SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 3, NO. 2, AUGUST 2020
DOI: 10.35377/saucis.03.02.714969

Improving Prediction Performance of Dynamic Neighbor Selection in User-Based Collaborative Filtering

Burcu Demirelli Okkalioglu[1]
[1]Corresponding Author; Yalova University, Faculty of Engineering, Computer Engineering Department; bdokkalioglu@yalova.edu.tr; https://orcid.org/0000-0003-2867-4667; +902268155346

Received 05 April 2020; Revised 09 June 2020; Accepted 17 July 2020; Published online 28 August 2020

Abstract

Recommender systems have become more and more popular in online environments in recent years. Although different approaches are introduced to build a powerful recommender system, collaborative filtering is one of the most used approaches in the recommender systems. Yet, researchers still introduce new methods to improve prediction performances in collaborative filtering. *k* nearest neighbor algorithm is one of the most dominant and prevalent one in collaborative filtering. The underlying approach behind it is to select a predefined *k* neighbors for an active user among all users. In the traditional algorithm, the value of *k* is constant and is determined before the prediction process. Recently, scholars proposed to use dynamic *k* neighbor selection for each user. Inspired from this work, we propose to improve prediction performance, accuracy and coverage, of collaborative filtering systems under *k* nearest neighbor approach. We first propose that users who rate the target item should become nominees for dynamic *k* neighbor selection instead of all possible users whose similarities can be calculated. The similarity calculation is the most crucial point of the *k* nearest neighbor algorithm. Furthermore, we also propose to use the significance-weighting approach in addition to the traditional Pearson correlation coefficient when identifying the best dynamic *k* neighbors for each user. The experimental results on the two well-known datasets show that the prediction accuracy and coverage improve in the dynamic *k* neighbor selection method by selecting neighbors among users who rated the target item and introducing the significance-weighting factor into the neighbor selection phase to find more eligible neighbors.

Keywords: dynamic *k*, significance-weighting, collaborative filtering, accuracy, coverage

Kullanıcı Tabanlı İşbirlikçi Filtrelemede Dinamik Komşu Seçiminin Tahmin Performansını Artırma

Öz

Öneri sistemleri son yıllarda çevrimiçi ortamlarda giderek daha popüler hale geldi. Güçlü bir öneri sistemi oluşturmak için farklı yaklaşımlar kullanılmasına rağmen, işbirlikçi filtreleme öneri sistemlerinde en çok kullanılan yaklaşımlardan biridir. Buna rağmen araştırmacılar, işbirlikçi filtrelemede tahmin performanslarını iyileştirmek için hala yeni yöntemler sunuyorlar. *k* en yakın komşu algoritması, işbirlikçi filtrelemede en dominant ve yaygın algoritmalardan biridir. Bu algoritmanın arkasındaki temel yaklaşım, aktif kullanıcı için tüm kullanıcılar arasından önceden tanımlanmış *k* tane komşu seçmektir. Geleneksel algoritmada *k* değeri sabittir ve tahmin işleminden önce belirlenir. Son zamanlarda, araştırmacılar her kullanıcı için dinamik *k* komşu seçimini kullanmayı önermişlerdir. Bu çalışmadan esinlenerek, en yakın komşu yaklaşımı altında işbirlikçi filtreleme sistemlerinin tahmin performansını, doğruluğunu ve kapsamını, iyileştirmeyi öneriyoruz. İlk olarak, benzerlikleri hesaplanabilen olası tüm kullanıcılar yerine hedef ürünü oylayan kullanıcıların dinamik *k* komşu seçimi için aday olmaları önerilir. Benzerlik hesaplaması, en yakın komşuluk algoritmasının en önemli noktasıdır. Ayrıca, her kullanıcı için en iyi dinamik *k* komşularını belirlerken geleneksel Pearson korelasyon katsayısına ek olarak önem-ağırlıklandırma yaklaşımını kullanmayı öneriyoruz. Bilinen iki veri kümesindeki deneysel sonuçlar, daha kalifiye komşular seçmek için komşuları hedef ürünü oylayan kullanıcılar arasından seçmenin ve komşu seçme aşamasına önem-ağırlıklandırma yöntemi eklemenin dinamik *k* komşu seçimi metodunun tahmin performansını iyileştirdiğini göstermiştir.

Anahtar Kelimeler: dinamik *k*, önem-ağırlıklandırma, işbirlikçi filtreleme, doğruluk, kapsam

1. Introduction

As the World Wide Web rapidly spread, people have begun to spend lots of time using it. People can buy anything they want in seconds or they can watch a movie without going to a cinema. The main problem is to find the right thing people need to buy or want to watch over the web. Recommender system (RS) techniques offer users to solve the information overload problem by forming a system that takes users' preferences and makes a prediction users will likely prefer. RSs have become popular in recent years [1]. They have been started to be used in many different sectors from product recommendations on e-commerce web sites to the music recommendation. The majority of e-commerce sites such as Netflix, Amazon, Spotify, and eBay use RSs to provide customers with personalized service of their favorite products [2]-[4]. The purpose of these companies is to speed up their customers' decision-making process with the help of RS.

There are mainly three types of RSs which are content-based RS, collaborative filtering (CF) and hybrid RS [5]. In content-based RSs [6], [7], keywords are used to describe items. Algorithms used in content-based RSs try to predict similar items for users that users like in the past. If the user did not like or buy an item containing a specific keyword, there might be no chance to recommend this type of item to the user, so the variety of suggestions is reduced, and this is undesirable. CF is the most known and popular algorithms in RS [8]-[10]. It is based on the assumption that users with similar preferences in the past will make the same choices in the future. Basically, CF systems collect users' ratings about items by explicitly or indirectly and compute the similarity between users or items and finally produce a recommendation. One of the most important advantages is that it does not need content of items in opposition to content-based RS. CF can correctly recommend complex items without requiring to know the contents of them. The hybrid RS has emerged to eliminate the disadvantages of existing RSs and to create a more efficient recommender system [11], [12].

CF algorithms can be divided into two main groups which are neighborhood-based and model-based. Although neighborhood-based approaches use a whole user-item matrix to produce a recommendation [13]-[15], model-based approaches use the ratings in the user-item matrix to learn a model [16]-[19]. Neighborhood-based approaches are also grouped in user-based and item-based [20]. In user-based CF, the similarities between users are calculated whereas item-based CF calculates similarities between items to make a prediction.

