Engineering Science and Technology, an International Journal 33 (2022) 101083

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Treating adverse effects of blockbuster bias on beyond-accuracy quality of personalized recommendations

Emre Yalcin [a,*], Alper Bilge [b]

[a] Computer Engineering Department, Sivas Cumhuriyet University, 58140 Sivas, Turkey
[b] Computer Engineering Department, Akdeniz University, 07058 Antalya, Turkey

ARTICLE INFO

Article history:
Received 19 July 2021
Revised 29 October 2021
Accepted 28 November 2021
Available online 23 December 2021

Keywords:
Recommender systems
Collaborative filtering
Popularity bias
Blockbuster items
Beyond-accuracy metrics

ABSTRACT

Collaborative filtering recommendation algorithms are vulnerable against the popularity bias, including the most popular items repeatedly into the produced ranked lists. However, the research on popularity bias focuses solely on the number of times items are rated rather than the magnitude of the provided ratings when scrutinizing the adverse effects of such bias. This paper introduces a metric describing the blockbuster items that are popular and highly rated simultaneously and investigates the potential biases of collaborative recommendation algorithms towards such items comprehensively. Then, we develop an algorithmic post-processing debiasing approach for potential blockbuster bias in recommendations. Specifically, this method aims to penalize blockbuster items in produced ranked lists by re-sorting items based on the artificial ranking scores, estimated by considering both the blockbuster degree of the items and the generated predictions for them simultaneously. The experiments conducted on three benchmark real-world datasets demonstrate that four prominent collaborative filtering algorithms lead to an undesirable bias in their recommendations towards blockbuster items. The empirical outcomes also indicate that our mitigation method helps treat the adverse effects of the blockbuster bias in terms of beyond-accuracy evaluations such as catalog coverage, diversity, and novelty, with negligible losses in ranking accuracy.

1. Introduction

Recommender systems are artificial intelligence solutions helping the decision-making process of individuals by recommending untasted but engaging content while filtering out irrelevant ones [1]. With increasing Internet usage in recent years, these systems have become more and more attractive for many digital platforms in different application domains, such as music (e.g., Spotify, Last.FM), movie (e.g., Netflix, Amazon Prime), e-commerce (e.g., eBay, Amazon), accommodation (e.g., Booking.com, Trivago), and news (e.g., Google News), due to their significant contributions to the attraction and usability of the platforms [2].

Recommender systems usually aim to assist users by producing a ranked list of desirable items (i.e., services or products) or predicting a rating score for items not yet experienced by them. These recommendation tasks are generally performed utilizing collaborative filtering (CF) algorithms, content- or knowledge-based filtering methods, or hybrid approaches [3]. CF algorithms are regarded as the most prevalent among them due to their high performance and efficiency. By following the assumption that individuals having similar tastes in the past are also incline to agree in the future, these algorithms mainly try to predict future choices of users about items by operating a user-item matrix reflecting their past preferences [4].

In the literature, the success of CF algorithms is generally evaluated with the accuracy level of the recommendation lists they have produced for individuals. However, in recent years, the beyond-accuracy aspects of recommended items, such as how much they cover the catalog and how much they are novel or diverse, have taken a step forward in evaluating recommendation quality [5–7]. Nevertheless, research on recommender systems has acknowledged that CF algorithms are commonly biased towards certain items due to their mechanisms to generate recommendations or the characteristics of rating data on which they are trained [8]. This phenomenon, unfortunately, ends up with having recommendation lists that are unqualified in terms of such beyond-accuracy perspectives.

The most well-known kind of such biases is called the popularity bias, which refers to the intrinsic leaning of CF algorithms to

* Corresponding author.
E-mail addresses: eyalcin@cumhuriyet.edu.tr (E. Yalcin), abilge@akdeniz.edu.tr (A. Bilge).

Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2021.101083
2215-0986/

feature a few popular items over the unpopular ones (i.e., niche) in their recommended lists [9,10]. Indeed, this phenomenon is usually caused by nonuniformly distributed individual preferences, i.e., a small fraction of items are evaluated by the majority of users while the remaining niche ones receive only a few ratings. This fact leads to having ranked lists where popular items gain more visibility in recommendation lists while niche ones cannot get the deserved attention, even if they would be of interest. Besides, the awareness of such a bias of the recommendation algorithms makes the system vulnerable to fake reviews or social bots in order to increase the number of ratings given to particular products to get more visibility in produced recommendations. Therefore, exploring the popularity bias issues in recommendations and developing efficient popularity-debiasing methods to treat its adverse impacts on recommendation quality has been getting more and more attention in recent years [11,12,9].

However, the research on popularity bias commonly focuses on the most frequently rated items by disregarding the magnitude of the given ratings (i.e., liking-degree of the items). Nevertheless, the popularity of a product does not always mean that individuals strongly desire it, or vice versa. For example, Mortal Kombat is one of the most discussed movies of 2021 based on the number of provided ratings in the Metacritic[1], which is a well-known website aggregating user reviews for contents in several different types. However, the average of the ratings provided for Mortal Kombat is 6.3 out of 10, which demonstrates that users are not so much into this movie despite being a huge hit. Such a trade-off between popularity and liking-degree can also be seen for many products in different application domains such as books, games, and music [13].

Therefore, we consider the blockbuster items defined as both popular and highly rated by individuals and investigate whether collaborative recommendation algorithms lead to an undesirable bias in favor of such items in their generated ranked lists. Moreover, we develop a post-processing debiasing approach to counteract the potential blockbuster bias and improve recommendation quality. The following summarizes the main contributions of this study.

1. We present a practical formulation defining the blockbuster level of the items by properly combining their popularity and liking-degree. We also show the undesired imbalances towards blockbuster items in three prominent datasets in the movie and social media domains by using the presented formulation.
2. We introduce an efficient evaluation metric that helps to measure the level of potential biases to blockbuster items in ranking-based recommendations. The conducted experiments based on this metric conclude that four prominent CF algorithms in two different families are strongly biased towards blockbuster items in their produced recommendations.
3. We propose a post-processing blockbuster-debiasing method that combines the items' blockbuster level and the predicted ratings by a proper recommendation algorithm to re-rank the recommendation lists by penalizing blockbuster items. The proposed method helps to treat the adverse effects of the observed blockbuster bias and thus improve the beyond-accuracy quality of the recommendations in terms of catalog coverage, diversity, and novelty, with negligible losses in ranking accuracy.

The rest of the study is organized as follows. The following section summarizes the literature on popularity bias and 3 presents the proposed novel formulation to describe blockbuster items. 4 gives detailed information about the developed metric measuring the level of potential blockbuster bias in recommendations. 5 presents the developed blockbuster-debiasing method, and the following section includes the performed experimental studies. 7 discusses the present study and gives some limitations about the blockbuster bias in recommendations. Finally, 8 concludes the study and gives detailed information about future research directions.

2. Related Work

In general, the main goal of recommender systems is to enhance users' satisfaction by filtering out and recommending appropriate items/products. Although the worth of such suggested content is usually evaluated with how they are accurate, the literature has recently introduced different evaluation concepts that measure users' interaction experience with the system from different perspectives and thus assessed the quality of the produced recommendations more deeply [7]. The most well-known types of such beyond-accuracy perspectives the literature has recently been concerned with and studied extensively are coverage, novelty, and diversity. More specifically, coverage is defined as the ratio of items recommended to users to the whole item catalog [7]. Novelty is another critical aspect that refers to the system's ability to recommend novel items that are never consumed or known by individuals in the past [6]. Finally, diversity indicates the variety of the items in the produced recommendation lists [5].

However, the literature has recently discovered many bias types of the recommendation algorithms, such as conformity [14], exposure [15], selection [16], position [17], and popularity [10,18,12], deteriorating the recommendation quality in terms of such beyond-accuracy aspects and challenging to achieve qualified recommendations. Among such bias types, popularity bias is the most prominent due to its highly adverse effects on beyond-accuracy recommendation quality. Therefore, many studies focused on discovering the level of bias caused by CF algorithms and scrutinizing how this bias affects the quality of produced recommendations for different application areas such as music [19], movies [20], online education [10], and tourism activities [21].

Also, various studies have attempted to develop practical treatment methods for dealing with popularity bias problems in recommendations. Such proposed popularity-debiasing approaches are also commonly divided into three main categories, *pre-*, *in-*, and *post-processing*, based on how they have participated in the recommendation process [9]. Since one of the main reasons for popularity bias is the imbalances in rating distribution, *pre-processing* approaches usually aim to reduce such inequalities by altering data on which CF algorithms are trained. For example, [22] introduces a method that initially split the item set into two different classes as *head* and *tail*; the *head* class consists of popular items having a significantly larger amount of ratings than those in the *tail* class. Then, recommendations for *tail* items are produced using ratings in the *tail* class only, while those for *head* items are estimated using all data. Also, [23] presents a practical popularity debiasing technique that first creates synthetic user-item tuples where the observed items are mostly unpopular and then utilizes them to train algorithms. Finally, [24] treats popularity bias by utilizing a probability distribution function based on item popularity.

