SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 5, NO. 2, AUGUST 2022
DOI: 10.35377/saucis.05.02.1065794

Research Article

Effects of Neighborhood-based Collaborative Filtering Parameters on Their Blockbuster Bias Performances

Emre Yalcin[1]
[1]Computer Engineering Department, Sivas Cumhuriyet University; eyalcin@cumhuriyet.edu.tr

Received 31 January 2022; Revised 26 April 2022; Accepted 15 June 2022; Published online 31 August 2022

Abstract

Collaborative filtering algorithms are efficient tools for providing recommendations with reasonable accuracy performances to individuals. However, the previous research has realized that these algorithms propagate an undesirable bias in favor of blockbuster items in their recommendations, resulting in recommendation lists dominated by such items. As one most prominent types of collaborative filtering approaches, neighborhood-based algorithms aim to produce recommendations based on neighborhoods constructed by considering similarities between users/items. Therefore, the utilized similarity function and the size of the neighborhoods are critical parameters for their recommendation performances. This study considers three well-known similarity functions, i.e., Pearson, Cosine, and Mean Squared Difference, and varying neighborhood sizes and observes how they affect the algorithms' blockbuster bias and accuracy performances. The extensive experiments conducted on two benchmark data collections conclude that as the size of neighborhoods decreases, these algorithms generally become more vulnerable to blockbuster bias while their accuracy increases. The experimental works also show that using the Cosine metric is superior to other similarity functions in producing recommendations where blockbuster bias is treated more. However, it leads to having unqualified recommendations in terms of predictive accuracy as they are usually conflicting goals.

Keywords: Recommender systems, neighborhood-based collaborative filtering, blockbuster bias, similarity function, neighborhood size.

1. Introduction

With the increasing Internet usage in recent years, individuals are inevitably faced with a vast amount of available information, making their decision-making process more complicated as they cannot find relevant services/products. Recommender systems (RSs) are highly-effective intelligent devices to cope with such an information overload problem [1]; since they aim to guide individuals by suggesting a list of preferable contents that are filtered out based on their preferences in the past [2]. Due to their significant advantages for both business and user sides, they have become more widespread in many digital systems on the Internet in different areas such as music[1], e-commerce[2], hotel accommodation[3], movies[4], etc.

In a typical recommendation scenario, a user (also called the active user) requests from the RS a numerical prediction for an item (also called the target item) untasted by himself or a ranked recommendation list containing preferable items. Researchers have recently introduced several methods for these recommendation tasks, such as collaborative filtering (CF) [3], content- or demographic-oriented filtering [4], or hybrid strategies [5]. CF techniques are the most prevalent among these methods as they are highly effective in achieving accurate recommendations. Also, studies have long been focusing on enhancing these algorithms in quantitative terms such as scalability, coverage, and accuracy. Finally, according to the following mechanism, CF techniques are commonly classified as memory- or model-based. While former methods usually provide recommendations based on similarities between users/items, the latter construct a model of the preference data for producing recommendations [2].

[1] https://spotify.com/
[2] https://www.ebay.com/
[3] https://www.booking.com/
[4] https://www.netflix.com/

As a prominent type of memory-based CF approaches, k-nearest neighbor (kNN) CF algorithms assume that people who have similar tastes in the past will show similar behaviors in the future [6]. Based on this assumption, they provide recommendations based on the neighborhoods constructed with the most similar users (also known as the user-based kNN) or items (also known as the item-based kNN) by performing a user-item rating matrix that includes the past choices of individuals on items [3]. The recommendation process of kNN CF algorithms is usually a two-step. Initially, it is located like-minded individuals called neighbors, and then a prediction score is computed based on the past preferences of neighbors on the target item. Also, it is a known phenomenon that their general success is firmly bound to the phase of neighborhood formation [7]. Therefore, the previous research has verified that the utilized similarity function and considered neighborhood size become vital parameters in properly locating neighborhoods, and the accuracy performance of the kNN algorithms is strongly related to how the tuning of such parameters [8]–[11].

CF algorithms are generally evaluated based on their accuracy performances. However, recent research on RSs has realized that CF algorithms are strongly biased towards popular and highly-liked items, also called the blockbuster items [12], [13], in their produced referrals due to their internal mechanism or imbalances in the rating data. This issue leads to having ranked lists where such a few blockbuster items in the catalog have appeared too often, while other vast items can not get the deserved chance even when they might be desirable for users. Unfortunately, such blockbuster bias propagation of the CF algorithms leads to low-qualified recommendations for beyond-accuracy dimensions like coverage and diversity [12]. In addition, this bias leads to having a system where unfair competition occurs, as the products of different providers are not equally treated. Moreover, this issue makes the system more unguarded to shilling attacks or social bots of malicious stakeholders to increase the visibility of their products and thus sale rates. Therefore, recent research on RSs has aimed to profoundly investigate the impacts of such a bias against blockbuster or popular items in recommendations and develop beneficial treatment approaches to counteract its adverse effects [13]-[15].

The presented study comprehensively evaluates how the blockbuster bias propagated by the kNN algorithms differentiates based on their parameter tuning. In the following, we summarize the main contributions of our study.

1. We consider three famous similarity functions: Pearson Correlation Coefficient, Cosine Similarity, and Mean Squared Difference Similarity. We observe how they affect the blockbuster bias of two prominent kNN CF methods, i.e., user- and item-based kNN, via an adopted efficient blockbuster bias evaluation protocol on two real-world datasets.
2. We also consider varying neighborhood sizes when applying kNN algorithms and investigate how they are related to the blockbuster bias in produced recommendations.
3. In addition, we analyze how these parameters of kNN algorithms affect the quality of the provided recommendation lists in terms of predictive accuracy.

