443909834.
- [9] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, Amsterdam, The Netherlands, 2007, doi: 10.1145/1277741.1277751.
- [10] N. Polatidis and C. K. Georgiadis, "A multi-level collaborative filtering method that improves recommendations," *Expert Systems with Applications*, vol. 48, pp. 100–110, Apr. 2016, doi: 10.1016/j.eswa.2015.11.023.
- [11] N. Polatidis and C. K. Georgiadis, "A dynamic multi-level collaborative filtering method for improved recommendations," *Computer Standards & Interfaces*, vol. 51, pp. 14–21, Mar. 2017, doi: 10.1016/j.csi.2016.10.014.
- [12] L. Candillier, F. Meyer, and F. Fessant, "Designing Specific Weighted Similarity Measures to

- Improve Collaborative Filtering Systems,” in *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, 2008, pp. 242–255.
- [13] E. Sadikoğlu and B. D. Okkaloğlu, “Increasing Prediction Performance Using Weighting Methods in Multi-Criteria Item-Based Collaborative Filtering,” *European Journal of Science and Technology*, pp. 110–121, Aug. 2020, doi: 10.31590/ejosat.779171.
- [14] E. Yalçın and A. Bilge, “Binary multicriteria collaborative filtering,” *Turk J Elec Eng & Comp Sci*, vol. 28, no. 6, pp. 3419–3437, Nov. 2020.
- [15] M. Plantié, J. Montmain, and G. Dray, “Movies Recommenders Systems: Automation of the Information and Evaluation Phases in a Multi-criteria Decision-Making Process,” in *Database and Expert Systems Applications*, 2005, pp. 633–644.
- [16] A. Bilge and C. Kaleli, “A multi-criteria item-based collaborative filtering framework,” in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Chon Buri, May 2014, pp. 18–22. doi: 10.1109/JCSSE.2014.6841835.
- [17] Q. Shambour and J. Lu, “A Hybrid Multi-criteria Semantic-Enhanced Collaborative Filtering Approach for Personalized Recommendations,” *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2011, pp. 71–78, doi: 10.1109/WI-IAT.2011.109.
- [18] Q. Shambour, M. Hourani, and S. Fraihat, “An Item-based Multi-Criteria Collaborative Filtering Algorithm for Personalized Recommender Systems,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 8, 2016, doi: 10.14569/IJACSA.2016.070837.
- [19] Q. Shambour, “A user-based multi-criteria recommendation approach for personalized recommendations,” *International Journal of Computer Science and Information Security*, vol. 14, no. 12, 2016.
- [20] A. Gazdar and L. Hidri, “A new similarity measure for collaborative filtering based recommender systems,” *Knowl. Based Syst.*, vol. 188, no. 105058, 2020.
- [21] Y. Wang, P. Wang, Z. Liu, and L. Y. Zhang, “A new item similarity based on α -divergence for collaborative filtering in sparse data,” *Expert Syst. Appl.*, vol. 166, no. 114074, 2021.
- [22] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, “Collaborative filtering and deep learning based recommendation system for cold start items,” *Expert Systems with Applications*, vol. 69, pp. 29–39, Mar. 2017, doi: 10.1016/j.eswa.2016.09.040.
- [23] H.-J. Kwon, T.-H. Lee, J.-H. Kim, and K.-S. Hong, “Improving Prediction Accuracy Using Entropy Weighting in Collaborative Filtering,” in *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, Jul. 2009, pp. 40–45. doi: 10.1109/UIC-ATC.2009.50.
- [24] M. Ghazanfar and A. Prugel-Bennett, “Novel Significance Weighting Schemes for Collaborative Filtering: Generating Improved Recommendations in Sparse Environments,” presented at the *DMIN’10, the 2010 International Conference on Data Mining!*, Jul. 12, 2010. Accessed: Jun. 02, 2021. [Online]. Available: <https://eprints.soton.ac.uk/270846/>
- [25] J. Han and M. Kamber, *Data mining*. Burlington, MA: Elsevier, 2012.
- [26] S. Choi and S. Cha, “A survey of Binary similarity and distance measures,” *Journal of Systemics, Cybernetics and Informatics*, pp. 43–48, 2010.
- [27] J. Podani, “Distance, similarity, correlation...”, in *Introduction to the exploration of multivariate biological data*. Leiden: Backhuys Publishers, 2000.
- [28] S.-H. Cha, C. Tappert, and S. Yoon, “Enhancing binary feature vector similarity measures,” *J. Pattern Recognit. Res.*, vol. 1, no. 1, pp. 63–77, 2006. doi: 10.13176/11.20.
- [29] D. Jannach, Z. Karakaya, and F. Gedikli, “Accuracy improvements for multi-criteria recommender systems,” in *Proceedings of the 13th ACM Conference on Electronic Commerce - EC ’12*, Valencia, Spain, 2012, doi: 10.1145/2229012.2229065.