On the other hand, *in-processing* approaches aim to modify the internal mechanisms of recommendation algorithms to consider both popularity and relevance simultaneously. This task is usually accomplished using specific constraints or conducting a joint optimization. For example, [25] proposes an optimized variant of the well-known RankALS algorithm, which contributes to producing recommendation lists where predictive accuracy and intra-list item diversity are balanced. Also, [11] presents a framework that first constructs the neighborhoods between the items based on

[1] https://www.metacritic.com/movie/mortal-kombat-2021

their popularity instead of the magnitude of their ratings; then, it eliminates some most popular ones to have a more balanced common-neighbor similarity index. Lastly, [9] introduces a technique that minimizes the correlation between item popularity and user-item relevance for treating items in the long tail fairly.

Lastly, *post-processing* techniques usually aim to re-rank a recommendation list that has already been generated or create a new one following a specific constraint. These are the most utilized approaches for mitigating popularity bias since they can be easily applied to the output of any recommendation algorithm, which is also why we focus on developing a post-processing method to counteract potential blockbuster bias in this study. For example, [26] introduces an approach that first calculates weight scores for items based on their popularity and then utilizes them to punish popular items during re-ranking recommended item lists. Likewise, [12] follows a similar strategy and presents two robust popularity debiasing methods for recommending to groups of users rather than individuals. Also, [27] presents the xQuad algorithm that is an enhanced re-ranking approach and helps balance the trade-off between long-tail item coverage and ranking accuracy more robustly. .

As seen from the presented literature, many recent studies consider biases towards popular items in the recommendations and try to treat this issue for achieving more qualified recommendations. However, as emphasized in the previous section, the popularity of an item does not always mean that individuals strongly desire it. Therefore, more comprehensive analyses are required to investigate potential bias towards blockbuster items and develop novel practical methods to mitigate its adverse effects on recommendations.

3. The *Blockbuster Level Estimation Procedure (BLEP)*

In the domain of recommender systems, the blockbuster term usually stands for the most favored items [28,29]; however, we utilize this term in this study to denote items having a high average rating and rated by a significant number of users. Here, one challenge is determining if an item is blockbuster or not by considering the properties of the ratings it has received. To accomplish this task, we first develop a practical method, namely *Blockbuster Level Estimation Procedure (BLEP)*, that measures the blockbuster degrees of the items by properly incorporating their popularity and liking degree.

Suppose that $r_{u,i}$ denotes the value of the rating provided by user u for the item i, the developed *BLEP* first calculates the popularity of the item i, i.e., p_i, by simply counting the number of users who rated the item. Then, it estimates the average rating of the item i, as $\mu_i = \frac{\sum_{u \in U_i} r_{u,i}}{|U_i|}$, where U_i indicates the set of users who have provided a rating for the item.

Here, the maximum possible value for $\mu_i$corresponds to the highest rating in the utilized rating scale, which can occur when all of the provided ratings for the item i are equal to the highest rating. On the other hand, the maximum possible value of p_i is the number of all available users, which can be observed when all users provide a rating. Accordingly, the calculated p_i values vary a broader range than μ_i ones, which calls for a normalization process for both p_i and μ_i for making them comparable before performing a combination procedure. For this purpose, the *BLEP* first transforms both the computed p_i and μ_i values into [0,1] range utilizing min–max normalization, and then combines by simply averaging them to achieve $\mathbb{B}_i$ scores that describe the blockbuster levels of the items, utilizing the formula given in Eq. 1.

$$\mathbb{B}_i = (\tilde{\mu}_i + \tilde{p}_i)/2 \tag{1}$$

where $\tilde{\mu}_i$ and $\tilde{p}_i$ denote the normalized average rating and popularity values for the item i, respectively. Here, we consider the popularity and the average rating of the items both necessary conditions that are equally important for an item to be considered as blockbuster. Therefore, we try to balance these two properties when estimating blockbuster levels by following the arithmetic mean strategy. This is because we attempt to penalize either unpopular or not-liked items when calculating blockbuster scores.

4. The *Blockbuster Recommendation Frequency (BRF)* evaluation metric

To investigate the potential biases towards blockbuster items properly in ranking-based recommendations, we present an efficient evaluation metric, named the *Blockbuster Recommendation Frequency (BRF)* [30], which measures the degree of blockbuster bias induced by a recommendation algorithm.

Before introducing details of this metric, we first attempt to categorize the items based on their blockbuster scores ($\mathbb{B}$) calculated using the *BLEP*. To this end, we first sort items in the whole catalog in descending order by their computed $\mathbb{B}$ scores. Then, we define two distinct item classes, namely *head* and *tail*, by following the well-known Pareto principle [31], i.e., the items having 20% of all provided ratings are classified as *head*, and the remaining items in the catalog are labeled as *tail* ones. Thus, *head* items refer to the most blockbuster items, while *tail* ones indicate the relatively underappreciated items.

After that, the proposed *BRF* evaluation metric estimates the blockbuster bias performance of an algorithm by computing how often the items in the *head*, i.e., the set of items having the highest blockbuster scores, are recommended to users as in the following. It first counts how many times all *head* items appeared in the produced recommendation lists and then computes its proportion to all recommended items. Thus, the higher *BRF* values mean more substantial bias in favor of the blockbuster items and, therefore, having recommendations that are more unqualified in terms of diversification.

Suppose that $\mathbb{N}$ represents the multiset of the recommendations constructed by stacking the top-*N* recommendation lists produced for each user through the utilized recommendation algorithm. The *BRF* value of the algorithm is calculated using the formula given in Eq. 2.

$$BRF = \frac{\sum_{i \in \mathbb{N}} \mathbb{1}(i \in \mathbb{H})}{|\mathbb{N}|} \tag{2}$$

where H is the set of *head* items in the whole catalog. For example, suppose that $\{i_1, i_2, \cdots, i_{10}\}$ is the whole item set in the catalog, and i_3 and i_5 are the head items among them. Also, assume that there are three available users in the system and the top-3 recommendation lists produced with any algorithm for them are $\{i_3, i_2, i_5\}$, $\{i_5, i_1, i_6\}$, and $\{i_5, i_3, i_9\}$. In such a scenario, the multiset of the recommendations constructed by stacking the individual recommendation lists, i.e., $\mathbb{N}$, ends up with $\{i_3, i_2, i_5, i_5, i_1, i_6, i_5, i_3, i_9\}$. Accordingly, the *BRF* performance of the utilized algorithm is calculated as the ratio of how many times the head items (i.e., i_3 and i_5) appear in the $\mathbb{N}$ to the size of $\mathbb{N}$, which is equal to $5/9 = 0.55$. This ratio concludes that almost more than half of all recommended items are the *head* items and shows a solid and undesirable blockbuster bias in such a scenario.

5. The proposed *Blockbuster-Debiasing Procedure (BDP)*

As emphasized in the Related Work section, one of the most common practices to mitigate popularity bias in recommendations

is to utilize post-processing approaches, as they can be easily applied to the results of any recommendation algorithm. Therefore, this section introduces an effective treatment technique inspired by this practice, named the *Blockbuster-Debiasing Procedure (BDP)*, which aims at re-ranking the recommended items to counteract potential bias issues towards blockbuster items.

The main goal of the proposed *BDP* is to feature underappreciated items by penalizing blockbuster ones based on their blockbuster level while performing the recommendation process. To this end, it follows a weighting strategy that assigns higher weights to *tail* items while gives relatively lesser weights to *head* ones to counterbalance the potential bias caused by their blockbuster level. More formally, the *BDP* estimates such a weight for each item w_i using the formula given in Eq. 3.

$$w_i = \frac{1}{1 + \mathbb{B}_i} \tag{3}$$

where $\mathbb{B}_i$ denotes the blockbuster level of the item i, which can be calculated as in Eq. 1.

After the weight value for each item are computed, the *BDP* predicts a rating $p_{u,i}$ for user u on item i via any conventional CF algorithm such as UserKNN, ItemKNN, or SVD. Then, it calculates an artificial ranking score $R_{u,i}$ for each user-item pair by incorporating the predicted rating and estimated weight value for the corresponding item, as in Eq. 4. Here, $\tilde{p_{u,i}}$ is the normalized predicted ratings in [0,1] range determined using min–max normalization so that they become comparable to the estimated w values. These computed $R_{u,i}$ scores are then utilized to re-sort items for each user to achieve a final recommendation list for them.

$$R_{u,i} = (1 - \alpha)\tilde{p_{u,i}} + \alpha w_i \tag{4}$$

where α denotes the controlling coefficient that adjusts the significance degree of each component in Eq. 4. Accordingly, lesser α values indicate more weight for the originally predicted ratings, maintaining ranking accuracy; on the other hand, higher α values penalize the *head* items more, shifting the balance in the recommendation lists to the *tail* ones.