Many approaches are introduced in neighborhood-based CF in literature. However, k nearest neighbor (k-nn) algorithm is the most famous and dominant one in user-based and item-based CF [13], [21], [22]. The algorithm begins by calculating similarities between users or items. The algorithm selects the most similar neighbors among all users based on predetermined k value. The algorithm ends with a prediction by using ratings of the k best neighbors. It is important to choose the k value correctly for the performance of the algorithm. On the other hand, a significant issue is that the k is a fixed value. A traditional k-nn algorithm does not allow users to choose their own k value. However, allowing users to select a dynamic k value increases the accuracy of the prediction [23]. Similar to traditional k-nn approaches, an active user, who is looking for a prediction, selects the best k neighbors among all users whose similarities can be computed in the dynamic k neighbors algorithm [23]. This may lead a poor coverage compared to accuracy [23]. In addition to accuracy, coverage is a prominent factor, as well. Therefore, instead of choosing the best k neighbors among all possible users, an active user should select the best k neighbors within users who rated the target item to increase the coverage when dynamic k neighbor selection is utilized. Otherwise, there may be some neighbors in the best k neighbors who have not rated the target item but join the prediction process, such an occurrence reduces the coverage result.

There is another prominent issue that needs to be addressed beyond the coverage. When the correlation between users is calculated, a classical Pearson correlation coefficient (PCC) is used for the similarity calculation between users. It considers commonly rated items of two users even if there is only one common item. However, PCC does not weigh up the number of commonly rated items. When the correlation is calculated as 1.0 between two users, it means that they match perfectly even if these two users have only one co-rated item. However, this result may not be enough to tell that two users are similar due to the fact that there is only one co-rated item. Therefore, as the number of co-rated items between users decreases, the similarity values between users should be decreased. Herlocker et al. [14] propose to add a correlation significance-weighting factor to PCC. With the help of the weighted factor, the similarity value of users with fewer common items is reduced. We propose to use modified PCC in conjunction with the dynamic neighbor selection approach in our work.

The main contributions of this paper are summarized as follows:

- We study how to improve coverage and accuracy results of dynamic *k* neighbors for each user while selecting neighbors among only users who rated the target item.
- We examine how the number of common ratings between two users besides dynamic *k* neighbors approach affects the prediction accuracy and coverage.
- We finally scrutinize how much contribution we achieve over existing methods using two well-known movie-recommender datasets by performing several experiments

The rest of the paper is organized as follows. In Section 2, the related work is given. Section 3 presents detailed information about neighborhood-based CF algorithms and a correlation significance-weighting factor approach. In Section 4, we describe our proposed method. In Section 5, several experiments are performed to evaluate the proposed approaches. Finally, Section 6 represents our conclusion and describes possible future research directions.

2. Related Work

Neighborhood-based schemes are the most popular algorithms used in CF systems, which are also called as memory-based CF [14], [24]-[26]. Neighborhood-based algorithms are divided into two parts in terms of the implementation, either user-based CF [24] or item-based CF [26]. User-based algorithms reveal a correlation between users, whereas item-based algorithms scrutinize relationships between items. The first comprehensive study on the design of the neighborhood-based CF is performed by Herlocker et. al [14]. The researchers tested various parameters to improve the quality of predictions. According to their study, different similarity weight functions have been tested and the PCC outperforms the others. In addition to selecting and calculating similarities, determining similar neighbors is also important in neighborhood-based algorithms. Although the most *k* similar neighbors are selected from descendingly sorted similarities weights [24], threshold-based approaches are also used to select similar neighbors. Kim and Yang [27] introduce a threshold-based approach to identify the best neighbors. If the similarity weight value of users is greater than the predefined value, those users are selected as appropriate neighbors.

Herlocker et. al [14] also point to a different issue. They claim that the number of commonly rated items between users is a significant factor while calculating similarities. Considering the structure of RSs, most of the items are unrated. Due to sparsity, it is difficult to find users with many co-rated items, which may lead to inaccurate predictions. PCC only calculates similarities between users regardless of how many items are commonly rated by two users. Therefore, it is not a good way to select the best neighbors just by PCC. The study in [14] introduced a new term called “a correlation significance-weighting factor” to decrease the effect of the similarity weight of a user when there are a few co-rated items between users. They claim that the more co-rated items two users vote, the more reliable the calculated similarity weight between them would be. Ma et al. [28] also use PCC-based significance weighting instead of classical PCC. The researchers modified the approach which was proposed by Herlocker et. al [14]. Polatidis and Georgiadis [29] proposed to use the enhanced PCC by dividing the similarity process into multi-levels. Their algorithm contains four levels instead of one as in [14]. The number of co-rated items is divided into four parts. The researchers define four positive numbers for each level to add the similarity weights. As the number of co-rated items increases between two users, the value of similarity weight is increased with the predefined positive number. Moreover, the researchers add a threshold value and compare this value with the similarity weight. If the similarity weight of users is greater than the threshold value and the number of co-rated items between users is appropriate for any level, the similarity weight is updated with the value of that level.

There are various studies that propose new approaches for neighborhood selections to increase the accuracy of the predictions. Zeybek and Kaleli [23] propose a dynamic neighbor selection instead of a constant number of neighbors. They claim that choosing the different number of *k* values for each user gives better results than the traditional neighbor selection method. Their algorithm starts by determining the best *k* value offline for each user and stores those values at a table to use at the online prediction process. Their result shows that using dynamic *k* values outperforms the traditional method. Motivated by this work, we examine how to improve the accuracy results using dynamic *k* values. Our study differs from the work in [23]. The researchers select the most *k* similar neighbors regardless of whether the target item is rated or not by those users. In our work, in the offline phase, we dynamically select the most similar *k* neighbors for each user among users who rated the target item. We also add “a correlation significance-weighting factor” to our study. Our aim is to improve the prediction accuracy and coverage of the dynamic *k* neighbor selection method by selecting neighbors among users who rated the target item and introducing the significance-weighting factor into the neighbor selection phase to find more eligible neighbors.

3. Neighborhood-based CF Algorithms

CF systems are based on the fact that users who had similar tastes in the past will likely have similar tastes in the future as well. The first step to form an effective CF system is to select similar users for an active user. There are mainly two approaches to find similar users, which is also called neighbors or neighbor users. These approaches are defined as correlation-based similarity and clustering. In this paper, the correlation-based approach is used to select neighbor users.

Neighborhood-based CF algorithms are commonly used in RS. The underlying approach of neighborhood-based CF is to find similar users when an active user asks a prediction for a target item (*q*). The algorithm starts by computing similarities between an active user and all users in the system. There are numerous methods to compute similarities between the active user and other users. The most popular and prevalent one is PCC [30]. Then, the algorithm selects the most similar *k* users as neighbors of the active user. The algorithm ends by producing a prediction for the active user on *q*. Figure 1 illustrates a general view of the neighborhood-based CF algorithm.

