DUoR: *Dynamic User-oriented re-Ranking* calibration strategy for popularity bias treatment of recommendation algorithms

Mert Gulsoy[a], Emre Yalcin[*,b], Yucel Tacli[b], Alper Bilge[c]

[a]*Distance Education Research Center, Alanya Alaaddin Keykubat University, 07400 Antalya, Turkey*
[b]*Computer Engineering Department, Sivas Cumhuriyet University, 58140 Sivas, Turkey*
[c]*Computer Engineering Department, Akdeniz University, 07058 Antalya, Turkey*

Abstract

Recommender systems are widely used to provide personalized recommendations to users to help them navigate the vast amount of available content. They have become pervasive in various online applications. However, they often suffer from popularity bias, where popular items receive more recommendations, leading to potential issues such as limited diversity, homogenized user experience, perpetuating existing inequalities, and filter bubble effects. In this paper, we propose a novel approach to mitigate popularity bias by incorporating users' inclination towards item popularity. The proposed method incorporates a practical popularity inclination measuring strategy considering the dynamic preference tendencies of individuals to capture their unique propensities towards item popularity better and to provide more calibrated referrals regarding expectations of individuals on item popularity. Experimental results on benchmark datasets demonstrate that our proposed method effectively mitigates popularity bias by generating more diverse and balanced recommendations compared to several benchmark post-processing methods and outperforming them in diversity and fairness metrics according to the Borda count system. Overall, the proposed method presents a promising approach to addressing popularity bias in recommender systems by incorporating users' inclination towards item popularity and opens up potential directions for further research in the field.

Key words: Recommender systems, popularity bias, calibrated recommendations, user-oriented.

1. Introduction

Recommender systems (RSs) are essential tools in today's information-overloaded world, where the sheer amount of available content makes it difficult for users to find what they are seeking. These systems use data mining, machine learning, and artificial intelligence techniques to analyze users' behavior and preferences and recommend items that are most likely to be of interest (Ricci et al., 2015). The importance of RSs lies in their ability to personalize the user experience by providing relevant recommendations, thereby increasing user engagement and satisfaction, which leads to increased sales and profits for businesses that utilize RSs and improved user convenience and efficiency. Additionally, such systems also have the potential to improve the overall quality of information available by promoting relevant content and suppressing irrelevant or low-quality items. Therefore, developing and deploying effective RSs are crucial for optimizing the use of information and driving growth in various industries (Li et al., 2018).

One of the key challenges in RSs is popularity bias, where algorithms often overemphasize widely consumed items rather than genuinely matching users' unique tastes (Elahi et al., 2021; Fabbri et al., 2022). Prioritizing global popularity signals can trigger a *rich-get-richer* cycle, amplifying exposure for a small set of mainstream items while overshadowing many lesser-known options (Seymen et al., 2021; Yalcin and Bilge, 2022b). Beyond merely homogenizing the user experience, such bias also perpetuates inequalities by disproportionately favoring certain content,

*Corresponding author
Email addresses: mert.gulsoy@alanya.edu.tr (Mert Gulsoy), eyalcin@cumhuriyet.edu.tr (Emre Yalcin), yuceltacli@gmail.com (Yucel Tacli), abilge@akdeniz.edu.tr (Alper Bilge)

thereby limiting overall diversity in the information landscape. Addressing this shortfall demands recalibrating recommendations away from a strictly populist approach and toward methods that factor in each user's subjective interests, broader item representations, and latent item attributes (Ahanger et al., 2022).

A diverse range of strategies has emerged to combat these skewed recommendation patterns, incorporating richer data sources, refined objectives, and more holistic evaluation metrics. For instance, matrix factorization methods increasingly incorporate both user- and item-side information (Su et al., 2023), while graph-based algorithms leverage user–item network structures to unveil hidden relationships (Zhang et al., 2021). Hybrid systems that blend multiple models further expand the scope of personalization (Ahanger et al., 2022). Meanwhile, counterfactual evaluations gauge how well an RS can surface items the user might otherwise have missed (Abdollahpouri et al., 2017; Boratto et al., 2021), and the adoption of fairness constraints and diversity metrics helps ensure recommendations are equitable, varied, and consistent with users' underlying interests (Abdollahpouri et al., 2021, 2019a; Liu et al., 2023). Collectively, these methods underscore the ongoing quest to mitigate popularity bias and deliver truly personalized, inclusive recommendations.