5.1. An illustrative example for the proposed BDP

This section provides a toy example to clarify how the proposed *BDP* computes the ultimate ranking scores (R) for items to be used for producing recommendation lists. To this end, we first present a sample rating matrix in 1 containing the preferences of four users on six items in {1,5} rating scale. In this example, we also give the predicted ratings for each user-item pair in parenthesis, which are estimated using a random predictor method. Note that $\perp$ represents the unrated items by the individuals in the 1.

Based on the original ratings of individuals given in 1, we present each step of the *BDP* with their explanations in 2. Besides, according to the estimated weight values w presented in the bottom row of 2 and the predicted ratings given in parenthesis in 1, the computed final artificial ranking scores (R) via the *BDP* (see Eq. 4) for each user-item pair are shown in 3. Note also that we set the value of α as 0.25 when estimating such ranking scores in this toy example.

As can be followed by the presented example, our *BDP* can produce recommendation lists where the sorting of the items would highly differ when compared to the ranked lists constructed based on only predicted ratings. For example, the most recommendable item for user u_1 is the i_5 based on the original predictions given in parenthesis in 1; however, it is the item having the highest $\mathbb{B}$ score among others, as can be followed by 2. On the other hand, our proposed *BDP* recommends item i_2 at the top of its produced ranked list for the corresponding user by penalizing the most blockbuster item i_5 and considering ranking accuracy, which in turn helps to mitigate potential bias towards blockbuster items in the generated recommendations.

6. Experimental studies

This section includes a broad set of experiments on two real-world datasets to potential blockbuster bias in recommendations and to evaluate the performance of the proposed *BDP*.

6.1. Datasets and experimentation methodology

In the experiments, we utilize three real-world benchmark datasets in different application domains; MovieLens-1 M (ML) [32], Personality2018 (Per) [33], and Ciao [12]. More specifically, the ML and Per consist of user preferences on movies, while the Ciao is crawled from a real-world social platform and includes user ratings on various products. Also, the provided ratings are on a five-star rating scale in ML and Ciao datasets, while on a [0.5, 5] scale in Per dataset. Additionally, since the original Ciao and Per are highly sparse and too large datasets, we employ subsets of the original collections, called Ciao20 and Per20, where each user and item has at least 20 ratings. Detailed information about these datasets is presented in 4.

To produce ranking-based recommendations for users, we follow the leave-one-out cross-validation experimentation methodology [34]. Accordingly, we select one active user as the test set and employ the rest of the users as the train set; then, we produce predictions for the actual ratings of the active user applying a CF algorithm on the train set. This procedure is repeatedly conducted for each user in the dataset. At the final step, we sort the items based on the generated predictions for each user in descending order and select Top-N items as a recommendation list, where N is set as 10.

6.2. The recommendation algorithms

This study employs four prevalent CF algorithms and examines their produced recommendations in terms of the potential blockbuster bias issues. Two of them are famous neighborhood-based methods, namely *UserKNN* and *ItemKNN*, which employ the most similar users or items and consider the computed baseline preferences to produce predictions [35]. Also, when applying these algorithms, we use the cosine metric to determine similarities between users or items and set the neighborhood size as 40, as such a configuration is one of the best settings of these algorithms in achieving accurate predictions [36].

The other two algorithms are the matrix factorization-based *SVD* [37] and *SVD++* [35], which are highly effective on accuracy and scalability performance and therefore widely adopted as a basis of many modern recommendation frameworks. In principle, these algorithms characterize both items and users with factor vectors extracted from the patterns of the provided ratings and produce recommendations based on the high correspondence of such constructed factors. More specifically, the utilized SVD algorithm is an enhanced version of the conventional matrix factorization-based CF approach, which also considers variations in rating values associated with either user or items (i.e., biases), also known as the *Biased Matrix Factorization* CF algorithm. On the other hand, the SVD++ is an extension of SVD taking into account not only explicit ratings but also implicit ones. Note that all of the used CF algorithms are applied via Surprise[2], introduced as a Python framework for recommender systems.

[2] https://surpriselib.com/

Table 1
A user-item matrix to exemplify the *BDP*.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	3 (2.45)	2 (3.01)	$\perp$ (2.97)	1 (2.05)	$\perp$ (3.03)	1 (1.70)
u_2	4 (3.45)	$\perp$ (2.23)	$\perp$ (3.65)	2 (1.28)	5 (4.80)	3 (2.12)
u_3	$\perp$ (3.35)	4 (3.67)	$\perp$ (3.25)	4 (3.36)	$\perp$ (1.78)	5 (4.88)
u_4	1 (1.45)	$\perp$ (2.23)	3 (3.55)	$\perp$ (4.55)	3 (3.32)	2 (2.22)

Table 2
Each step of BDP with their explanations.

Step	i_1	i_2	i_3	i_4	i_5	i_6	Explanations
p	3.00	2.00	1.00	3.00	2.00	4.00	Item popularity
μ	2.67	3.00	3.00	2.33	4.00	2.75	The average rating
$\tilde{p}$	0.67	0.33	0.00	0.67	0.33	1.00	Normalized item popularity
$\tilde{\mu_i}$	0.20	0.40	0.40	0.00	1.00	0.25	Normalized average rating
$\mathbb{B}$	0.43	0.36	0.20	0.33	0.66	0.62	Blockbuster level, using Eq. 1
w_i	0.69	0.73	0.83	0.74	0.60	0.61	Weight score, using Eq. 3

Table 3
The calculated final R ranking scores with the *BDP*.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	2.01	2.44	2.43	1.72	2.42	1.42
u_2	2.76	1.85	2.94	1.14	3.75	1.74
u_3	2.68	2.93	2.64	2.70	1.48	3.81
u_4	1.26	1.85	2.87	3.59	2.64	1.81

Table 4
Detailed information about the utilized datasets.

Dataset	#Users	#Items	#Ratings	Density(%)	Rating Scale
ML	6,040	3,952	1,000,209	4.25	{1, 2, ⋯, 5}
Ciao20	3,278	1,351	52,717	1.19	{1, 2, ⋯, 5}
Per20	1,780	7,228	911,369	7.08	{0.5, 1, 1.5, ⋯, 5}

6.3. Evaluation metrics

To evaluate the performance of the proposed *BDP*, we utilize three different metrics measuring the beyond-accuracy quality of the recommendations: Long-Tail Coverage (LC) [5], Average Percentage of Long-tail items (APL) [20], and Novelty [38].

Concretely, the LC is a prevalent metric in popularity bias research and mainly measures the proportion of tail items observed in the produced recommendation lists of all users. Therefore, this metric can be considered a variant of the well-known *aggregate diversity* metric [20], applying only to the items in the long-tail partition of the whole catalog.

Suppose that N_u is the ranked list of items recommended to user u and T is the set of *tail* items, the LC performance of the utilized recommendation algorithm is computed using the formula given in Eq. 5.

$$LC = \frac{|\bigcup_{u \in U}(N_u \cap T)|}{|T|} \tag{5}$$

where U denotes the set of all available users and $\bigcup$ indicates the union set operator.

The APL is another helpful metric in evaluating the recommendation lists' beyond-accuracy quality and computes what percentage of the items recommended to each user belongs to the *tail* partition of the catalog. Accordingly, the overall APL performance of the used recommendation algorithm is calculated using the formula given in Eq. 6.

$$APL = \frac{1}{|U|}\sum_{u \in U}\frac{|N_u \cap T|}{|N_u|} \tag{6}$$

The last utilized metric is the Novelty which estimates the recommendation algorithm's ability to suggest unknown, i.e., novel, items in the *tail* set, and is calculated as in Eq. 7. Note also that higher LC, APL, and Novelty values conclude having more qualified ranked lists in terms of beyond-accuracy aspects.

$$Novelty = \frac{1}{|U|}\sum_{u \in U}\frac{|\{i, i \in (T - I_u)\}|}{|N_u|} \tag{7}$$

where I_u denotes the list of rated items in the profile of user u.

To evaluate how the proposed *BDP* influences the accuracy of the produced recommendation lists, we also employ the F1-score and normalized Discounted Cumulative Gain (nDCG) metrics. More specifically, F1-score is calculated as the harmonic mean of Precision and Recall [39]. Precision refers to the ratio of the number of recommended appropriate items to the total number of recommended items, while Recall demonstrates the ratio of the number recommended appropriate items to all rated items in the user profile. Here, we assign 3.5 as the threshold value in estimating if a recommended item is appropriate for users, as positive votes intuitively correspond to 4 and 5 for a five-star rating scale [40].