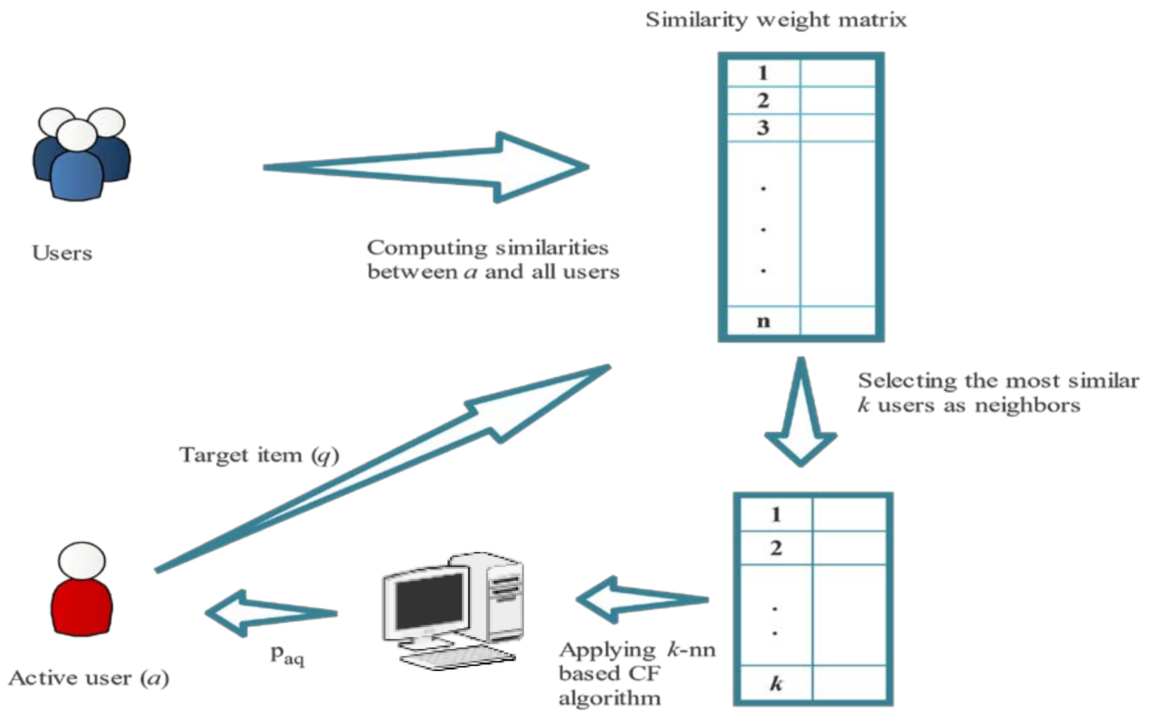

Similarity weight matrix
1
2
3
n
Users
Computing similarities between a and all users
Selecting the most similar k users as neighbors
Target item (q)
1
2
k
Active user (a)
p_{aq}
Applying k-nn based CF algorithm

Figure 1 A General View of the Neighborhood-based CF Algorithm

3.1 Similarity Calculation

PCC is commonly used in user-based CF systems. PCC calculates similarities between an active user and other users as in Equation 1,

$$sim(\boldsymbol{a},\boldsymbol{u}) = \frac{\sum_{\boldsymbol{p}\in\boldsymbol{P}}(\boldsymbol{r}_{\boldsymbol{a},\boldsymbol{p}} - \bar{\boldsymbol{r}}_{\boldsymbol{a}})(\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{p}} - \bar{\boldsymbol{r}}_{\boldsymbol{u}})}{\sqrt{\sum_{\boldsymbol{p}\in\boldsymbol{P}}(\boldsymbol{r}_{\boldsymbol{a},\boldsymbol{p}} - \bar{\boldsymbol{r}}_{\boldsymbol{a}})^2}\sqrt{\sum_{\boldsymbol{p}\in\boldsymbol{P}}(\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{p}} - \bar{\boldsymbol{r}}_{\boldsymbol{u}})^2}} \tag{1}$$

where *P* demonstrates the set of items rated both an active user, a, and a user, u, $\boldsymbol{r}_{\boldsymbol{a},\boldsymbol{p}}$ and $\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{p}}$ represent ratings for item *p* rated by *a* and *u*, respectively. $\bar{\boldsymbol{r}}_{\boldsymbol{a}}$ and $\bar{\boldsymbol{r}}_{\boldsymbol{u}}$ are the mean of *P* items' ratings of a and u. According to Equation 1, similarities are calculated in the range between -1.0 and 1.0. 1.0 illustrates that two users are perfectly matched. It means they are the most similar. -1.0 shows the greatest degree of dissimilarity of two users.

3.2 Selecting *k* Neighbors

After calculating the similarities between users, the most important step is to select appropriate neighbors in order to produce accurate predictions. There are mainly two neighbor selection methods. The first one sorts calculated similarities and then selects top *k* users as the most similar neighbors [24], [25]. According to the second one named correlation weight threshold, a threshold value is defined, and all calculated similarities are compared to that value. If the similarity value is greater than the predefined value, that user is selected as a neighbor of an active user [27]. In this paper, we use the first method and select the most similar *k* users as neighbors.

3.3 Providing Predictions

After calculating similarities between the active user and all users using Equation 1 and selecting the most *k* similar users for the active user, the prediction $\boldsymbol{p}_{\boldsymbol{a},\boldsymbol{q}}$ is calculated as seen in Equation 2,

$$p_{a,q} = \bar{r}_a + \frac{\sum_{u=1}^{k}(r_{u,q} - \bar{r}_u)\, sim(a,u)}{\sum_{u=1}^{k} sim(a,u)} \tag{2}$$

where $\bar{\boldsymbol{r}}_{\boldsymbol{a}}$ and $\bar{\boldsymbol{r}}_{\boldsymbol{u}}$are mean ratings for user *a* and user *u* respectively. *k* represents *k*-nearest-neighbors of a. $\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{q}}$ is the rating of user *u* on item *q* and $\boldsymbol{sim}(\boldsymbol{a},\boldsymbol{u})$ illustrates the similarity value calculated using PCC between user *a* and user *u*.

3.4 A Correlation Significance-Weighting Factor

As mentioned before PCC is the most common method to measure similarities between users and is able to achieve higher performance than other similarity functions [30]. However, PCC does not take the number of co-rated items between users into account. Table 1 shows an example of the user-item matrix. The matrix contains six users and ten items. The first row of the table indicates the active user's ratings. The active user asks for a prediction for item 10.