As documented by Klimashevskaia et al. (2024), popularity bias manifests in multiple ways, including item-side inequities (e.g., underexposure of niche products) and user-side misalignment (e.g., overly mainstream suggestions for those who prefer lesser-known items). Although existing solutions often emphasize fair item exposure (i.e., producer- or item-side fairness), they risk overlooking the fact that individual users have distinct appetites for popular *vs.* niche content. This is where the notion of popularity calibration becomes critical: rather than forcing a global reduction in mainstream recommendations, the system should ensure each user's final list accurately reflects that user's personal inclination toward popular *vs.* niche offerings. This perspective extends earlier *calibration* concepts—originally developed to align recommendations with a user's own distribution of item attributes such as genres (Steck, 2018)—to the dimension of item popularity. By framing popularity bias as a calibration challenge, we move beyond simply lowering the system-wide prevalence of mainstream content; instead, we align each recommendation list with users' genuine preference distributions. This approach not only preserves individual satisfaction for those who truly enjoy mainstream hits, but also enables the discovery of long-tail items by those with more niche tastes, thereby facilitating a more equitable and diverse recommender ecosystem overall.

1.1. Problem statement and motivation

Despite the many techniques proposed to mitigate popularity bias in RSs, a persistent challenge lies in capturing users' true inclinations toward popular *vs.* niche content. While several studies have introduced popularity-debiasing strategies (Abdollahpouri et al., 2021, 2019c; Yalcin and Bilge, 2022a), many of these rely on simplifying assumptions—often measuring users' interaction with popular items as if it were a direct reflection of a preference for mainstream content. However, such assumptions overlook the multifaceted reasons behind user engagement (e.g., social influence, cultural relevance, or simple accessibility), and thus fail to calibrate recommendations to each user's authentic popularity appetite (Steck, 2018).

In real-world scenarios, a lack of popularity calibration triggers at least two major problems. From a user perspective, recommendations can become misaligned with personal tastes—particularly when a user has either a strong desire for niche items or, conversely, a genuine enthusiasm for mainstream hits. Without calibration, the system may inadvertently recommend an excess of popular items to someone whose interest lies in the long tail, or conversely constrain popular items for users who actually appreciate them. This misalignment undermines user satisfaction and trust, erodes perceived personalization, and can ultimately discourage continued engagement with the platform (Abdollahpouri et al., 2019a).

From a service-provider perspective, failing to calibrate popularity not only hampers the ability to deliver individually appealing content but also constrains potential revenue streams. Users who feel their tastes are not adequately reflected in recommendations may churn or reduce their interaction with the system, resulting in decreased monetization opportunities and reduced long-term brand loyalty (Boratto et al., 2021). Furthermore, overlooking niche preferences can lead to underexposure of certain items or services, impeding broader content exploration and stunting the organic diversity that can add value to a platform's catalog (Abdollahpouri et al., 2017).

The most common approach to gauge users' popularity tendency involves measuring the percentage of popular items in their profiles (Abdollahpouri et al., 2021, 2019c; Yalcin and Bilge, 2022a). However, this simplistic metric fails to capture the complexities of user tastes. A user might rate a popular item simply because it is widely available or

culturally prominent, rather than genuinely preferring it over less popular alternatives. Consequently, such measures risk reinforcing mainstream content at the expense of authentic personalization. Complicating matters further is the evolving nature of user preferences: factors like time, social context, and cultural trends may influence whether a user gravitates toward—or away from—popular content (Abdollahpouri et al., 2021). A static assessment of popularity bias thus locks users into outdated assumptions, perpetuating suboptimal recommendations. By contrast, a truly calibrated approach can adapt over time, continuously reevaluating a user's appetite for mainstream *vs.* niche offerings and aligning recommendations more faithfully with the user's evolving tastes.