We organize the remaining of this paper as follows: Section 2 gives a literature review on bias issues, especially those towards blockbuster items, in RSs. Section 3 gives some background information about our study, including the working mechanism of the kNN CF algorithms and blockbuster items. Section 4 presents the experimental studies realized to analyze how blockbuster bias performance of the kNN CF algorithms changes based on their parameter-tuning. Finally, Section 5 concludes the presented work and introduces our future directions.

2. Related work

In recent years, one of the main concerns of RSs has been exploring bias issues in recommendations, such as position [16], selection [17], conformity [18], and popularity [19], and treating their adverse effects on recommendation quality [14], [20], [21].

Popularity bias is the most prominent among such bias types, and it is known as the intrinsic tendency of recommendation algorithms to recommend popular items too frequently while not giving the unpopular ones enough chance [19]. Therefore, previous research has primarily aimed at exploring the

degree of popularity bias induced by different recommendation strategies for different areas like music [22], movies [23], and online education [24]. Also, several studies attempt to explore how the parameter-tuning of some CF algorithms affects their popularity bias performance [19], [24]. Besides, several previous research attempts to develop efficient procedures to achieve more qualified referrals by treating this problem. The existing popularity bias treatment approaches are usually classified as pre-processing, in-processing, and post-processing [21], according to how they are involved in the phase of recommendation generation. More specifically, pre-processing methods aim to decrease the degree of imbalances in the original rating matrix where algorithms are trained [25]. The methods of in-processing try to modify the mechanism of the recommendation algorithms for achieving recommendations where popularity bias is treated [26]. Finally, post-processing methods create new recommendation lists or re-sort products in the produced ranked lists [21], [27].

In a recent study [12], the authors have evaluated item popularity from a different perspective and hypothesized that the popularity of a product does not always mean that it is strongly preferable for users or vice versa. Therefore, they consider blockbuster items, both popular and highly-liked by users, and show that some well-known recommendation algorithms, including neighborhood-based CF ones, are unfortunately biased in favor of such blockbuster items in generated recommendations. In other words, they have introduced a new bias type, referred to as blockbuster bias, in recommendations. To achieve more diverse recommendations by mitigating this bias issue, they have also introduced an efficient post-processing method that motivates re-sorting the produced ranked lists by penalizing blockbuster items [13]. However, more explorative analyses of blockbuster bias in recommendations are required by considering the parameters of the CF algorithms.

Considering neighborhood-based algorithms are the most used CF methods, many previous studies have analyzed the parameterization of these algorithms on the success of recommendations [8], [9], [11]. However, these studies usually consider predictive accuracy and examine how the similarity function and neighborhood size affect the accuracy performances of the algorithms. Such analyses are also performed for different types of RSs, such as multi-criteria [28] and group recommender systems [29].

However, to the best of our knowledge, there is no study investigating how such parameters of neighborhood-based algorithms influence their bias issues, especially those towards blockbuster items. Therefore, this study mainly aims to elaborate on how the blockbuster bias of the neighborhood-based CF methods changes according to their parameter-tuning.

3. Preliminaries

This section introduces background information on the kNN CF algorithms, similarity functions, and blockbuster items.

3.1. The kNN CF algorithms

In traditional RSs, the kNN algorithms are the most prominent approaches to providing referrals to individuals. They operate a user-item rating matrix that contains preference information from *n* users to *m* items. Such preferences are mostly numerical values in a specified rating scale or sometimes binary ratings such as like or dislike, depending on the choices of the service providers. During an online interaction with an RS, an active user ($\boldsymbol{a}$) requests a prediction value for a target item ($\boldsymbol{q}$) after sharing her available preferences. The kNN algorithms perform the prediction estimation process in two steps: *(i)* initially determining neighbors by calculating correlations/similarities between $\boldsymbol{a}$ and all other available individuals in the system, and then *(ii)* computing a prediction value as a weighted average based on neighbors' ratings on $\boldsymbol{q}$. Such correlations between $\boldsymbol{a}$ and any user $\boldsymbol{u}$ can be computed using various methods referred to as similarity functions [8], [30]. In this study, we consider three famous similarity functions explained in detail in the following.

- **Pearson Correlation Coefficient (Pearson):** This similarity function is a type of correlation coefficient that represents the linear relationships between two variables measured on the same ratio scale [31]. Also, the Pearson measures the strength of the association between two

continuous variables. More formally, in RSs domain, it calculates the similarity ($\boldsymbol{w_{au}}$) between $\boldsymbol{a}$ and any user $\boldsymbol{u}$, as in the formula given in Equation 1.

$$\boldsymbol{w_{au}} = \frac{\sum_{i \in I_{au}} [(\boldsymbol{r_{ai}} - \overline{\boldsymbol{r_a}})(\boldsymbol{r_{ui}} - \overline{\boldsymbol{r_u}})]}{\sqrt{\sum_{i \in I_{au}} (\boldsymbol{r_{ai}} - \overline{\boldsymbol{r_a}})^2} \sqrt{\sum_{i \in I_{au}} (\boldsymbol{r_{ui}} - \overline{\boldsymbol{r_u}})^2}} \tag{1}$$

where $\boldsymbol{r_{ai}}$ and $\boldsymbol{r_{ui}}$ denote the ratings for the item $\boldsymbol{i}$ by users $\boldsymbol{a}$ and $\boldsymbol{u}$, respectively. Similarly, $\overline{\boldsymbol{r_u}}$ and $\overline{\boldsymbol{r_a}}$ demonstrate the average votes of $\boldsymbol{u}$ and $\boldsymbol{a}$, respectively, and $\boldsymbol{I_{au}}$ indicates the set of co-voted items by both $\boldsymbol{a}$ and $\boldsymbol{u}$.