Table 1 An Example of User-Item Matrix

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
$\boldsymbol{a}$	5		3	4			4			?
$\boldsymbol{u_1}$	4		1	2		2	3			3
$\boldsymbol{u_2}$	3	3	3	4			3			5
$\boldsymbol{u_3}$	2		3	2			4		5	4
$\boldsymbol{u_4}$	1		5	5	1		2	2		1
$\boldsymbol{u_5}$		2		4		2		2	4	5

Table 2 shows the similarity weights between the active user and all users in Table 1. According to Table 2, user 5 is the most similar neighbor of the active user. However, they have one common item and it is hard to tell that they have similar preferences.

Table 2 Similarity Values Between the Active User and Other Users

$\boldsymbol{sim(a, u_1)}$	$\boldsymbol{sim(a, u_2)}$	$\boldsymbol{sim(a, u_3)}$	$\boldsymbol{sim(a, u_4)}$	$\boldsymbol{sim(a, u_5)}$
0,9487	0	-0,4264	-0,7921	1

Herlocker et al. [14] proposed an approach that adds a correlation significance-weighting factor to classical PCC. A recommender system is usually sparse, and it is difficult to find users rated more ratings common with the active user. The more items are rated by the user and the active user, the more they have likely similar preferences. Herlocker et al. [14] proposed to use Equation 3, which adds a correlation significance-weighting factor to the similarity calculation.

$$sim'(a,u) = \left\{\begin{array}{ll} \frac{|I_a \cap I_u|}{SW} * sim(a,u), & if\ |I_a \cap I_u| < SW \\ sim(a,u), & otherwise \end{array}\right\} \tag{3}$$

In Equation 3, $|\boldsymbol{I_a} \cap \boldsymbol{I_u}|$ shows the number of co-rated items, which are rated by the active user and user *u*. SW is a threshold value, which is a constant value. If the value of $|\boldsymbol{I_a} \cap \boldsymbol{I_u}|$ is less than SW, the similarity weight is decreased. Otherwise, the similarity weight remains the same. Note that the more co-rated items between users there are, the more reliable results would be.

4. Proposed Method

It is crucial that how many users are selected as the best neighbors in the *k*-nn based CF algorithms. Most of the algorithms apply the fixed *k* value. It means that the system selects the fixed *k* best users among all users whenever an active user attends in the system and wants a prediction for an item (*q*). Zeybek and Kaleli [23] proposed to use dynamic *k* value for all users to improve accuracy. The researchers show that accuracy results get better while the coverage decreases. Our method improves the overall accuracy and coverage of the dynamic *k* neighbor selection method. After calculating similarities between an active user and all users, similarities are sorted in the traditional *k*-nn based CF algorithm, and top *k* users are selected. However, we propose that the best *k* users should be selected among users who rated the target item, *q*, because some of the selected neighbors may not have rated the target item. Even in the worst case, none of them may not have rated the target item. Such an occurrence is closely associated with a decrease in coverage. Therefore, we first propose to select top *k* neighbors among users who have rated the target item when calculating a dynamic *k* value for each user offline.

We also claim that the number of co-rated items is prominent to get a higher prediction performance. As seen in Table 2, the similarity value between *a* and user 5 is 1.0. It is the greatest similarity value. Is it enough to tell that these two users have the same taste? Unfortunately, Table 1 shows that they have only one common item. A few number of co-rated items between *a* and any user do not reflect the real correlation. The more co-rated items two users have, the more reliable the correlation is. Herlocker et. al [14] proposed to use a correlation significance-weighting factor in order to improve prediction accuracy. According to their experimental results, the significance-weighting factor which is selected 50 gives the best results among other values. We also set the significance-weighting factor to 50 in our experiment. If the number of co-rated items between users is less than the predefined weighting factor, their similarity value was decreased using Equation 3. Otherwise, as seen from Equation 3 the similarity value is equal to classical PCC.

We combine two approaches, significance-weighting factor in the correlation calculation and an update to the neighbor selection, in our work as the second proposed method. After similarities are calculated using Equation 1, we recalculate similarities using Equation 3 based on commonly rated items (significance-weighting). The first step is to identify k value for each user in dynamically. Note that top k neighbors are selected among only users who rated the target item. This is the training phase. The second, testing phase, is to produce predictions using the dynamic k-value selected in the first case.

5. Experiments

Throughout our experiments, we test if the accuracy and the coverage results get better than the existing methods [14], [23]. We perform different experiments to test our intuition and demonstrate how dynamic k values and the significance-weighting factor contribute to the results.

5.1. Datasets and Evaluation Metrics

A well-known recommender system data sets, MovieLens and FilmTrust are used in the experiments. MovieLens is a movie recommendation website (www.grouplens.org) and each rating is based on a 5-star scale. The standard 100K MovieLens data set, including 943 users and 1862 items, is used. FilmTrust is also a movie recommender dataset that consists of 1508 users and 2071 items. The rating values used in dataset are 0.5, 1, 1.5, 2, 2.5, 3, 3.5 and 4. Table 3 shows general information about datasets such as the number of users, items, and ratings and rating range.

Table 3 Datasets

	users	items	ratings	rating scale
MovieLens	943	1682	100000	1-5
FilmTrust	1508	2071	35497	0.5-4

Mean Absolute Error (MAE) is used to measure the accuracy of the recommender system as given in Equation 4, where $\boldsymbol{p_i}$ is the prediction value, $\boldsymbol{o_i}$ is the real rating value, and R is the number of provided predictions. Since this metric measures the error between the predicted and original rating, lower MAE results indicate a better prediction.

$$MAE = \frac{1}{R}\sum_{i=1}^{R}|p_i - o_i| \tag{4}$$

We also measure the coverage using Equation 5. Coverage is the percentage of the items that the recommender system can produce a prediction.

$$coverage = \frac{\#\ of\ provided\ predictions}{\#\ of\ all\ queried\ items\ for\ a\ prediction} \tag{5}$$

5.2. Methodology

Our experiments are divided into two parts: offline and online phase. In all experiments, an active user is selected using the leave-one-out cross-validation method (AllBut1). Each time, one active user is selected from the whole dataset as a test data and the rest of users form our training dataset. After selecting an active user, a prediction is produced for each item (AllBut1) to identify the best k value for each active user offline. All k values are stored in a table for the stage of the prediction process.

In the online part, we randomly select five items among rated items by the active user as in [23] and estimate predictions for those items using dynamic k values and predefined significance weight value. We repeat this experiment 100 times to obtain average prediction results.

5.3. Experimental Results and Discussion

As explained before, each user should pick his value of k dynamically. Possible k values are set 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50. After calculating dynamic k values for each user, these values are stored in the table. When the prediction process is executed, *k* value for the active user is looked up from the table and top *k* neighbors among users who have rated the target item are selected.