Therefore, the impetus for more sophisticated and adaptive popularity calibration methods is clear. Systems should not merely count how often a user engages with popular items, but rather discern whether those interactions signal genuine preference or reflect other contextual factors. By accurately modeling these nuances, RSs can tailor the balance of mainstream and niche content to each individual, leading to higher user satisfaction, more equitable item exposure, and enhanced overall performance. Consequently, our work aims to fill this gap by proposing a dynamic calibration-driven framework that captures the complexities of user engagement with popular items and aligns recommendations more faithfully to users' evolving preferences for popularity.

1.2. Research Questions and Contributions

Although various strategies have been proposed to alleviate popularity bias, the aim of this work is to go beyond simplistic user-level metrics (e.g., counting the fraction of popular items) by dynamically customizing the final recommendation list in accordance with each individual's calibrated preferences. To achieve this, three focused research questions are posed, each reflecting a distinct aspect of our proposed approach, and resulting concrete contributions are given in the following.

RQ1: **In what way can we capture real user tendencies for item popularity without conflating them with unintentional or exploratory actions?**
A refined approach is devised for capturing each user's genuine inclination toward popular items, ensuring that incidental engagements (e.g., driven by social pressure, cultural prominence, or simple availability) are not conflated with true preference. By isolating users' actual tastes for mainstream *vs.* niche content, this metric delivers a more accurate popularity profile that underpins the subsequent dynamic re-ranking process.

RQ2: **How can we construct an iterative and efficient re-ranking pipeline that leverages users' popularity tendencies for well-aligned, user-centric recommendations?**
Rather than applying a one-shot post-processing re-ranking rule across all items at once, the proposed *DUoR* introduces a stepwise (i.e., dynamic) re-ranking framework. At each selection step, *DUoR* compares the partial list's current ratio of popular items with the user's desired popularity profile, ensuring that the final top-N list remains continuously calibrated to individual preferences. This iterative mechanism enables *DUoR* to adaptively shape recommendations to align precisely with each user's unique inclination toward mainstream *vs.* niche content.

RQ3: **How do we measure if a popularity-debiasing method truly captures each user's original taste for popular vs. niche items in the final recommendations?**
A new metric—*Rmse-PC*—is proposed to assess how well the recommended list aligns with each user's targeted balance of mainstream *vs.* niche content. By pairing this calibration-oriented measure with established RS benchmarks (e.g., precision, coverage), a comprehensive view of recommendation quality is achieved. Moreover, experiments on real-world datasets illustrate that the dynamic calibration framework not only advances user-level alignment with popularity preferences but also preserves or improves upon traditional performance metrics (e.g., accuracy, diversity).

These contributions deliver a calibration-centric solution to popularity bias by incorporating a dynamic re-ranking component. This end-of-pipeline approach offers broad applicability to a range of baseline recommendation methods, thereby addressing a persistent gap in the literature regarding the adaptive calibration of popular content. Moreover, *DUoR* can be effectively applied within dynamic recommendation environments, such as streaming services (e.g., YouTube or Spotify) and advertising on social media platforms. Maintaining a balance between user preferences for popular and niche content in these domains is critical for engagement and content diversity, making *DUoR* a practical and impactful solution.

1.3. Study structure and organization

The outline of the study is organized as follows: The following section provides a literature summary on calibration and popularity bias in RSs. Section 3 presents our proposed popularity-debiasing method, i.e., *DUoR*, and includes a toy example to clarify how it produces final recommendations for individuals. The following section presents the settings of experimental studies, such as utilized datasets and evaluation metrics, applied experimentation methodology, and considered benchmark popularity-debiasing strategies. Section 5 presents and discusses the experimental outcomes to evaluate the proposed method. Finally, Section 7 concludes the study and outlines ideas for future research directions. Note that the source code accompanying this work is made publicly available[1] for enabling the reproducibility of the experimental studies.

2. Related work

RSs have become ubiquitous tools for people's daily lives, providing personalized recommendations of products, services, and content. These systems rely on data-driven algorithms to predict user preferences and generate recommendations. However, there is a growing concern that these algorithms may suffer from popularity bias, where popular items are recommended more frequently than niche or lesser-known options (Ahanger et al., 2022). This bias can lead to the homogenization of user preferences and limited diversity in recommendations. In recent years, researchers have focused on developing methods to mitigate the popularity bias problem of recommendation algorithms. This section reviews the existing literature on the popularity bias problem and solutions for it in RSs.