Suppose that top-$N = \{i_1, i_2, \cdots, i_N\}$ indicates the ranked list produced for user u using a CF algorithm, the F1-score performance of the algorithm is computed using the formula given in Eq. 8.

$$F1@N = \frac{2 \times P@N \times R@N}{P@N + R@N} \tag{8}$$

where *P@N* and *R@N* denote the estimated Precision and Recall values of the produced top-*N* recommendation list, respectively. The *n*DCG metric, on the other hand, mainly measures the level of quality of the recommended items by considering their actual votes as well as their positions in the produced recommendation list. Concretely, the Discounted Cumulative Gain (DCG) and *n*DCG of a estimated top-*N* list for user u are calculated using the formula presented in Eqs. 9 and 10, respectively.

$$\mathrm{DCG}_N^u = r_{u,i_1} + \sum_{n=2}^{N} \frac{r_{u,i_n}}{log_2(n)} \tag{9}$$

$$n\mathrm{DCG}_N^u = \frac{\mathrm{DCG}_N^u}{\mathrm{IDCG}_N^u} \tag{10}$$

where $r_{u,i}$ denotes the actual vote of user u on item i and the $IDCG_N^u$ indicates the maximum amount of possible gain for user u, which can be computed with re-ranking of N items so that it will be the ideal order for u. Note that the higher F1-scores and *n*DCG values, the more successful the produced top-*N* lists are.

6.4. Benchmark methods

In the literature, there is no existing research concerned with blockbuster bias in recommendations, and therefore any blockbuster-debiasing method for comparison has not been introduced yet. Nevertheless, we select three efficient popularity-debiasing strategies as the benchmark methods to evaluate the performance of our proposed *BDP* method more comprehensively.

The first benchmark popularity-debiasing approach is called *Value-Aware Re-ranking (VaR)* [26], which is a post-processing method re-ranking items by following a weighting strategy. More specifically, the *VaR* method gives higher weights for unpopular items while assigns lesser weights to popular ones for treating bias imposed by item popularity. Then, it re-sorts items based on the artificial ranking scores, which are calculated by combining such weights and prediction scores through a convex combination strategy.

The other selected benchmark methods are two different variants of *Enhanced-Re-ranking Procedures (ERP)* [12], ERP_{Mul} and ERP_{Aug}, which are improved post-processing methods following two different combination strategies. These methods follow a more robust item weighting strategy overcoming some limitations of the *VaR* method. Also, to re-rank items, ERP_{Mul} combines the estimated item weights and prediction scores with a multiplicative strategy, while ERP_{Aug} produces ultimate ranking scores by following an augmentative strategy.

6.5. Experimental results

This section presents the empirical outcomes of the conducted experiments to explore blockbuster bias in recommendations and evaluate the performance of the developed blockbuster-debiasing method.

6.5.1. Imbalances in the distribution of the blockbuster items

As explained in the Introduction section, the observed biases in recommendations usually originate from the imbalances in the users' feedback on items; therefore, we first attempt to explore if the most used datasets are imbalanced towards blockbuster items before going further. To this end, we first calculate the blockbuster score ($\mathbb{B}$) for each item in all utilized datasets using the developed *BLEP*; and then observe the $\mathbb{B}$ score distributions for the ML, Ciao20, and Per20 datasets as depicted in 1a, 1b, and 1c, respectively.

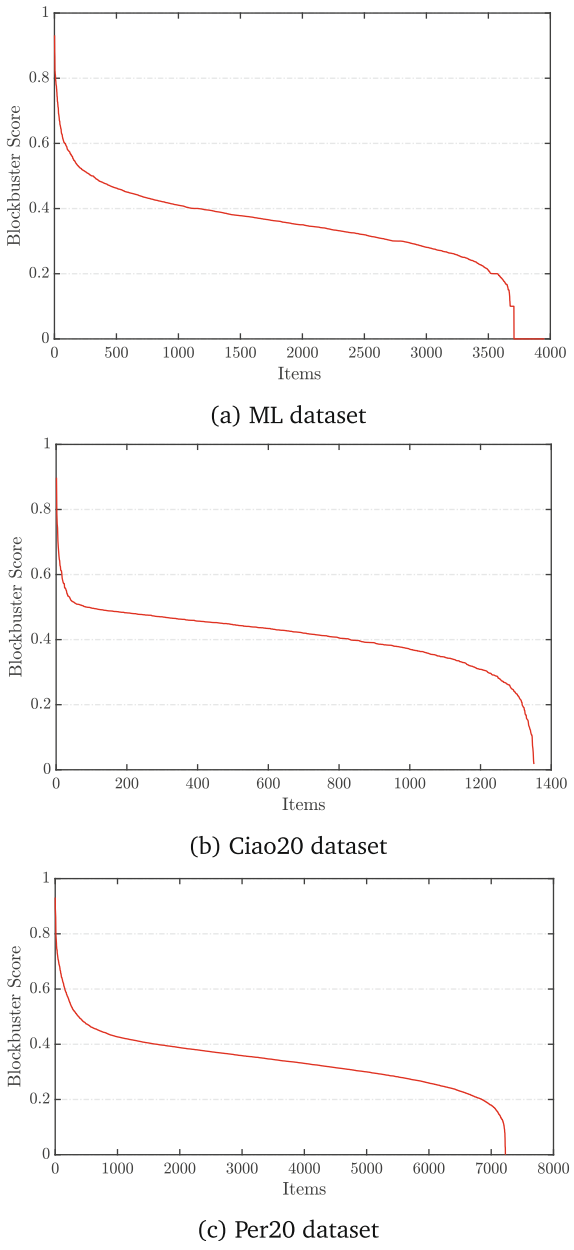

Fig. 1. The distribution of items' blockbuster scores $\mathbb{B}$ for (a) ML, (b) Ciao20, and (c) Per20 datasets. In the x-axis, items are sorted by descending order based on their calculated $\mathbb{B}$ scores..

As shown in 1, a small fraction of the items have significantly higher $\mathbb{B}$ scores than the remaining massive part of the item set for all utilized datasets. This trend is also quite similar to the imbalance towards popular items in data, which, as is known, lead to recommendation algorithms include a few popular items repeatedly in produced lists and create competitional distortions for the items in the long tail. Therefore, our following analysis investigates potential biases towards blockbuster items in recommendations, which such imbalances in blockbuster score distribution may cause.

6.5.2. Analyzing how blockbuster bias dissociate from famous popularity bias

Before going further in the experiments, in this section, we attempt to concretely understand how the observed biases towards blockbuster items and popular items in rating data differ-

entiate. In other words, we try to show why blockbuster bias in datasets is not the same problem as the famous popularity bias phenomenon in practice.

To this end, we first determine two different head item sets, referred as to $head_B$ and $head_P$ for each dataset, by following the Pareto principle explained in 4. The former is constructed by considering items' blockbuster scores estimated with the BLEP, while the latter is formed by considering the frequency of provided ratings for them (i.e., item popularity). Then, we calculate an overlap ratio reflecting what percentage of items in the $head_B$ set also belongs to the $head_P$ set. Thus, the value of overlap ratio calculated for a dataset provides a clear picture of the difference between blockbuster and popularity bias in the original rating data. Accordingly, the lesser overlap ratios mean the items classified as blockbuster are highly different from items labeled as popular, which also concludes that blockbuster bias and popularity bias are two different phenomena in the data. Under these settings, we present the size of constructed $head_B$ and $head_P$ item sets and their overlap ratios for three utilized datasets in 5.

As can be seen from 5, the calculated overlap ratios can vary for each dataset. This finding is caused by the number of available items in the datasets since the determined $head_B$ and $head_P$ sets quite likely become more similar with a more extensive item catalog. In other words, as the number of available items proliferates, the decision-making process of individuals becomes more complicated, and therefore they get more inclined to consume blockbuster or popular items, which in turn leads to similar $head_B$ and $head_P$ item sets.

Also, the obtained overlap ratios conclude that $head_B$ sets usually contain many items not involved in $head_P$. This observation is more apparent for Ciao20 and ML datasets, as almost one-half of Ciao20's $head_B$ and one-quarter of ML's $head_B$ are unpopular items. Therefore, it can be concluded that the problem of blockbuster bias in rating data does not originate from item popularity and is a particularly different issue from the popularity bias problem.

6.5.3. Exploring potential blockbuster bias in recommendations

To measure the potential blockbuster bias of the CF algorithms, we utilize the proposed *BRF* metric, which is explained in detail in 4. The *BRF* outcomes of the four used CF algorithms, i.e., *UserKNN*, *ItemKNN*, *SVD*, and *SVD++*, based on their produced top-10 recommendation lists for ML, Ciao20, and Per20 datasets are depicted in 2.