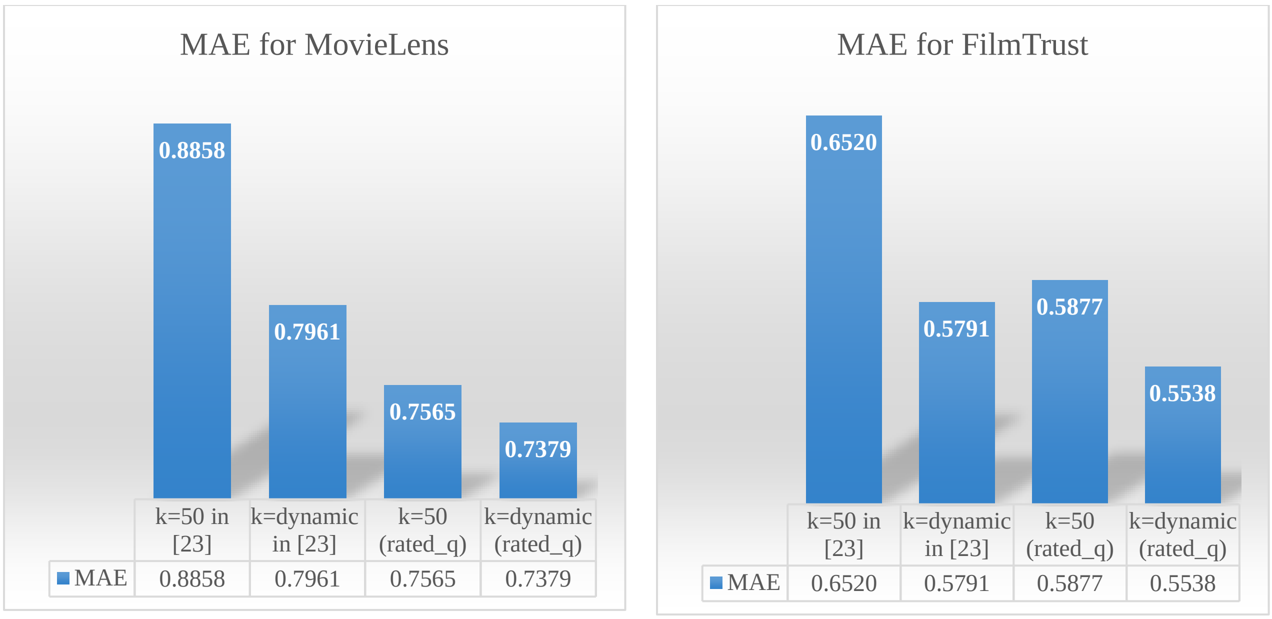
Before the detailed experimental results, we want to demonstrate how dynamic *k* values and MAE change based on the proposed methods. Table 4 shows randomly selected ten users and their calculated dynamic *k* values for FilmTrust dataset. Table 4 contains four columns. The first two columns show the results for only dynamic *k* value and its MAE and the last two columns illustrate the results for the proposed method. The number of co-rated items, which is indeed SW is set 50. "NaN" shows user 4 has no neighbors or co-rated items with any users and his *k* value remains 50 as a default value. In such a case, the *k* value is not important because the prediction cannot be provided for such users.

Table 4 Dynamic *k* Value for FilmTrust Users Based on the Proposed Method

	the best *k*	**MAE**	**the best *k* when SW =50**	**MAE**
u_1	35	0.40045	15	**0.32749**
u_2	10	**0.22344**	5	0.26139
u_3	20	0.52459	5	**0.35026**
u_4	50	NaN	50	NaN
u_5	50	0.69064	10	**0.63377**
u_6	25	1.40790	25	**1.33525**
u_7	15	**1.07709**	50	1.11342
u_8	35	0.47680	20	**0.44062**
u_9	10	0.43242	25	**0.30979**
u_{10}	45	1.39214	5	**1.26631**

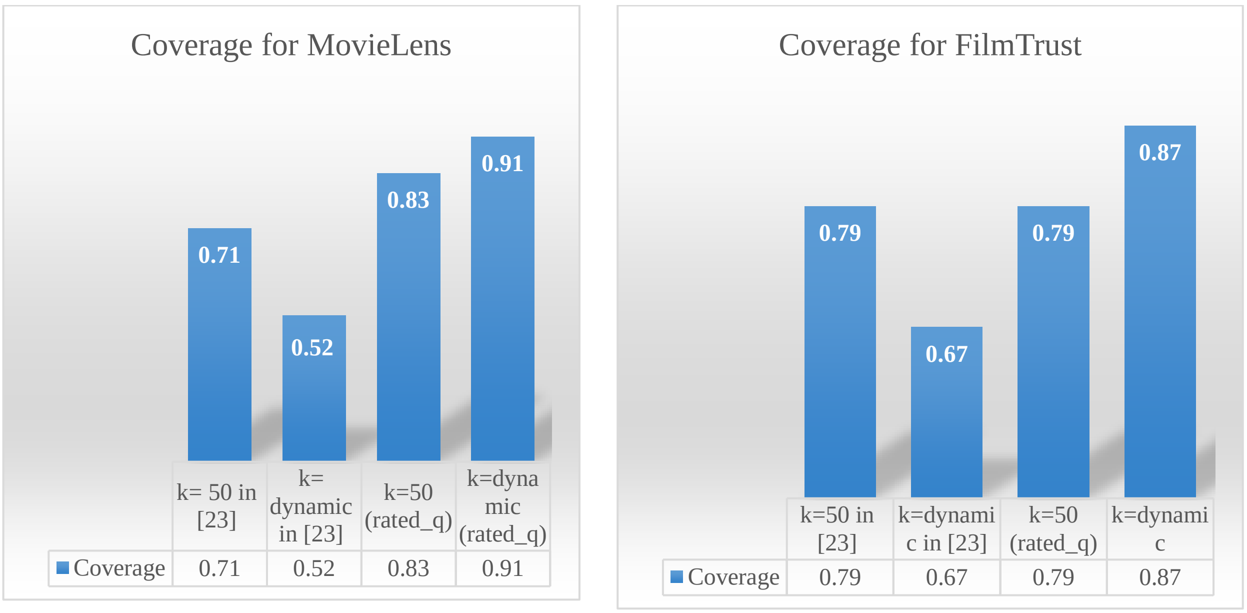
We conduct a new experiment in order to measure the effect of nominating users who rated the target item as possible neighbors in dynamic *k* neighbor selection. Recall that dynamic *k* neighbor selection nominates all users whose correlation/similarity can be calculated [23]. Each user in the dataset is selected as an active user and the rest form our training data. All rated items for each active user are estimated and MAE is calculated. Figure 2 shows the experimental results and the experiment compares four different approaches. The first and third use a fixed *k* value while the second and fourth algorithms use dynamic *k* value. First, we would like to show how MAE results change when neighbors are selected among users who have rated the target item. When *k* is fixed at 50, Figure 2 (a) shows that MAE is 0.8858 when neighbors are selected from all users while MAE is 0.7565 when neighbors are selected from users who rated the target item for MovieLens dataset. Even only choosing a fixed *k* neighbors among users who have rated the target item increases the accuracy dramatically. Furthermore, we also compare dynamic *k* neighbor selection in [23] with our first proposed method. MAE of our first proposed method is 0.7379 while MAE in [23] is 0.7961. Figure 2 illustrates that selecting dynamic *k* neighbors among users who have rated the target item for each active user gets the best accuracy by outperforming the previous two baseline methods and dynamic *k* neighbor selection in [23]. One can see in Figure 2 (b) that the accuracy results for FilmTrust dataset confirm our findings.