The initial studies on this issue primarily investigate its formulation and degree induced by recommendation algorithms in different domains, including movies (Boratto et al., 2021; Borges and Stefanidis, 2021), online education (Boratto et al., 2019), tourism (Sánchez, 2019), books (Naghiaei et al., 2022a), and music (Jannach et al., 2016; Kowald et al., 2020). One pioneering study comprehensively analyzes several prominent recommendation methods, particularly collaborative filtering algorithms, and reveals that their recommendations heavily favor a tiny popular fraction of the item spectrum (Jannach et al., 2015). This study also examines the impact of algorithmic design and parameterization on popularity bias and suggests hyperparameter tuning to enhance recommendation diversity. Another related work evaluates a set of representative algorithms against various biases related to course category popularity, catalog coverage, and course popularity in massive open online courses (Boratto et al., 2019). It is concluded that the utilized algorithms can differ significantly in the courses they recommend and may display undesirable biases with corresponding educational implications. Additionally, another related study empirically assess the same representative algorithms and demonstrate how the popularity bias issue affects different stakeholders, such as individuals and item suppliers (Abdollahpouri, 2020).

Also, many recent studies investigate the potential discrimination of the popularity bias problem; in other words, how the bias propagated by the recommendation algorithms towards popular items unfairly affects different users with varying characteristics or different stakeholders of the system, such as users or service providers (Abdollahpouri, 2020), or user classes like age, race, or gender (Lesota et al., 2021). The initial study concerning the unfairness of this issue observes that users who are primarily interested in unpopular (i.e., niche) items are more affected by the popularity bias in movie recommendations (Abdollahpouri et al., 2019c). Such an unfairness problem across individuals has also appeared in books (Naghiaei et al., 2022a) and music recommendations (Kowald et al., 2020). This issue is also analyzed for five essential attributes related to users' rating behavior. It is concluded that exacting, highly interacting, and hard-to-predict individuals face extremely unfair recommendations compared to others, although they are the most substantial contributors to the system (Yalcin and Bilge, 2022a). Also, a recent study analyzes such an unfairness issue for five fundamental personality traits of the big-five factor model and concludes that users who are less extroverted or avoid trying new experiences are subjected to more unfair referrals regarding item popularity (Yalcin and Bilge, 2023).

In the literature, many practical solutions have been developed to tackle such a bias propagation of the recommenders. These approaches are commonly categorized as pre-, in-, and post-processing according to their integration strategy to the recommendation generation phase. Since a critical reason for the popularity bias problem is the existing inequalities in the distribution of the provided ratings, pre-processing methods aim to reduce such imbalances by

[1] `https://github.com/SiriusFoundation/DUoR`

altering data on which recommenders are trained. One example of a pre-processing approach initially classifies items in the catalog as head or tail based on their received ratings and then produces predictions for head items using all ratings, while for tail items, using ratings of the corresponding class (Park and Tuzhilin, 2008). Thus, the predictions estimated for tail ones are not affected by the dominance of popular items. Another approach of this category is constructing user-item tuples, where popular items are eliminated. The recommendation algorithms are later trained on such tuples rather than the original user-item rating matrix (Jannach et al., 2015). The last example of such approaches is employing a probability-distribution function defined according to the popularity level of the items in order to give more chances to tail items in the produced ranking-based recommendations (Chen et al., 2018).

In-processing popularity-debiasing methods aim to modify the internal mechanism of a recommendation algorithm to counteract its propagated bias towards item popularity. These methods are often referred to as non-generalizable solutions since they only assist specific recommenders and cannot be applied generally. One example is assessing how likely individuals are to dislike certain items and using this information to penalize popular items when estimating recommendations (Kamishima et al., 2014). Also, a recent study proposes an optimized variant of the well-known RankALS recommender to achieve more balanced recommendation lists regarding intra-list diversity and ranking accuracy (Abdollahpouri et al., 2017). Furthermore, another related work proposes a recommendation framework that first estimates common neighbors of two items based on their initial popularity, then removes the most popular ones to achieve a more balanced co-neighbor similarity index (Hou et al., 2018). Also, another approach is debiasing popularity by minimizing the correlation between the relevance of user-item pairs and item popularity, which results in a more even treatment of items in the long tail (Boratto et al., 2021). The last method in this category is based on Variational Autoencoders, which penalizes the scores assigned to items based on their historical popularity to reduce bias and improve diversity in the recommended results (Borges and Stefanidis, 2021).