Based on the obtained *BRF* results, it can be concluded that the utilized CF algorithms produce ranked lists where the *head* items are recommended too frequently for all datasets, meaning that an undesirable and strong bias towards blockbuster items in recommendations is observed. For example, the *BRF* values for the ML dataset vary around 0.55 for each algorithm, which means that about 55% of all recommended items are the *head* items. Also, the highest *BRF* outcomes for the *SVD* and *SVD++* algorithms are observed when the MLM dataset is employed. On the other hand, the lowest *BRF* results for them are achieved with the Per20 dataset. This fact may be caused by the number of available users in the datasets since more users provide a more extensive multiset of the recommendation lists (N) when calculating *BRF* performances of the algorithms (see Eq. 2). Thus, it leads to having recommendation lists where head items appear more and increases *BRF* outcomes of the algorithms. However, *UserKNN* and *ItemKNN* algorithms do not follow such a trend, as shown in 2. Therefore, it can be concluded that the blockbuster bias of the matrix factorization-based CF algorithms depends on the number of available users in the utilized dataset, while those of neighborhood-based ones are not influenced by such a property of the dataset.

Table 5
The calculated overlap ratios for the utilized datasets.

Dataset	$\lvert head_B \rvert$	$\lvert head_P \rvert$	*Overlap Ratio*
ML	118	113	0.77
Ciao20	53	39	0.52
Per20	223	219	0.88

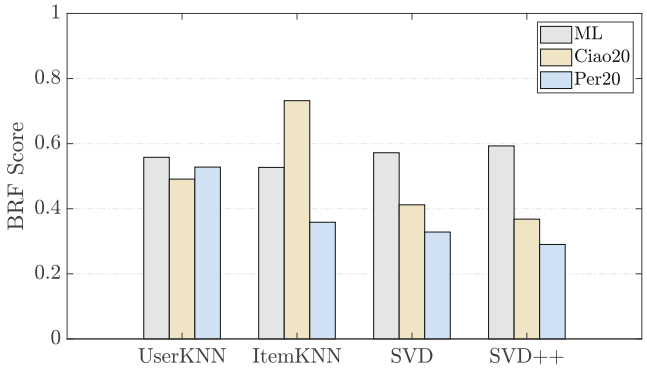

Fig. 2. The BRF results of the utilized CF algorithms for ML, Ciao20, and Per20 datasets.

The obtained *BRF* results also demonstrate that the neighborhood-based methods (i.e., *UserKNN* and *ItemKNN*) are slightly more robust than others against blockbuster bias for the ML dataset, while the matrix factorization-based methods (i.e., *SVD* and *SVD++*) are usually the most successful ones for both Ciao20 and Per20 datasets. In conclusion, these observations demonstrate that the blockbuster bias of the recommendation algorithms can significantly vary based on the characteristics of the dataset where they have trained.

6.5.4. Evaluation of the beyond-accuracy performance of the proposed BDP

To scrutinize the performance of the proposed *BDP*, we also perform a broad set of additional experiments where we observed how the *BDP* improves recommendation quality in terms of the beyond-accuracy perspectives. When applying the *BDP* in these experiments, we also consider varying α values from 0 to 0.5 to see how it affects the recommendation quality. Here, lesser α values mean more weight for originally produced predictions while higher α values penalize the blockbuster items more, as can be followed by Eq. 4. Under these settings, we present the LC, APL, and Novelty results of the *BDP* for four utilized CF algorithms, i.e., *UserKNN*, *ItemKNN*, *SVD*, and *SVD++*, on the ML, Ciao20, and Per20 datasets in 3, 4, and 5, respectively.

As can be seen from 3, 4, and 5, the proposed *BDP* helps to enhance LC, APL, and Novelty performances of the utilized CF algorithms by alleviating their biases towards blockbuster items in their generated recommendation lists. We also perform statistical significance tests to explore whether the obtained improvements are significant or not. Accordingly, all achieved LC, APL, and Novelty enhancements on three datasets are statistically significant for all utilized algorithms at 95% confidence level, except that in the LC observed when α is set to 0.5 on the ML dataset for *UserKNN* algorithm.

The empirical results also conclude that the improvements of our *BDP*, especially for APL and Novelty, are more apparent for the ML when compared to Ciao20 and Per20 datasets. This observation may be because the original versions of the utilized CF algorithms, i.e., without empowering with the proposed *BDP*, which is

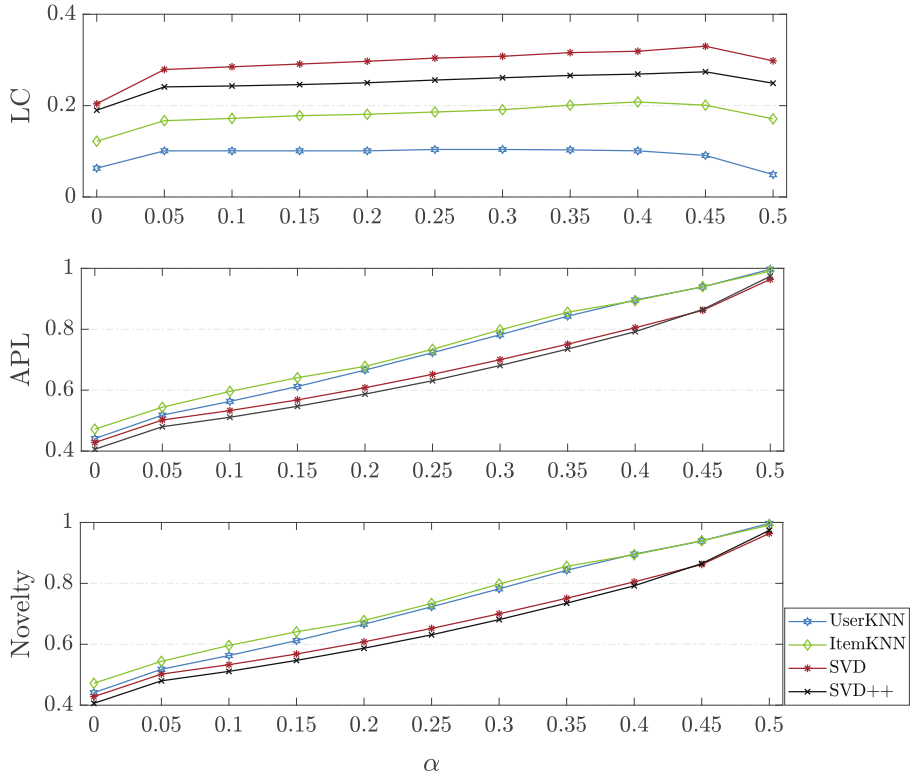

Fig. 3. The beyond-accuracy performance, (i.e., LC, APL, and Novelty) of the proposed *BDP* for the ML.

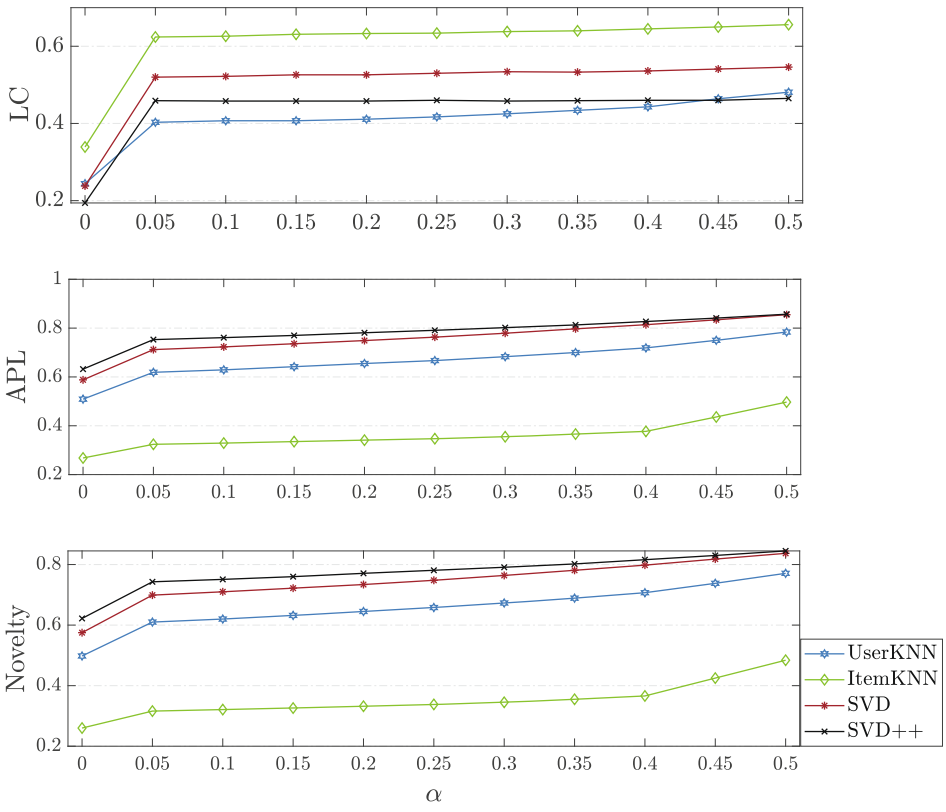

Fig. 4. The beyond-accuracy performance, (i.e., LC, APL, and Novelty) of the proposed *BDP* for the Ciao20.