Besides accuracy, the coverage is also important in RS. Therefore, we also calculate the coverage values for baseline methods and the first proposed method. As expected, the coverage in our method of nominating users who rate the target item for the fixed *k* neighbor selection is better than nominating all users regardless of their rating on the target item. Furthermore, in dynamic *k* neighbor selection, our proposed method gives the best coverage. As mentioned before, we inspired from an earlier work by Zeybek and Kaleli [23]. The most important difference in our study is to choose the value of *k* from the users who have voted for the target item. When we compare our work with them, our coverage values increase although their coverage decreases. As seen from Figure 3, the coverage for MovieLens is 0.91 and the coverage for FilmTurst is 0.87. However, Zeybek and Kaleli [23] have shown the value of coverage in the range of 0.54 – 0.71 for MovieLens dataset in their work.



(a) (b)

Figure 2 Accuracy Results Based on Selecting *k* (a) Accuracy Results for MovieLens; (b) Accuracy Results for FilmTrust

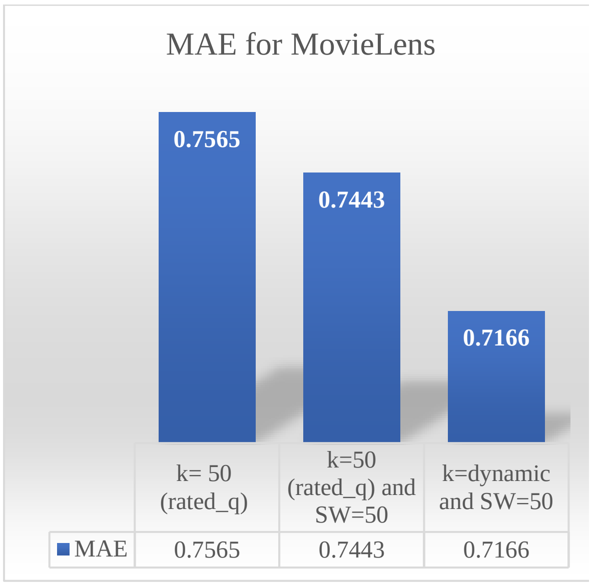


(a) (b)

Figure 3 Coverage Results Based on Selecting *k* (a) Coverage Results for MovieLens; (b) Coverage Results for FilmTrust

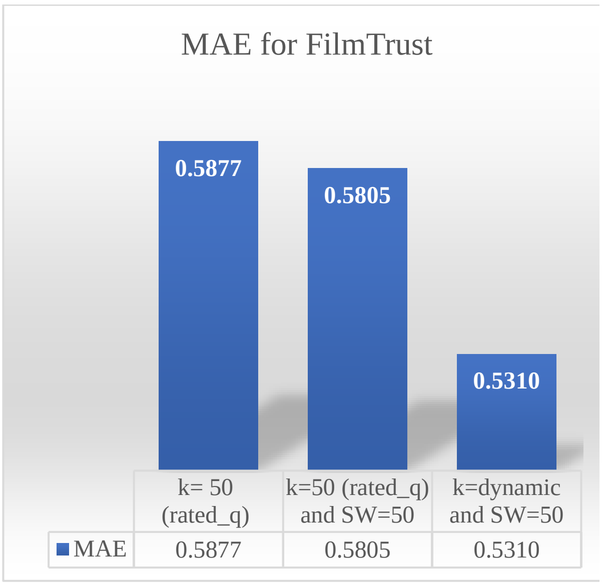
We also test our second proposed approach. The number of co-rated items between users is crucial as much as higher similarity weights. As discussed throughout the paper, higher similarities between users may not always indicate that these two users have common tastes. Recall that the significance-weighting factor tries to balance the similarity of two users by considering the number of commonly rated items they have. The correlation significance-weighting factor is set 50 as in the work [14]. Figure 4 contains two sub-figures which are MAE results for MovieLens and FilmTrust. In this case, our second proposed method is compared with baseline methods. The first result shows the classical PCC with users who have rated the target item while the second result depicts the modified PCC with the significance-weight factor. The third result shows our proposed method’s result and allows the system to decrease similarity weights between users if the co-rated items are smaller than the threshold and choose dynamic *k* values. Figure 4 illustrates that using dynamic *k* value for each user with the significance-weight factor increases predictions of accuracy.

When we compare the results of Figure 4 with Figure 2, MAE for MovieLens of the baseline method, the first proposed method, and the second proposed method is 0.7565, 0.7379 and 0.7166, respectively. A lower MAE means a higher accuracy. FilmTrust dataset shows similar results. MAE for FilmTrust of the baseline method, the first proposed method, and the second proposed method is 0.5877, 0.5538 and 0.5310. The accuracy for the second proposed method gives the best.



	k= 50 (rated_q)	k=50 (rated_q) and SW=50	k=dynamic and SW=50
■ MAE	0.7565	0.7443	0.7166

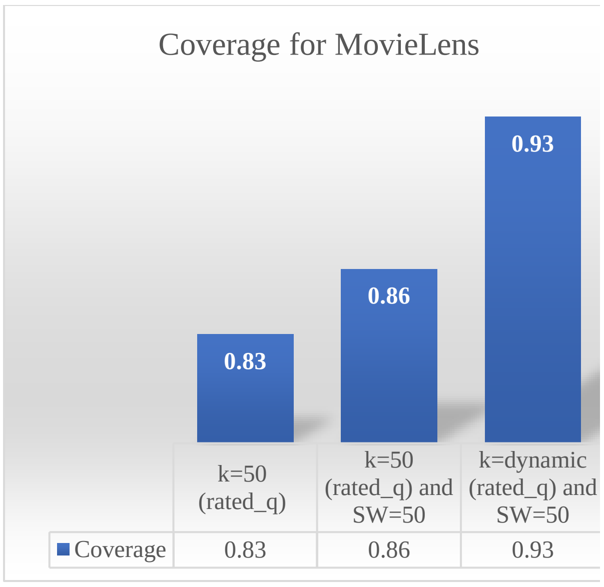
(a)



	k= 50 (rated_q)	k=50 (rated_q) and SW=50	k=dynamic and SW=50
■ MAE	0.5877	0.5805	0.5310