To calculate the similarity value ($\boldsymbol{w_{qt}}$) between $\boldsymbol{q}$ and any item $\boldsymbol{t}$, the Pearson function can be modified as in the formula given in Equation 2.

$$\boldsymbol{w_{qt}} = \frac{\sum_{u \in U_{qt}} [(\boldsymbol{r_{uq}} - \overline{\boldsymbol{r_q}})(\boldsymbol{r_{ut}} - \overline{\boldsymbol{r_t}})]}{\sqrt{\sum_{u \in U_{qt}} (\boldsymbol{r_{uq}} - \overline{\boldsymbol{r_q}})^2} \sqrt{\sum_{u \in U_{qt}} (\boldsymbol{r_{ut}} - \overline{\boldsymbol{r_t}})^2}} \tag{2}$$

where $\boldsymbol{r_{uq}}$ and $\boldsymbol{r_{ut}}$ are the votes for $\boldsymbol{u}$ on items $\boldsymbol{q}$ and $\boldsymbol{t}$, respectively, and $\overline{\boldsymbol{r_q}}$ and $\overline{\boldsymbol{r_t}}$ denote the average votes of $\boldsymbol{q}$ and $\boldsymbol{t}$, respectively. The $\boldsymbol{U_{qt}}$ denotes the set of individuals who rated both $\boldsymbol{q}$ and $\boldsymbol{t}$.

- **Cosine similarity (Cosine):** In traditional settings, the Cosine metric calculates the cosine of the angle observed between two vectors in n-dimensional [9]. It can be considered the dot product of such vectors divided by the product of their magnitudes or lengths. Therefore, the Pearson explained above can be considered as centered cosine similarity. More formally, in an RS scenario, it calculates the similarity ($\boldsymbol{w_{au}}$) between $\boldsymbol{a}$ and any user $\boldsymbol{u}$, as in the formula given in Equation 3.

$$\boldsymbol{w_{au}} = \frac{\sum_{i \in I_{au}} \boldsymbol{r_{ai}}\, \boldsymbol{r_{ui}}}{\sqrt{\sum_{i \in I_{au}} \boldsymbol{r_{ai}}^2} \sqrt{\sum_{i \in I_{au}} \boldsymbol{r_{ui}}^2}} \tag{3}$$

or the similarity value ($\mathbf{w_{qt}}$) between $\mathbf{q}$ and any item $\mathbf{t}$, as in the formula given in Equation 4.

$$\boldsymbol{w_{qt}} = \frac{\sum_{u \in U_{qt}} \boldsymbol{r_{uq}}\, \boldsymbol{r_{ut}}}{\sqrt{\sum_{u \in U_{qt}} \boldsymbol{r_{uq}}^2} \sqrt{\sum_{u \in U_{qt}} \boldsymbol{r_{ut}}^2}} \tag{4}$$

- **Mean Squared Difference (MSD**): It is based on the geometrical principles of the well-known Euclidean distance metric [8] and computes the similarity ($\boldsymbol{w_{au}}$) between $\boldsymbol{a}$ and any user $\boldsymbol{u}$, as in the following.

$$\boldsymbol{w_{au}} = \frac{\mathbf{1}}{\left(\frac{\mathbf{1}}{|\boldsymbol{I_{au}}|} \sum_{i \in I_{au}} (\boldsymbol{r_{ai}} - \boldsymbol{r_{ui}})^2\right) + \mathbf{1}} \tag{5}$$

or the similarity value ($\boldsymbol{w_{qt}}$) between $\boldsymbol{q}$ and any item $\boldsymbol{t}$, as in the formula given in Equation 6.

$$\boldsymbol{w_{qt}} = \frac{\mathbf{1}}{\left(\frac{\mathbf{1}}{|\boldsymbol{U_{qt}}|} \sum_{u \in U_{qt}} (\boldsymbol{r_{uq}} - \boldsymbol{r_{ut}})^2\right) + \mathbf{1}} \tag{6}$$

where $+\mathbf{1}$ is utilized to shirk dividing by zero, which is the case of any co-rated item set not being constructed.

After computing similarities between users (or items), the most similar $\boldsymbol{k}$ users (or items) are labeled as neighbors. The user-based kNN algorithm produces a prediction for $\boldsymbol{a}$ on $\boldsymbol{q}$, denoted with $\boldsymbol{p_{aq}}$, as a weighted average of the ratings of active user's neighbors on $\boldsymbol{q}$, using the formula given in Equation 7.

$$\boldsymbol{p_{aq}} = \overline{\boldsymbol{r_a}} + \frac{\sum_{\boldsymbol{u}\in\boldsymbol{k}}[(\boldsymbol{r_{uq}} - \overline{\boldsymbol{r_u}}) \times \boldsymbol{w_{au}}]}{\sum_{\boldsymbol{u}\in\boldsymbol{k}} \boldsymbol{w_{au}}} \tag{7}$$

or the item-based kNN estimates the $\boldsymbol{p_{aq}}$ value as the weighted average of the ratings of the target item's neighbors on $\boldsymbol{q}$, as in Equation 8.

$$\boldsymbol{p_{aq}} = \overline{\boldsymbol{r_q}} + \frac{\sum_{\boldsymbol{j}\in\boldsymbol{k}}[(\boldsymbol{r_{ui}} - \overline{\boldsymbol{r_j}}) \times \boldsymbol{w_{qj}}]}{\sum_{\boldsymbol{j}\in\boldsymbol{k}} \boldsymbol{w_{qj}}} \tag{8}$$

where $\boldsymbol{w_{au}}$ represents the similarity weight between the user $\boldsymbol{a}$ and $\boldsymbol{u}$ for the user-based kNN algorithm. Similarly, for the item-based kNN algorithm, $\boldsymbol{w_{qj}}$ indicates the similarity weight between items $\boldsymbol{q}$ and $\boldsymbol{j}$.