On the other hand, post-processing debiasing methods aim at re-ranking recommendation lists produced by conventional algorithms or generating entirely new rankings by penalizing popular items and promoting unpopular ones. These methods are the most frequently adopted solutions, as their generalizability allows them to be easily integrated with the outputs of any recommender algorithm. For instance, one early approach suggests employing predefined user-specific weights to mitigate popularity bias while maintaining recommendation accuracy (Jannach et al., 2015). Similarly, another method (i.e., Paiw) computes synthetic ranking scores by weighting items inversely proportional to their popularity levels, after which items are re-ranked accordingly to finalize recommendations (Abdollahpouri et al., 2018). Adopting a comparable penalization strategy, we have previously introduced two re-ranking methods, namely Augmentative and Multiplicative, specifically tailored for addressing popularity bias in group recommendations (Yalcin and Bilge, 2021). The Augmentative approach emphasizes maintaining recommendation accuracy, whereas the Multiplicative strategy prioritizes the inclusion of niche items.

Recent developments in post-processing approaches have introduced more sophisticated methodologies to enhance item fairness and aggregate diversity. For example, a recent debiasing method employs stock volume constraints that limit the maximum allowable recommendation frequency of items based on their historical interaction patterns (Dong et al., 2021). This approach, grounded in a Minimum Cost Maximum Flow strategy, addresses popularity bias by effectively controlling item exposure and mitigating the Matthew Effect, thus ensuring fairness in user-item matching. Another prominent study introduces a diversity enhancement method inspired by the xQuad algorithm, explicitly promoting long-tail items to improve recommendation diversity and reduce popularity bias (Abdollahpouri et al., 2019a). Furthermore, Discrepancy Minimization (DM) uses a minimum-cost network flow model to optimize diversity by analyzing recommendation sub-graphs (Antikacioglu and Ravi, 2017). Lastly, another notable method, FA*IR, balances recommendations between popular and unpopular items by maintaining separate queues for each category and subsequently merging them into a single ranked list, effectively achieving a balanced item exposure (Zehlike et al., 2017).

Although these methods address popularity bias to varying degrees, many do not ensure that the final recommendations truly reflect each user's individual preferences. Steck (2018) proposes a method that aligns recommendation lists with each user's attribute distribution (e.g., item genres), thereby bridging standard metrics (e.g., accuracy) and user-level calibration. This idea has inspired further exploration into tuning recommender outputs to match user preferences across various features—including popularity. In addition, Deldjoo et al. (2024) offer a comprehensive survey on fairness in RSs, highlighting key research gaps and showing how popularity bias may overlap with other forms of user- or item-level discrimination. They emphasize that achieving multi-stakeholder fairness requires robust bias-mitigation strategies as well as precise measurement.

Calibration-driven solutions, therefore, have evolved to emphasize re-ranking or constraint-based formulations that preserve the user's desired distribution of item attributes—such as popularity—in the final recommendation list (Abdollahpouri et al., 2019a; Antikacioglu and Ravi, 2017; Zehlike et al., 2017). Many methods in this vein personalize item exposure or employ heuristic weighting to reduce the saturation of mainstream items (Dong et al., 2021). In particular, Abdollahpouri et al. (2021) expands this calibration concept by explicitly considering users' original propensities toward item popularity when performing debiasing. Their method determines users' actual interests across three item categories—popular (head), mid-level, and niche—based on historical interactions, integrating this knowledge to calibrate recommendations precisely. Similarly, Abdollahpouri et al. (2023) propose a calibrated recommendation strategy modeled as a minimum-cost flow optimization, embedding user-specific popularity preferences within system-level constraints, thus providing a flexible framework to simultaneously address calibration and popularity bias. Moreover, methods like CPFair (Naghiaei et al., 2022b) offer valuable insights into consumer- and producer-side fairness through constraint-based re-ranking, although not directly targeting popularity calibration. Despite not directly addressing popularity, CPFair underscores how fairness considerations across multiple stakeholders can enrich recommendation equity and might inspire future adaptations explicitly accounting for popularity bias. However, these calibration-based approaches share a critical limitation: they uniformly consider all ratings in users' profiles without adequately distinguishing between genuinely positive preferences and neutral or negative interactions. Such an indiscriminate approach can misrepresent users' true preferences for mainstream content, potentially leading to suboptimal calibration.