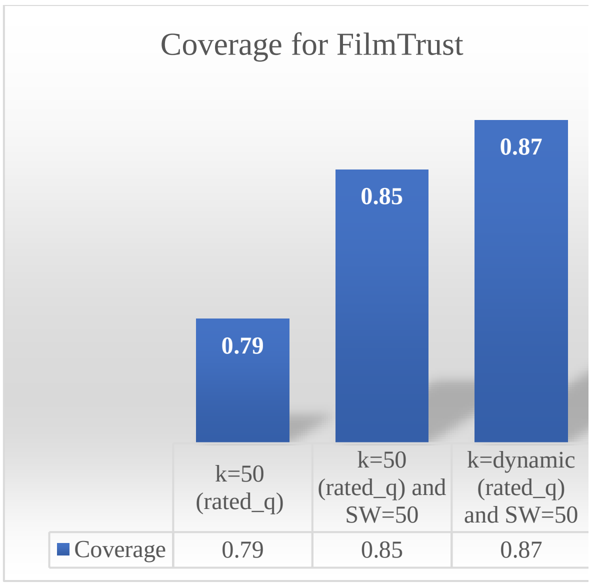
(b)

Figure 4 Accuracy Results Based on Selecting *k* with a Correlation Significance-weighting Factor (a) Accuracy Results for MovieLens; (b) Accuracy Results for FilmTrust



	k=50 (rated_q)	k=50 (rated_q) and SW=50	k=dynamic (rated_q) and SW=50
■ Coverage	0.83	0.86	0.93

(a)



	k=50 (rated_q)	k=50 (rated_q) and SW=50	k=dynamic (rated_q) and SW=50
■ Coverage	0.79	0.85	0.87

(b)

Figure 5 Coverage Results Based on Selecting *k* with a Correlation Significance-weighting Factor (a) Coverage Results for MovieLens; (b) Coverage Results for FilmTrust

After MAE results are given for the second proposed method in Figure 4, coverages for both datasets are measured. As seen from Figure 5, our second proposed method increases the coverage. The coverage of the classical PCC is 0.8308 while the coverage of the proposed method is 0.9247 for MovieLens.

Likewise, the coverage of the traditional PCC is 0.7896 while the coverage of the proposed method is 0.8673 for FilmTrust. Approximately 8 to 10 percent of increase has been achieved with the significance-weighting factor and the dynamic k neighbor selection.

A correlation significance-weighting factor is set 50 in all experiments. However, we also test to see if the results change with different significance-weighting factor values. The values of significance-weighting factors are set 20, 40, 50, 60 and 80. Note that dynamic *k* values for each user are selected in each experiment based on the predefined significance-weighting value offline. The results are shown in Table 5 and Table 6. When the results are examined, they are very close to each other. Table 5 illustrates MAE and coverage results of MovieLens dataset. In previous experiments, we set the significance-weighting factor as 50. However, someone may want to choose the significance-weighting factor as 40 according to this experiment.

Table 5 MovieLens Results Based on Various Significance-weighting (SW)

	SW=20	**SW=40**	**SW=50**	**SW=60**	**SW=80**
MAE	0.7201	**0.7159**	0.7166	0.7162	0.7150
Coverage	0.9236	**0.9251**	0.9247	0.9242	0.9237

The results in Table 6 demonstrate similar trends with Table 5. A correlation significance-weighting factor may be set as 60 according to this experiment. However, the improvement is minimal. The reason why the results are so close is that different *k* neighbors for each user are chosen for each value (SW).

Table 6 FilmTrust Results Based on Various Significance-weighting (SW)

	SW=20	**SW=40**	**SW=50**	**SW=60**	**SW=80**
MAE	0.5358	0.5313	0.5310	**0.5303**	0.5298
Coverage	0.8676	0.8676	0.8673	**0.8674**	0.8668

We also perform t-tests to show whether our proposed approaches results are statistically significant. In our experiments, null hypothesis regards that the prediction performances of the proposed approaches are the same as the baseline method whereas the alternative hypothesis considers that the prediction performances of the proposed approaches are better than the baseline method. Table 7 proves that the proposed approaches are statistically significant at the 99% level of confidence interval (all p values < 0.01) and the alternative hypothesis is accepted. In other words, the proposed approaches improve significantly the prediction performance based on the baseline method.

Table 7 Comparision of t-tests Between Baseline Method and Proposed Approaches

	MovieLens	**FilmTrust**
Baseline method (*k* = 50 in [23]) vs. *k* = dynamic (rated_q)	t = 43.061 p = 4.4131e-225	t = 30.880 p = 1.6734e-154
Baseline method (*k* = 50 in [23]) vs. *k* = dynamic (rated_q) and SW=50	t = 42.093 p = 5.8468e-219	t = 30.764 p = 1.2178e-153

6. Conclusions and Future Work

Neighborhood-based approaches are used commonly in collaborative filtering systems. Researchers find out new ways to improve the accuracy results of neighborhood-based approaches. Some of them propose new best neighbor selection algorithms although some introduce new similarity metrics. *k*-nn based CF algorithm is one of the most common neighbor selection algorithms in collaborative filtering. The important point of the traditional *k*-nn based algorithm is to select a predefined number of users as the best neighbors of the active user. A recent best neighbor selection algorithm offers to select dynamic *k* neighbors for each active user instead of a constant *k* value. However, there is a critical issue when identifying the best neighbors: some of the selected neighbors should not have rated the target item. This might cause a decrease in coverage. During the neighbor selection process, we propose that dynamic *k* neighbors should be selected within users who have rated the target item. This criterion is important to identify the best neighbors. We also propose to use modified PCC instead of classical PCC. In the classical PCC, co-rated items between users participate in the similarity calculation. As the number of co-rated items between users increases, the related correlation metric is likely to be more reliable. This can be achieved by introducing significance weighting in similarity calculation. We combine this approach with the dynamic neighbor selection approach.