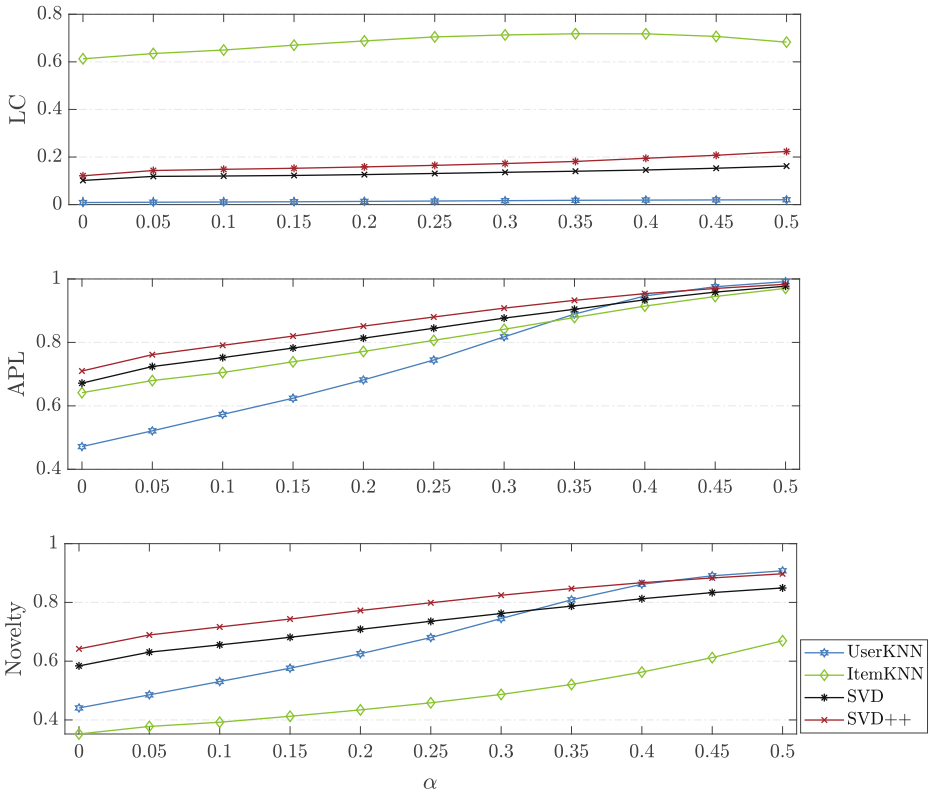

Fig. 5. The beyond-accuracy performance, (i.e., LC, APL, and Novelty) of the proposed *BDP* for the Per20.

also the case when the value of α is set to 0, usually perform relatively better on both Ciao20 and Per20 than the ML. This fact, in turn, intuitively limits the contributions of the *BDP* approach on recommendation quality for the Ciao20 and Per20, especially in terms of APL and Novelty perspectives.

The obtained results also demonstrate that the enhancements of the *BDP* on the LC performances are relatively lower than those on the APL and Novelty, especially for Per20 and ML datasets. Also, it can be concluded that α value has not a significant effect on the LC performances of the algorithms, as the obtained enhancements have interestingly become stable for all datasets in the case of α values larger than 0.05. On the other hand, as the value of α increases, the APL and Novelty quality of the produced recommendations significantly improves. This observation meets our expectations since higher α values penalize blockbuster items more in the generated ranked lists and help achieve more diverse and novel recommendations.

Moreover, especially in terms of APL and Novelty performances, the modified neighborhood-based CF algorithms, i.e., *UserKNN* and *ItemKNN*, with our *BDP* are the best-performing methods for the ML dataset. On the other hand, matrix factorization-based ones, i.e., *SVD* and *SVD++*, are usually more prominent for Per20 and Ciao20 datasets. This observation is unsurprisingly parallel with the outcomes of the experiments in the previous section - based on the *BRF* results presented in 2, the less blockbuster-biased algorithms are the neighborhood-based ones for the ML dataset, while the matrix factorization-based ones for both Per20 and Ciao20 datasets. Such trends of the original version of the algorithms are inevitably quite similar with their modified variants with our *BDP*, as lesser blockbuster-biased ranked lists mean having more qualified recommendation lists in terms of beyond-accuracy perspectives.

6.5.5. Evaluation of accuracy performance of the proposed BDP

While our proposed *BDP* is found to be highly effective in improving the beyond-accuracy quality of the recommendations, it quite likely leads to unignorable deteriorations in ranking accuracy, as such two quality aspects of the recommendations are generally conflicting goals. We, therefore, also perform additional experiments where we observed how the *BDP* negatively affects the accuracy performances of the utilized CF algorithms. To this end, we measure F1-scores and *n*DCG values of the modified CF algorithms with our *BDP* for all datasets and different α values varying 0 to 0.5, as presented in 6 and 7, respectively. Here, we again perform statistical significance tests to determine whether the observed deteriorations in the ranking accuracy are significant or not, as presented in the footnote of the table.

The obtained F1-scores and *n*DCG values demonstrate that almost all CF algorithms and their modified versions with our *BDP* produce more accurate recommendation lists on the ML and Per20 when compared to the Ciao20. The main reason for this consequence is that the average number of ratings per user on ML (around 165) and Per20 (around 512) datasets is significantly higher than those on the Ciao20 dataset (around 39). This fact allows improving precision and recall values of the produced recommendations in calculating F1-scores since the chance of getting a hit to appropriate items for users increases with more extensive user profiles. This fact also increases *n*DCG values since the numerator of this metric (i.e., DCG) positively correlated with the number of actual ratings per user.

Moreover, the best performing CF algorithms in terms of beyond-accuracy metrics for a dataset usually lead to the worst accuracy results for the corresponding dataset. This observation is as we expected because of the trade-off between the accuracy and beyond-accuracy aspects of the recommendations. For

Table 6
F1-scores of the *BDP* for different α values on ML, Ciao20, and Per20 datasets.

Dataset	**ML**				**Ciao20**				**Per20**			
α value	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*
0	0.089	0.047	0.078	0.071	0.015	0.017	0.021	0.019	0.073	0.060	0.047	0.037
0.05	0.083	0.043	0.074	0.066	0.013	0.016	0.017	0.015∗	0.067	0.057	0.042	0.032
0.10	0.078	0.039	0.071	0.063	0.013	0.016	0.017	0.014∗	0.061	0.056	0.039	0.029
0.15	0.073	0.036	0.068	0.060	0.014	0.016	0.016	0.014∗	0.054	0.054	0.036	0.026∗
0.20	0.067∗	0.033∗	0.065	0.056∗	0.014	0.016	0.016∗	0.013∗	0.047∗	0.052	0.032	0.023∗
0.25	0.060∗	0.030∗	0.060∗	0.052∗	0.014	0.014∗	0.016∗	0.013∗	0.039∗	0.050∗	0.029∗	0.020∗
0.30	0.053∗	0.025∗	0.056∗	0.047∗	0.015	0.014∗	0.015∗	0.013∗	0.030∗	0.047∗	0.026∗	0.017∗
0.35	0.045∗	0.022∗	0.051∗	0.042∗	0.015	0.014∗	0.015∗	0.012∗	0.021∗	0.044∗	0.023∗	0.015∗
0.40	0.038∗	0.020∗	0.046∗	0.036∗	0.016	0.014∗	0.015∗	0.012∗	0.015∗	0.040∗	0.020∗	0.013∗
0.45	0.032∗	0.017∗	0.040∗	0.028∗	0.017	0.013∗	0.015∗	0.012∗	0.011∗	0.035∗	0.018∗	0.011∗
0.50	0.007∗	0.008∗	0.019∗	0.011∗	0.018	0.013∗	0.016∗	0.011∗	0.010∗	0.030∗	0.016∗	0.010∗

∗ For significance at.%95; w.r.t. pure CF algorithm (i.e., $\alpha = 0$).

Table 7
*n*DCG values of the *BDP* for different α values on ML, Ciao20, and Per20 datasets.