3.1. Definition of blockbuster items

In RSs, the blockbuster term is used to describe items that are both popular (i.e., evaluated by many users) and highly liked (i.e., evaluated with high ratings) at the same time. The literature has verified that recommendation algorithms are biased toward such blockbuster items and therefore produce recommendation lists a few blockbuster items have dominated [12]. On the other hand, many other items in the catalog are under-represented in the produced recommendation lists. Although recommending such blockbuster items seem to be desired for satisfying users, it can negatively affect the system's overall success for some critical aspects. For example, featuring only blockbuster items makes it difficult to discover new items and, therefore, negatively affects getting diverse recommendations. Also, any system subject to blockbuster bias might lack opportunities to discover more obscure items, resulting in a system where a few large service providers or well-known products have dominated. Therefore, such a system becomes more homogeneous; thus, it offers fewer options for creativity and diversification.

Recent studies have analyzed potential bias issues in favor of blockbuster items in generated recommendations [12] and tried to alleviate its adverse effects to achieve more qualified recommendations on beyond-accuracy sides such as coverage, diversity, and novelty [13]. In doing so, their primary concern is how blockbuster items should be defined and formulated. A relevant study has proposed a practical formulation that labels whether a product/item is blockbuster or not based on the characteristics of its received votes. In this study, we have adopted this strategy in analyzing blockbuster bias performances of the kNN algorithms.

This strategy mainly aims to calculate the blockbuster level for the item by incorporating its popularity and liking-degree through harmonic combination. Suppose that r_{ui} is the rating value of user u on the item i, it initially determines a popularity value for the i referred as to P_i by considering total number of individuals who rated i, and a liking-degree value referred as to ld_i for i, as in the following.

$$\boldsymbol{ld_i} = \frac{\sum_{\boldsymbol{u}\in\boldsymbol{U_i}} \boldsymbol{r_{ui}}}{|\boldsymbol{P_i}|} \tag{9}$$

where U_i is the set of users who provided a rating for i.

Accordingly, the obtained P_i values inevitably vary a broader interval when compared to ld_i. For the former, the maximum possible value corresponds to the total number of users, while for the latter, it equals the highest rating score in the used rating scale. Such differences in observed values require a normalization process to scale them on the same interval before operating a combination process. To this end, this strategy firstly transform P_i and ld_i values into [0, 1] interval through well-known min-max normalization, and then incorporates the normalized $\overline{P_i}$ and $\overline{ld_i}$ using a harmonic combination procedure to calculate B_i scores that indicate the blockbuster levels of the items, as in Equation 10. The followed harmonic combination strategy is also helpful in balancing the potential trade-off between such

two properties of items as they might be conflicting properties in some circumstances. In other words, the obtained B_i scores properly reflect such two features of the products.

$$\boldsymbol{B_i = \frac{2 \times \overline{P_i} \times \overline{ld_i}}{\overline{P_i} + \overline{ld_i}}} \tag{10}$$

4. Evaluating how kNN parameters affect their blockbuster bias performance

This section presents and discusses the observed outcomes of extensive experimental studies performed to observe how the parameters of the kNN algorithms affect the blockbuster bias of these algorithms. The following sections give detailed information about the used data collections, evaluation metrics, experimental findings, and discussions.

4.1. Datasets

We have used two real-world publicly-available benchmark data collections in the conducted trials, namely MovieLens-100K (MLP) and MovieLens-1M (MLP), released by GroupLens Research Team[5]. Both include users' preferences for movies, and the ratings are discrete and on a five-star rating scale. Table 1 presents the properties of the MLP and MLM datasets.

Table 1 Details of used datasets

Dataset	Number of Users	Number of items	Number of Ratings	Density (%)
MLP	943	1,682	100,000	6.3
MLM	6,040	3,952	1,000,209	4.3

4.2. Evaluation protocols

To analyze the blockbuster bias of the kNN algorithms, we have adopted *Blockbuster Recommendation Frequency* (BRF) metric that has recently been proposed to measure how much blockbuster items overwhelm the generated top-*N* ranked lists [13]. When using the BRF metric, it is required to categorize items in the catalog as the blockbuster or not. To this end, we sort items in descending order according to their blockbuster level scores calculated using the formula given in Equation 10, and split them into two different classes, i.e., *head* and *tail*, through the well-known Pareto principle [32]. Hence, items in the *head* class are the most blockbuster, which have received 20% of all ratings in the dataset. On the other hand, the tail class contains the remaining underappreciated items in the catalog.

After determining *head* and *tail* item classes, the BRF measures the blockbuster bias degree of a recommendation algorithm in a two-step process: *(i)* It initially counts how many times *head* items have been observed in generated top-*N* lists, and then *(ii)* calculates its ratio to the total number of recommended items. Therefore, higher BRF results indicate more unwanted blockbuster bias and worse top-*N* lists in terms of beyond-accuracy perspectives.

More formally, assume that $\mathbb{N}$ is the multiset of the recommendations generated by stacking top-*N* ranked lists provided for each individual by a proper recommendation method. The BRF score of the employed method is estimated as in Equation 11.

$$\boldsymbol{BRF = \frac{\sum_{i \in \mathbb{N}} \mathbb{1}\,(i \in H)}{|\mathbb{N}|}} \tag{11}$$

where $\boldsymbol{H}$ is the set of items in the *head* class.