As seen from MovieLens results, the coverage of the method which selects the best *k* neighbors regardless of users who rated the target item is 0.71, whereas the coverage of the method which selects the best *k* neighbors among the users who rated the target item is 0.83 in traditional fixed *k* neighbor selection. Besides, when the dynamic *k* neighbors are selected among users who rated the target item, the coverage value is increased to 0.91. The coverage of the proposed method with the significance-weighting even reaches 0.93. Likewise, the coverage results of FilmTrust dataset increase approximately from 0.79 to 0.87. Approximately 8 to 10 percent of increase has been achieved with the significance-weighting factor and dynamic *k* value.

Mean absolute error results also decrease with the proposed methods. Mean absolute error of the baseline method, which does not consider whether the target item is rated or not, is 0.8858 for MovieLens whereas mean absolute error of the second proposed method, which considers if the target item is rated and utilizes significance weighting, is 0.7166. FilmTrust dataset shows similar results and mean absolute errors decrease with the proposed methods. The results show that it is prominent that selected neighbors should be within users who have rated the target item.

We plan to offer a new user similarity metric to improve our proposed method. There are methods that consider co-rated items between two users. These methods can be combined with existing user similarity metrics to improve prediction performance.

References

[1] J. Bobadilla, F. Ortega, A. Hernando and A. Gutiérrez, "Recommender systems survey", *Knowledge-Based Systems*, vol. 46, pp. 109-132, 2013. Available: 10.1016/j.knosys.2013.03.012.

[2] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering", *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, 2003. Available: 10.1109/mic.2003.1167344.

[3] C. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation ", *ACM Transactions on Management Information Systems*, vol. 6, no. 4, pp. 1-19, 2016. Available: 10.1145/2843948.

[4] J. Pérez-Marcos and V. López Batista, “Recommender System Based on Collaborative Filtering for Spotify’s Users,” in Trends in Cyber-Physical Multi-Agent Systems. *The PAAMS Collection - 15th International Conference*, PAAMS 2017, Cham, 2018, pp. 214–220, doi: 10.1007/978-3-319-61578-3_22.

[5] P. B. Thorat, R. M. Goudar, and S. Barve, “Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System,” *IJCA*, vol. 110, no. 4, pp. 31–36, Jan. 2015, doi: 10.5120/19308-0760.

[6] P. Messina, V. Dominguez, D. Parra, C. Trattner, and A. Soto, "Content-based artwork recommendation: integrating painting metadata with neural and manually-engineered visual features," *User Model User-Adap Inter*, vol. 29, no. 2, pp. 251–290, Apr. 2019, doi: 10.1007/s11257-018-9206-9.

[7] P. Lops, D. Jannach, C. Musto, T. Bogers, and M. Koolen, "Trends in content-based recommendation: Preface to the special issue on Recommender systems based on rich item descriptions," *User Model User-Adap Inter*, vol. 29, no. 2, pp. 239–249, Apr. 2019, doi: 10.1007/s11257-019-09231-w.

[8] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, 2009, doi: 10.1155/2009/421425.

[9] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–45, Jul. 2014, doi: 10.1145/2556270.

[10] D. Kluver, M. D. Ekstrand, and J. A. Konstan, "Rating-Based Collaborative Filtering: Algorithms and Evaluation," in Social Information Access, vol. 10100, P. Brusilovsky and D. He, Eds. Cham: Springer International Publishing, 2018, pp. 344–390.

[11] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition," Information Sciences, vol. 180, no. 22, pp. 4290–4311, Nov. 2010, doi: 10.1016/j.ins.2010.07.024.

[12] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, "A hybrid recommender system using artificial neural networks," *Expert Systems with Applications*, vol. 83, pp. 300–313, Oct. 2017, doi: 10.1016/j.eswa.2017.04.046.

[13] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005, doi: 10.1109/TKDE.2005.99.

[14] 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.

[15] C. Kaleli, "An entropy-based neighbor selection approach for collaborative filtering," *Knowledge-Based Systems*, vol. 56, pp. 273–280, Jan. 2014, doi: 10.1016/j.knosys.2013.11.020.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi: 10.1109/MC.2009.263.

[17] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks," *International Journal of Approximate Reasoning*, vol. 51, no. 7, pp. 785–799, Sep. 2010, doi: 10.1016/j.ijar.2010.04.001.

[18] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural Collaborative Filtering,” in *Proceedings of the 26th International Conference on World Wide Web - WWW ’17*, Perth, Australia, 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[19] J. Li *et al.*, “Category Preferred Canopy–K-means based Collaborative Filtering algorithm,” *Future Generation Computer Systems*, vol. 93, pp. 1046–1054, Apr. 2019, doi: 10.1016/j.future.2018.04.025.

[20] X. Ning, C. Desrosiers, and G. Karypis, “A Comprehensive Survey of Neighborhood-Based Recommendation Methods,” in Recommender Systems Handbook, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA: Springer US, 2015, pp. 37–76.

[21] Y. Park, S. Park, W. Jung, and S. Lee, “Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph,” *Expert Systems with Applications*, vol. 42, no. 8, pp. 4022–4028, May 2015, doi: 10.1016/j.eswa.2015.01.001.

[22] D.-K. Chae, S.-C. Lee, S.-Y. Lee, and S.-W. Kim, “On identifying k -nearest neighbors in neighborhood models for efficient and effective collaborative filtering,” *Neurocomputing*, vol. 278, pp. 134–143, Feb. 2018, doi: 10.1016/j.neucom.2017.06.081.

[23] H. Zeybek and C. Kaleli, “Dynamic k Neighbor Selection for Collaborative Filtering,” *Anadolu university journal of science and technology A - Applied Sciences and Engineering*, pp. 1–1, Mar. 2018, doi: 10.18038/aubtda.346407.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: an open architecture for collaborative filtering of netnews,” in *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW ’94*, Chapel Hill, North Carolina, United States, 1994, pp. 175–186, doi: 10.1145/192844.192905.

[25] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, “GroupLens: applying collaborative filtering to Usenet news,” *Commun. ACM*, vol. 40, no. 3, pp. 77–87, Mar. 1997, doi: 10.1145/245108.245126

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the tenth international conference on World Wide Web - WWW ’01*, Hong Kong, Hong Kong, 2001, pp. 285–295, doi: 10.1145/371920.372071.

[27] T.-H. Kim and S.-B. Yang, “An Effective Threshold-Based Neighbor Selection in Collaborative Filtering,” in *Advances in Information Retrieval*, vol. 4425, G. Amati, C. Carpineto, and G. Romano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 712–715.

[28] 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, p. 39, doi: 10.1145/1277741.1277751.

[29] 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

[30] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence,* Madison, Wisconsin, Jul. 1998, pp. 43–52.