Dataset	**ML**				**Ciao20**				**Per20**			
α value	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*	*UserKNN*	*ItemKNN*	*SVD*	*SVD++*
0	0.406	0.236	0.349	0.316	0.023	0.028	0.033	0.033	0.416	0.554	0.342	0.281
0.05	0.366	0.213	0.316	0.280	0.021	0.027	0.026	0.024∗	0.375	0.548	0.318	0.263
0.10	0.343	0.199	0.307	0.268	0.022	0.026	0.026	0.024∗	0.356∗	0.538	0.294	0.251
0.15	0.317∗	0.185∗	0.295∗	0.255∗	0.021	0.026	0.025	0.024∗	0.298∗	0.526	0.291∗	0.232∗
0.20	0.286∗	0.172∗	0.282	0.240∗	0.021	0.025	0.025	0.023∗	0.256∗	0.503	0.281∗	0.223∗
0.25	0.252∗	0.158∗	0.267∗	0.223∗	0.021	0.024∗	0.025	0.024∗	0.208∗	0.483∗	0.271∗	0.205∗
0.30	0.217∗	0.143∗	0.250∗	0.204∗	0.021∗	0.024∗	0.024∗	0.024∗	0.167∗	0.460∗	0.258∗	0.193∗
0.35	0.180∗	0.131∗	0.233∗	0.183∗	0.022	0.024∗	0.024∗	0.023∗	0.129∗	0.435∗	0.212∗	0.130∗
0.40	0.146∗	0.120∗	0.212∗	0.160∗	0.023	0.023∗	0.023∗	0.023∗	0.108∗	0.395∗	0.193∗	0.115∗
0.45	0.118∗	0.105∗	0.190∗	0.133∗	0.024	0.023∗	0.023∗	0.023∗	0.096∗	0.349∗	0.176∗	0.102∗
0.50	0.030∗	0.065∗	0.119∗	0.070∗	0.026	0.023∗	0.023∗	0.023∗	0.094∗	0.295∗	0.158∗	0.093∗

∗ For significance at.%95; w.r.t. pure CF algorithm (i.e., $\alpha = 0$).

Table 8
The comparison of the BDP with the benchmark methods in terms of both accuracy and beyond-accuracy metrics.

Dataset	Method	LC (⇑)	APL (⇑)	Novelty (⇑)	F1-score (⇑)	*n*DCG (⇑)
ML	**BDP**	**0.297**	**0.608**	**0.503**	**0.065**	**0.282**
	VaR	0.239∗	0.489∗	0.403∗	0.074	0.315
	ERP_{Mul}	0.235∗	0.849	0.754	0.008	0.044
	ERP_{Aug}	0.275∗	0.895	0.612	0.015	0.081
Ciao20	**BDP**	**0.526**	**0.749**	**0.734**	**0.016**	**0.025**
	VaR	0.194∗	0.710∗	0.704∗	0.014	0.022
	ERP_{Mul}	0.177∗	0.841	0.931	0.008	0.013
	ERP_{Aug}	0.184∗	0.878	0.868	0.009	0.014
Per20	**BDP**	**0.158**	**0.813**	**0.708**	**0.032**	**0.281**
	VaR	0.124∗	0.719∗	0.630∗	0.041	0.307
	ERP_{Mul}	0.114∗	0.878	0.831	0.006	0.066
	ERP_{Aug}	0.130∗	0.845	0.814	0.007	0.083

⇑; desired trend for the indicator.
∗ For significance at.%95; beyond-accuracy improvements.

example, according to the APL and Novelty scores obtained in the previous experiments, the *ItemKNN* usually performs relatively better than others for the ML dataset; however, the achieved F1-scores and *n*DCG values conclude that it is the worst CF algorithm in terms of accuracy performance among others for ML dataset, as can be seen in 6 and 7.

As can be followed by 6 and 7, as the value of α increases, the achieved F1-scores and *n*DCG values dramatically decrease in almost all schemes except for the *UserKNN* algorithm on the Ciao20 dataset, as we expected. However, the performed *t*-tests demonstrate that such deteriorations in ranking-accuracy performance of the algorithms are statistically insignificant with α values smaller than 0.2 in general, meaning that the obtained losses in ranking accuracy are negligible for such configurations. Considering that almost all α values help improve the beyond-accuracy quality of the produced recommendations significantly, we conclude that the BDP variant with $\alpha = 0.2$ is the most successful setting in maintaining reasonable ranking accuracy.

6.5.6. Comparison of our BDP method with benchmark methods

We also conduct several additional experiments to compare the beyond-accuracy and accuracy performances of the proposed *BDP* method (with $\alpha = 0.2$) against three benchmark approaches (i.e., *VaR*, ERP_{Mul}, and ERP_{Aug}) for ML, Ciao20, and Per20 datasets, as presented in 8. We also performed one-tailed *t*-tests to examine whether the improvements in beyond-accuracy metrics are statistically significant at the 95% confidence level. In doing so, we compare our *BDP* method against each benchmark method.

According to the obtained results, it can be concluded that our *BDP* significantly outperforms the *VaR* method in terms of all beyond-accuracy metrics (i.e., LC, APL, and Novelty) and achieves comparable accuracy performances (i.e., F1-score and *n*DCG) for all datasets. Likewise, the LC performance of our *BDP* method is significantly higher than both ERP_{Mul} and ERP_{Aug}. On the other hand, *ERP* variants seem to be the best-performing methods in terms of APL and Novelty perspectives. However, their accuracy performances are dramatically lower than both our *BDP* and *VaR* methods, and these deteriorations in ranking accuracy reach unignorable levels, especially for ML and Per20 datasets.

To sum up, we can conclude that our *BDP* method is a more successful debiasing strategy than all utilized benchmarks when considering both beyond-accuracy and accuracy recommendation quality simultaneously.

7. Limitations and Discussion

Recommendation algorithms typically produce ranked lists by considering the explicit feedbacks (i.e., ratings) derived from users' past interactions on items. Considering their internal mechanisms, most of them utilize the mean of the ratings provided for an item as an additive factor when calculating the final prediction score. For example, the *ItemKNN*, a neighborhood-based CF algorithm, typically produces a prediction for a specific item by summing up its average rating value with the weighted average of the ratings of most similar items. Likewise, *SVD*, a matrix factorization-based CF algorithm, utilizes the variations in rating values associated with items, named as biases, as an aggregation factor in estimating predictions. Thus, it can be concluded that the values of the provided ratings are critical for the calculated predictions and recommendation algorithms undesirably calculate higher predictions for the items that have higher averages when compared to ones having relatively lower averages. This also inevitably ends up having recommendation lists where the highly-liked items appear more frequently than others. In other words, the liking-degree of the items quite likely leads to a different and undesirable bias in recommendations from the famous popularity bias. Although the proposed blockbuster bias issue differentiates from popularity bias, it has limitations on detectability in some recommendation scenarios since it relies on determining the liking-degree of an item based on its existing explicit numerical rating values. Such limitation might be observed in market-basket analysis, which collects implicit ratings of users based on their click history or time spent on particular products. In such a scenario, items will receive ratings indicating if they attract attention or not on the user side. Therefore, calculated blockbuster scores would principally converge to popularity scores, obstructing the detection of such bias. In addition, some recommendation environments (such as YouTube videos) only collect ratings indicating if an item is liked or disliked by a user. In such a scenario, the liking-degree of a particular item cannot be estimated as the average rating score but as a ratio of likes over dislikes. Although such a scheme would allow computing a distinct blockbuster score for an item apart from its popularity, the discriminative property of such a score would be weak for understanding how blockbuster bias originates from data and propagates through recommendation algorithms.

8. Conclusion and future work

Recommender systems are highly effective machine learning tools that support the decision-making process of individuals by recommending items related to their interests and provide various benefits of the digital platform where they are used. However, the literature has recently acknowledged that recommendation algorithms are unwittingly biased for some particular items because of their specific characteristics, leading to unqualified recommendations. In this study, we evaluate such a problem from a new and different perspective - blockbuster items that are popular among individuals and highly liked by them, and focus on developing a practical solution to eliminate potential biases towards such items.

For this purpose, in this study, we first introduce a practical method estimating the blockbuster level of the items by appropriately incorporating their liking-degree and popularity. We also present a novel evaluation metric relying on this method, named *Blockbuster Recommendation Frequency (BRF)*, measuring the degree of potential bias towards blockbuster items in the produced recommendation lists. The performed experiments demonstrate that two benchmark datasets in three different application domains are highly imbalanced towards blockbuster items. Additionally, the most prominent four collaborative algorithms recommend such items too frequently in their produced ranked lists, indicating a solid and undesirable blockbuster bias has occurred in the recommendations.

To treat the adverse effects of such a blockbuster bias in recommendations, we also propose the *Blockbuster-Debiasing Procedure (BDP)* that focuses on re-ranking recommended lists by penalizing items based on their blockbuster levels. The empirical outcomes conclude that the proposed *BDP* highly effectively alleviates blockbuster bias in recommendations and helps to improve the recommendation quality in terms of beyond-accuracy aspects, such as coverage, diversity, and novelty, with maintaining reasonable ranking accuracy. Therefore, many digital platforms utilizing recommender systems can efficiently adopt the proposed *BDP* to achieve recommendation lists where such a blockbuster bias problem is treated; thus, they can generate more diverse recommendations by providing more visibility of the underappreciated items and increase the sales of such items.