To better understand how the BRF metric works, we provide a toy example in the following. Assume that $\{\boldsymbol{i_1}, \boldsymbol{i_2}, \ldots, \boldsymbol{i_5}\}$ is the set of available items in the system, and $\boldsymbol{i_1}$ and $\boldsymbol{i_5}$ are the items classified as the *head*. Also, suppose that there exist two available individuals and generated top-3 lists through any

[5] http://www.grouplens.org/

algorithm for them are $\{\boldsymbol{i_4}, \boldsymbol{i_2}, \boldsymbol{i_1}\}$ and $\{\boldsymbol{i_1}, \boldsymbol{i_5}, \boldsymbol{i_2}\}$. For such a recommendation scenario, the multiset of top-*N* lists, i.e., $\mathbb{N}$, is constructed as $\{\boldsymbol{i_4}, \boldsymbol{i_2}, \boldsymbol{i_1}, \boldsymbol{i_1}, \boldsymbol{i_5}, \boldsymbol{i_2}\}$. Thus, the BRF value for the employed algorithm is computed as the ratio of how many times the items in the *head* class, i.e., $\boldsymbol{i_1}$ and $\boldsymbol{i_5}$, have appeared in the $\mathbb{N}$ to the size of the $\mathbb{N}$, which is equivalent to 3/6=0.5. This calculated BRF value demonstrates that half of the recommended items are in the *head* class, and this observation concludes that an unwanted bias towards blockbuster items has occurred in the recommendations.

In the evaluation phase, we also measure the accuracy performances of the kNN algorithms via the normalized Discounted Cumulative Gain (nDCG) metric [13], [33], which is widely used in previous research on RSs. It is a metric aimed at measuring the quality level of the suggested items by considering their actual ratings and their positions in the produced top-*N* recommendation lists. Suppose that r_{ui} is the actual vote of user u on the item i and $\{i_1, i_2, \dots, i_N\}$ is the produced top-*N* item list for u, then the Discounted Cumulative Gain (DCG) and nDCG for that user are calculated as in Equations 12 and 13, respectively.

$$\boldsymbol{DCG_N^u = r_{u,i_1} + \sum_{n=2}^{N} \frac{r_{u,i_n}}{\log_2(n)}} \tag{12}$$

$$\boldsymbol{nDCG_N^u = \frac{DCG_N^u}{IDCG_N^u}} \tag{13}$$

where $IDCG_N^u$ is the maximum possible gain for user u, and it is observed by re-sorting *N* items to achieve the perfect order for u based on her actual ratings. Note that higher nDCG scores mean more accurate recommendation lists.

4.3. Experimentation strategy

As the experimentation methodology, we have followed the well-known all-but-one strategy. Accordingly, we consider one of the users in the original data as the test user and the remaining ones as the train set. For each item of the test user, we produce a prediction value applying the user- or item-based kNN algorithm on the train set and then select the top-10 items with the highest prediction scores as the recommendation list for the test user. This process is repeatedly performed for each user in the data collection. When applying the user- or item-based kNN algorithm, we also consider different similarity measures and neighborhood sizes to monitor how they affect the recommendation quality in terms of blockbuster bias in recommendations. Finally, we measure the quality of the top-10 lists produced for each user with BRF and nDCG metrics and average them to achieve final BRF and nDCG scores for each considered kNN variant. We employ a well-known Python library named Surprise[6] to implement the kNN algorithms.

4.4. Experiment results

In this section, we present the results of the experiments realized to investigate how the parameters of the kNN algorithms affect both their blockbuster bias and accuracy performances. Our experiments consider three similarity functions, i.e., Pearson, Cosine, and MSD, and six maximum neighborhood sizes (*k*) varying from 5 to 100. Accordingly, we first present the BRF results of top-10 recommendations obtained on both MLP and MLM datasets for two variants of kNN algorithms, i.e., UserKNN and ItemKNN, in Figures 1 and 2, respectively.