Although our analysis shows the existence of the blockbuster bias in recommendations, such a bias might affect individuals differently based on their past interests in blockbuster items. Therefore, our future directions might include investigating how different users with different interest levels towards blockbuster items are affected by recommendation algorithms. Also, the proposed BDP can be improved by adjusting from such an user-centered perspective to achieve more fair recommendations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the Scientific Research Project Fund of Sivas Cumhuriyet University under the project number M-2021-811.

References

[1] F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, in: Recommender systems handbook, Springer, 2011, pp. 1–35..
[2] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys 52 (1). doi: 10.1145/3285029..
[3] R. Chen, Q. Hua, Y.S. Chang, B. Wang, L. Zhang, X. Kong, A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks, IEEE Access 6 (2018) 64301–64320, https://doi.org/10.1109/ACCESS.2018.2877208.

[4] M. Karimi, D. Jannach, M. Jugovac, News recommender systems - survey and roads ahead, Information Processing & Management 54 (6) (2018) 1203–1227, https://doi.org/10.1016/j.ipm.2018.04.008.
[5] M. Kunaver, T. Požrl, Diversity in recommender systems - a survey, Knowledge-Based Systems 123 (2017) 154–162, https://doi.org/10.1016/j.knosys.2017.02.009.
[6] M. Mendoza, N. Torres, Evaluating content novelty in recommender systems, Journal of Intelligent Information Systems 54 (2) (2020) 297–316, https://doi.org/10.1007/s10844-019-00548-x.
[7] T. Silveira, M. Zhang, X. Lin, Y. Liu, S. Ma, How good your recommender system is? a survey on evaluations in recommendation, International Journal of Machine Learning and Cybernetics 10 (5) (2019) 813–831, https://doi.org/10.1007/s13042-017-0762-9.
[8] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, X. He, Bias and debias in recommender system: A survey and future directions (2020). arXiv:2010.03240..
[9] L. Boratto, G. Fenu, M. Marras, Connecting user and item perspectives in popularity debiasing for collaborative recommendation, Information Processing & Management 58 (1) (2021), https://doi.org/10.1016/j.ipm.2020.102387 102387.
[10] L. Boratto, G. Fenu, M. Marras, The effect of algorithmic bias on recommender systems for massive open online courses, European Conference on Information Retrieval, Springer (2019) 457–472, https://doi.org/10.1007/978-3-030-15712-8_30.
[11] L. Hou, X. Pan, K. Liu, Balancing the popularity bias of object similarities for personalised recommendation, The European Physical Journal B 91 (3) (2018) 1–7, https://doi.org/10.1140/epjb/e2018-80374-8.
[12] E. Yalcin, A. Bilge, Investigating and counteracting popularity bias in group recommendations, Information Processing & Management 58 (5) (2021). URL: https://www.sciencedirect.com/science/article/pii/S0306457321001047 102608.
[13] P. Cremonesi, F. Garzotto, S. Negro, A.V. Papadopoulos, R. Turrin, Looking for "good" recommendations: A comparative evaluation of recommender systems, in: IFIP Conference on Human-Computer Interaction, Springer, 2011, pp. 152–168. doi: 10.1007/978-3-642-23765-2_11..
[14] Y. Liu, X. Cao, Y. Yu, Are you influenced by others when rating? improve rating prediction by conformity modeling, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 269–272, https://doi.org/10.1145/2959100.2959141.
[15] D. Liu, P. Cheng, Z. Dong, X. He, W. Pan, Z. Ming, A general knowledge distillation framework for counterfactual recommendation via uniform data, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 831–840, https://doi.org/10.1145/3397271.3401083.
[16] J.M. Hernández-Lobato, N. Houlsby, Z. Ghahramani, Probabilistic matrix factorization with non-random missing data, in: International Conference on Machine Learning, PMLR, 2014, pp. 1512–1520..
[17] T. Joachims, L. Granka, B. Pan, H. Hembrooke, G. Gay, Accurately interpreting clickthrough data as implicit feedback, in: ACM SIGIR Forum, vol. 51, Acm New York, NY, USA, 2017, pp. 4–11. doi: 10.1145/3130332.3130334..
[18] O. Hinz, J. Eckert, B. Skiera, Drivers of the long tail phenomenon: an empirical analysis, Journal of management information systems 27 (4) (2011) 43–70, https://doi.org/10.2753/MIS0742-1222270402.
[19] D. Kowald, M. Schedl, E. Lex, The unfairness of popularity bias in music recommendation: A reproducibility study, in: European Conference on Information Retrieval, Springer, 2020, pp. 35–42, https://doi.org/10.1007/978-3-030-45442-5_5.
[20] H. Abdollahpouri, Popularity bias in recommendation: A multi-stakeholder perspective (2020). arXiv:2008.08551..
[21] P. Sánchez, Exploiting contextual information for recommender systems oriented to tourism, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 601–605, https://doi.org/10.1145/3298689.3347062.
[22] Y.J. Park, A. Tuzhilin, The long tail of recommender systems and how to leverage it, in: Proceedings of the 2008 ACM conference on Recommender systems, 2008, pp. 11–18, https://doi.org/10.1145/1454008.1454012.
[23] D. Jannach, L. Lerche, I. Kamehkhosh, M. Jugovac, What recommenders recommend: an analysis of recommendation biases and possible countermeasures, User Modeling and User-Adapted Interaction 25 (5) (2015) 427–491, https://doi.org/10.1007/s11257-015-9165-3.
[24] C. Chen, M. Zhang, Y. Liu, S. Ma, Missing data modeling with user activity and item popularity in recommendation, in: Asia Information Retrieval Symposium, Springer, 2018, pp. 113–125, https://doi.org/10.1007/978-3-030-03520-4_11.
[25] H. Abdollahpouri, R. Burke, B. Mobasher, Controlling popularity bias in learning-to-rank recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 42–46, https://doi.org/10.1145/3109859.3109912.
[26] H. Abdollahpouri, R. Burke, B. Mobasher, Popularity-aware item weighting for long-tail recommendation (2018). arXiv:1802.05382..
[27] H. Abdollahpouri, R. Burke, B. Mobasher, Managing popularity bias in recommender systems with personalized re-ranking, arXiv preprint arXiv:1901.07555..
[28] D. Jannach, L. Lerche, M. Zanker, Recommending based on implicit feedback, in: Social Information Access, Springer, 2018, pp. 510–569. doi: 10.1007/978-3-319-90092-6_14..
[29] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, The unfairness of popularity bias in recommendation (2019). arXiv:1907.13286..
[30] E. Yalcin, Blockbuster: A new perspective on popularity-bias in recommender systems, in: 2021 6th International Conference on Computer Science and Engineering (UBMK), 2021, pp. 107–112, https://doi.org/10.1109/UBMK52708.2021.9558877.
[31] R. Sanders, The pareto principle: its use and abuse, Journal of Services Marketing..
[32] E. Yalcin, A. Bilge, Novel automatic group identification approaches for group recommendation, Expert Systems with Applications 174 (2021), https://doi.org/10.1016/j.eswa.2021.114709 114709.
[33] T.T. Nguyen, F. Maxwell Harper, L. Terveen, J.A. Konstan, User personality and user satisfaction with recommender systems, Information Systems Frontiers 20 (6) (2018) 1173–1189, https://doi.org/10.1007/s10796-017-9782-y.
[34] A. Vehtari, A. Gelman, J. Gabry, Practical bayesian model evaluation using leave-one-out cross-validation and waic, Statistics and computing 27 (5) (2017) 1413–1432, https://doi.org/10.1007/s11222-016-9696-4.
[35] Y. Koren, Factor in the neighbors: Scalable and accurate collaborative filtering, ACM Transactions on Knowledge Discovery from Data (TKDD) 4 (1) (2010) 1–24, https://doi.org/10.1145/1644873.1644874.
[36] E. Yalçin, Isbirlikçi filtreleme algoritmalarinin Çok-begenilen Ürünlere yönelik yanliligi, Bilecik Seyh Edebali Üniversitesi Fen Bilimleri Dergisi 8 (2021) 279–291, https://doi.org/10.35193/bseufbd.884634.
[37] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37, https://doi.org/10.1109/MC.2009.263.
[38] S. Wang, M. Gong, H. Li, J. Yang, Multi-objective optimization for long tail recommendation, Knowledge-Based Systems 104 (2016) 145–155, https://doi.org/10.1016/j.knosys.2016.04.018.
[39] E. Yalcin, A. Bilge, Binary multicriteria collaborative filtering, Turkish Journal of Electrical Engineering & Computer Sciences 28 (6) (2020) 3419–3437, https://doi.org/10.3906/elk-2004-184.
[40] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowledge-Based Systems 23 (6) (2010) 520–528, https://doi.org/10.1016/j.knosys.2010.03.009.