[6] http://surpriselib.com/

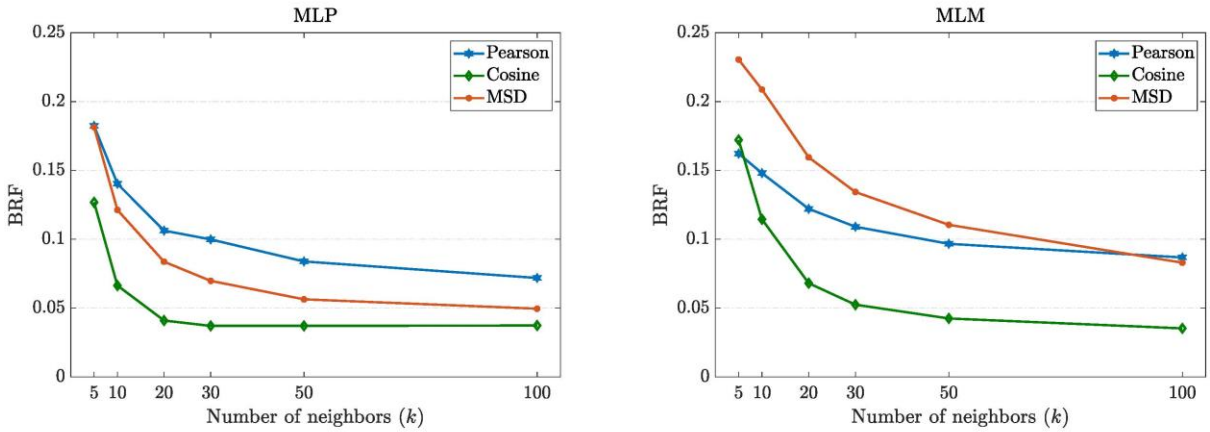

Figure 1 BRF results for the UserKNN algorithm

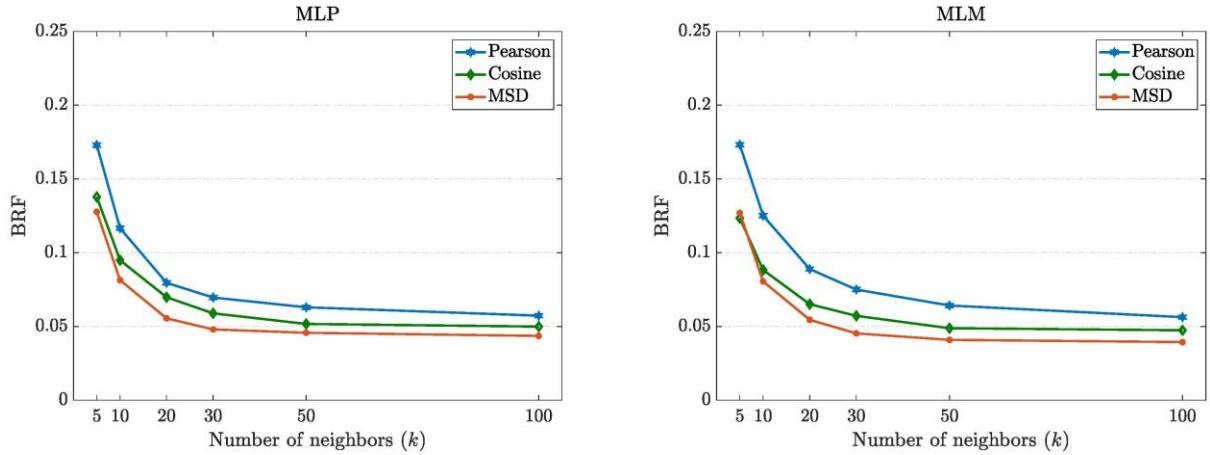

Figure 2 BRF results for the ItemKNN algorithm

As shown in Figures 1 and 2, the highest BRF results, i.e., the most blockbuster biased recommendations, are obtained for both UserKNN and ItemKNN algorithms when the number of neighbors (k) is selected as 5. However, the BRF results obtained for both algorithms significantly decrease as the value of k increases, concluding that they become less biased towards blockbuster items with large neighborhoods. Such BRF trends of the algorithms also hold regardless of the utilized dataset and similarity functions. However, as can be seen from Figures 3 and 4, choosing relatively larger neighborhoods leads to significant decreases in the ranking accuracy performance of both algorithms due to the well-known trade-off between recommendation accuracy and diversification [11]. The obtained results also suggest that both UserKNN and ItemKNN algorithms show similar BRF performances for the MLP. However, the former algorithm seems to be more vulnerable to blockbuster bias than the latter for the MLM dataset. A similar trend is observed for nDCG scores; both algorithms have identical nDCG scores for the MLP, but the UserKNN is superior to the ItemKNN for the MLM dataset.

The experimental results also indicate that the obtained BRF and nDCG results converge at a constant level with neighbors bigger than 50, even if there are slight decrements for the MLM dataset for neighborhood size larger than 50. This finding is because of the size of the utilized dataset, making it challenging to find at least k numbers of similar individuals who have rated target items or k numbers of similar products evaluated by the active users. It leads to obtaining identical BRF or nDCG results after a specific number of neighbors, and such threshold value is relatively smaller for MLP than the MLM as the number of available users/items in the MLP is relatively smaller than those in the MLM dataset.

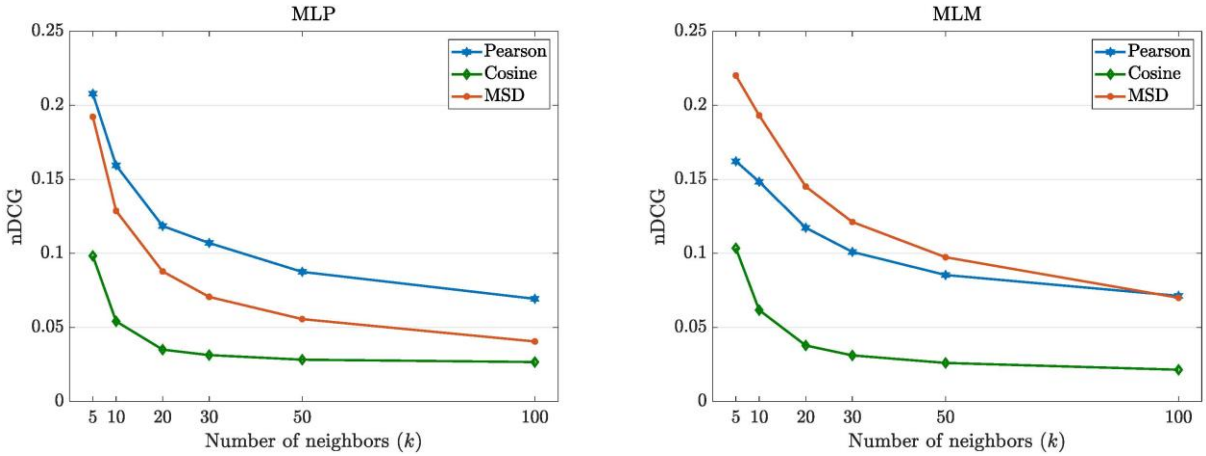

Figure 3 nDCG results for the UserKNN algorithm

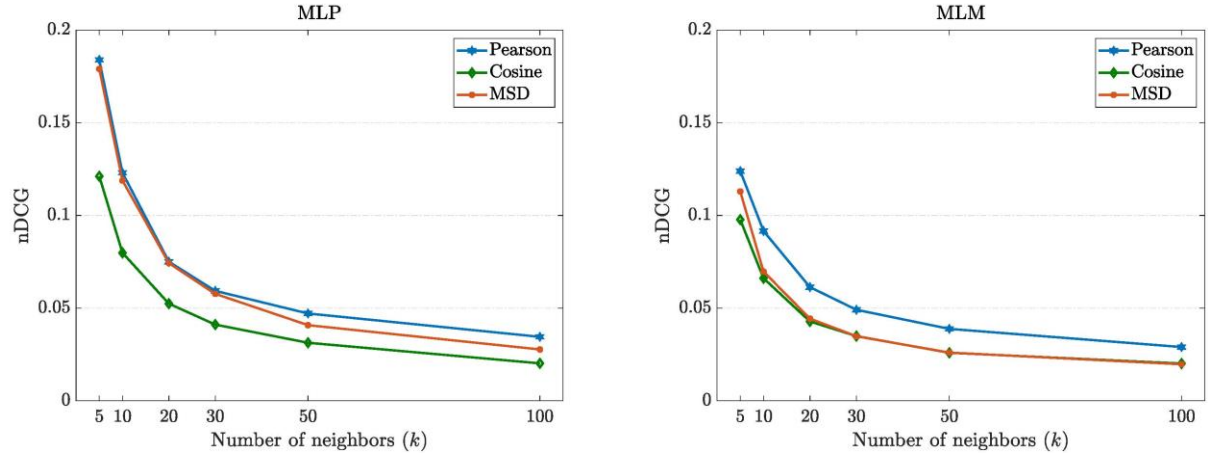

Figure 4 nDCG results for the ItemKNN algorithm

As shown in Figures 3 and 4, the worst recommendations in terms of accuracy are usually obtained when the Cosine similarity function is utilized. However, it seems to be the most robust metric against blockbuster bias, as shown from the BRF results in Figures 1 and 2. On the other hand, as shown in Figures 3 and 4, Pearson is the most prominent similarity function in terms of ranking accuracy, except only for the UserKNN algorithm on the MLM dataset. The underlying reason for the success of the Pearson metric is that it carries out an adjusted normalization step with the average of the ratios of each user since there may be users with an inclination to evaluate very negatively and others with a propensity to evaluate very positively. This finding is also strongly parallel with the outcomes in previous related studies [8]. Even if employing the Pearson metric for kNN algorithms helps produce recommendations with high accuracy, it propagates a significant bias in favor of blockbuster items in the referrals, as depicted in Figures 1 and 2. One explanation for this finding is that the accuracy and beyond-accuracy qualities of the recommendations are usually assumed as conflicting goals.

The experimental results also demonstrate that the kNN algorithms, especially the UserKNN, slightly achieve higher BRF results for the MLM when compared to MLP dataset, as can be followed in Figures 1 and 2. This observation is caused by the sparsity ratio of the data collection; the CF algorithms become more inclined to feature the most blockbuster items in the generated top-*N* ranked lists since the dataset on which they are trained becomes sparser. However, it positively affects the recommendation accuracy since the algorithms show more successful nDCG performances for the MLP dataset, followed by Figures. 3 and 4.

4.5. Insights and discussion

One of the most important findings of our analysis is that both user- and item-based kNN algorithms become less biased blockbuster items with large neighborhoods. The main reason for this observation is that as the neighborhood size increases, more users (for the UserKNN) or items (for the ItemKNN) contribute to the computed prediction score. Therefore, users' degree of variance is averaged out over the more significant numbers, which improves the chance of including non-popular or not highly-liked items into recommendation lists. This fact enables to produce more diverse and thus less blockbuster-biased ranked lists. On the other hand, selecting relatively larger neighborhoods provides more accurate recommendations for both algorithms. This observation also verifies the trade-off between accuracy and biased performances of the algorithms.

Our study also concludes that the Cosine is the worst similarity metric in predictive accuracy. One explanation for this finding is that this metric considers individuals who rated very different votes as highly similar users. For example, in a [1, 5] rating scale, if an individual rated two items as strongly bad (1, 1) and another one rated them as highly perfect (5, 5), this similarity function becomes ultimately unsuccessful since it outputs the maximum likelihood of similarity between these quite different individuals, by its nature. Even if it is assumed that the probability of maintaining the proportionality in the ratios of two individuals is low for larger datasets, this metric still leads to achieving the worst ranking accuracy performance in our experiments. However, another important finding of our analysis is that such drawback of the Cosine ends with improving beyond-accuracy aspects of recommendations and, as a result, provides having less blockbuster biased ranked lists compared to both Pearson and MSD metrics.

5. Conclusion and future work

As a prominent type of memory-based collaborative filtering (CF) algorithms, neighborhood-based, i.e., also referred to as k-nearest neighbor (kNN) CF methods, are widely used in recommender systems due to their success in providing personalized recommendations. It is a known phenomenon that parameter-tuning, such as similarity function and neighborhood size, significantly impacts their accuracy performances when locating neighborhoods. Also, they are subject to blockbuster bias issues, i.e., they expose blockbuster (i.e., both popular and highly-liked) items more in their recommendations than other ones.

This study mainly focuses on evaluating how the parameters of two well-known kNN algorithms affect their blockbuster bias performances through an efficient evaluation protocol. We consider three different similarity functions, namely Pearson, Cosine, and Mean Squared Difference, and varying neighborhood sizes. Also, we investigate how these parameters influence their recommendation accuracy performances. Experiments conducted on two benchmark datasets demonstrate that as the neighborhood size decreases, the kNN algorithms generally become more vulnerable to blockbuster bias while their accuracy increases. One explanation for this finding is that more users (for the UserKNN) or items (for the ItemKNN) contribute to the computed prediction with larger neighborhoods, achieving more diverse and thus less blockbuster biased recommendations. Also, using the Cosine metric for the kNN algorithms is superior to other similarity functions in producing recommendations where blockbuster bias is treated more; however, it leads to having unqualified recommendations in terms of predictive accuracy as they are usually conflicting goals. On the other hand, we found that the Pearson usually performs better than other functions regarding recommendation accuracy, which is also parallel with the outcomes in previous related studies [11]. In conclusion, our analysis provides inference on how parameter-tuning of the kNN algorithms should be performed according to the purposes of the service providers.

Improving beyond-accuracy quality while maintaining accuracy is essential for recommender systems in several aspects. Therefore, our future direction is to develop a mitigation strategy modifying the prediction calculation step of the kNN algorithms by including a tuning parameter penalizing blockbuster items to alleviate the effects of such blockbuster bias on recommendation quality.

Acknowledgments

This study is supported by Project No. M-2021-811 from the Sivas Cumhuriyet University.

References

[1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*, Springer, pp. 1–35, 2011.

[2] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, vol. 52, no. 1–37, 2019, doi: 10.1007/s10462-018-9654-y.

[3] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004, doi: https://doi.org/10.1145/963770.963772.

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

[5] R. Chen, Q. Hua, Y.-S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018, [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2877208.

[6] M. Karimi, D. Jannach, and M. Jugovac, "News recommender systems – Survey and roads ahead," *Inf. Process. \& Manag.*, vol. 54, no. 6, pp. 1203–1227, 2018, [Online]. Available: https://doi.org/10.1016/j.ipm.2018.04.008.

[7] M. Nilashi, O. Bin Ibrahim, and N. Ithnin, "Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and Neuro-Fuzzy system," *Knowledge-Based Syst.*, 2014, doi: 10.1016/j.knosys.2014.01.006.

[8] J. L. Sánchez, F. Serradilla, E. Martínez, and J. Bobadilla, "Choice of metrics used in collaborative filtering and their impact on recommender systems," 2008, doi: 10.1109/DEST.2008.4635147.

[9] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. from Data*, vol. 4, no. 1, pp. 1–24, 2010, [Online]. Available: https://doi.org/10.1145/1644873.1644874.

[10] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Syst.*, vol. 23, no. 6, pp. 520–528, 2010, [Online]. Available: https://doi.org/10.1016/j.knosys.2010.03.009.

[11] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr. Boston.*, 2002, doi: 10.1023/A:1020443909834.

[12] E. Yalcin, "Blockbuster: A New Perspective on Popularity-bias in Recommender Systems," in *6th International Conference on Computer Science and Engineering (UBMK)*, Sep. 2021, pp. 107–112, doi: 10.1109/UBMK52708.2021.9558877.

[13] E. Yalcin and A. Bilge, "Treating adverse effects of blockbuster bias on beyond-accuracy quality of personalized recommendations," *Eng. Sci. Technol. an Int. J.*, vol. 33, p. 101083, Sep. 2022, doi: 10.1016/J.JESTCH.2021.101083.

[14] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and Debias in Recommender System: A Survey and Future Directions," *arXiv Prepr. arXiv2010.03240*, 2020.

[15] L. Boratto, G. Fenu, and M. Marras, "Connecting user and item perspectives in popularity debiasing for collaborative recommendation," *Inf. Process. \& Manag.*, vol. 58, no. 1, p. 102387, 2021, [Online]. Available: https://doi.org/10.1016/j.ipm.2020.102387.

[16] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay, "Accurately Interpreting Clickthrough Data as Implicit Feedback," *ACM SIGIR Forum*, vol. 51, no. 1, pp. 4–11, Aug. 2017, doi: 10.1145/3130332.3130334.

[17] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *International Conference on Machine Learning*, 2014, pp. 1512–1520.

[18] S. Krishnan, J. Patel, M. J. Franklin, and K. Goldberg, "A methodology for learning, analyzing, and mitigating social influence bias in recommender systems," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 137–144, doi: https://doi.org/10.1145/2645710.2645740.

[19] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac, "What recommenders recommend: an analysis of recommendation biases and possible countermeasures," *User Model. User-adapt. Interact.*, vol. 25, no. 5, pp. 427–491, Dec. 2015, doi: 10.1007/S11257-015-9165-3.

[20] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," *Proc. 32nd Int. Florida Artif. Intell. Res. Soc. Conf. FLAIRS 2019*, pp. 413–418, 2019.

[21] E. Yalcin and A. Bilge, "Investigating and counteracting popularity bias in group recommendations", *Inf. Process. Manag.*, vol. 58, no. 5, Sep. 2021, doi: 10.1016/j.ipm.2021.102608.

[22] D. Kowald, M. Schedl, and E. Lex, "The unfairness of popularity bias in music recommendation: A reproducibility study," in *European Conference on Information Retrieval*, 2020, pp. 35–42, [Online]. Available: https://doi.org/10.1007/978-3-030-45442-5_5.

[23] H. Abdollahpouri, M. Mansoury, R. Burke, and B. Mobasher, "The unfairness of popularity bias in recommendation", *arXiv preprint arXiv:1907.13286*, 2019.

[24] L. Boratto, G. Fenu, and M. Marras, "The effect of algorithmic bias on recommender systems for massive open online courses," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11437 LNCS, pp. 457–472, 2019, doi: 10.1007/978-3-030-15712-8_30.

[25] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Missing data modeling with user activity and item popularity in recommendation", in *Asia Information Retrieval Symposium*, 2018, pp. 113–125, [Online]. Available: https://doi.org/10.1007/978-3-030-03520-4_11.

[26] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Correcting Popularity Bias by Enhancing Recommendation Neutrality", in *RecSys Posters*, 2014.

[27] H. Abdollahpouri, R. Burke, and B. Mobasher, "Popularity-Aware Item Weighting for Long-Tail Recommendation", *arXiv preprint arXiv:1802.05382*, 2018.

[28] G. Adomavicius and Y. Kwon, "Multi-criteria recommender systems," in *Recommender Systems Handbook, Second Edition*, 2015.

[29] N. A. Najjar and D. C. Wilson, "Differential neighborhood selection in memory-based group recommender systems," in *Twenty-Seventh International Flairs Conference*, 2014.

[30] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, Jan. 2010, doi: 10.1145/1644873.1644874.

[31] K. Choi and Y. Suh, "A new similarity function for selecting neighbors for each target item in collaborative filtering," *Knowledge-Based Syst.*, 2013, doi: 10.1016/j.knosys.2012.07.019.

[32] R. Sanders, "The Pareto principle: its use and abuse," *J. Serv. Mark.*, 1987.

[33] L. Baltrunas and F. Ricci, "Group Recommendations with Rank Aggregation and," *Proc. fourth ACM Conf. Recomm. Syst. ACM*, 2010.