We also provide a concise overview of prominent recent strategies introduced for calibration and mitigating popularity bias in RSs in Table 1, evaluating their key characteristics such as user-centric calibration, processing type, calibration approach, fairness focus, popularity bias handling, scalability, and evaluation metrics. Within the table, the symbols denote the level of fulfillment for each attribute: "+" indicates that the method explicitly supports or incorporates the corresponding characteristic, "–" signifies that it lacks the particular attribute, and "±" indicates that it only partially addresses the attribute or that its applicability is context-dependent.

Distinctively, our proposed *DUoR* method emphasizes user-centric calibration through its *better-than-average* filtering approach, selectively considering only positively rated items when assessing users' true inclination toward item popularity. Unlike conventional methods, *DUoR* dynamically calibrates recommendations through an iterative re-ranking procedure, continuously adjusting the balance between popular and niche items according to each user's individual preference. This approach not only ensures precise calibration of recommendations in line with actual user interests but also inherently mitigates popularity bias by avoiding uniform popularity penalties that disregard individual taste variations. Furthermore, *DUoR*'s iterative greedy mechanism ensures high computational efficiency and scalability, making it particularly suitable for large-scale, real-world recommendation scenarios. Thus, *DUoR* significantly advances beyond existing one-shot re-ranking or static penalty-based approaches by simultaneously achieving user-specific calibration, effective popularity debiasing, enhanced recommendation diversity, and equitable item exposure.

In conclusion, while existing literature provides various approaches for tackling popularity bias, many are limited by their generic or non-adaptive nature and fail to adequately capture individual users' preferences regarding item popularity. As reflected in the comparative overview, there is a clear necessity for sophisticated, adaptive, and personalized solutions. *DUoR* specifically addresses this gap by accurately determining users' genuine interests and dynamically adjusting recommendations to deliver personalized, well-calibrated lists. Such precise and adaptive calibration enhances user satisfaction and represents a substantial improvement over prior methods in addressing popularity bias in RSs.

3. The proposed *Dynamic User-oriented re-Ranking* (*DUoR*) popularity debiasing approach

One of the main goals of RSs is to produce recommendations that fit the original propensities of individuals on item popularity. Considering such a fact in designing a treatment method for the popularity bias problem intuitively helps to achieve less biased and more calibrated referrals, maximizing the satisfaction of the individuals. Based on this motivation, in this study, we propose a novel post-processing method, namely *Dynamic User-oriented re-Ranking (DUoR)*, treating the popularity bias problem by following a user-centered approach.

The initial stage of the proposed *DUoR* is to determine the popularity tendency of individuals. The most common ways for such an aim is to compute what percentage of rated items in users' profiles are popular or to estimate their

Study	User-centric Calibration	Processing Type	Calibration Aprroach	Fairness Focus	Popularity bias Handling	Scalability	Evaluation Metrics
Zehlike et al. (2017)	– (group fairness, not user-level)	Post-Processing (re-ranking)	Constraint-based (fairness constraints)	+ Producer-/group-level fairness (protected items)	– (not specific to popularity)	± (constraint-checking for top-k)	Accuracy, group-fairness
Antikacioglu and Ravi (2017)	– (aims at system-level diversity)	Post-Processing (re-ranking)	Optimization-based (min-cost flow/discrepancy minimization)	– (diversity-oriented, not explicit fairness)	– (indirectly may reduce it by improving diversity)	± (optimization can be heavy for large data)	Accuracy, diversity, coverage
Abdollahpouri et al. (2018)	– (weights by popularity, not per-user)	Post-Processing (re-ranking)	Weighted re-ranking (popular items penalized)	– (focus on item coverage, not fairness)	+ (explicitly targets pop. bias)	+ (simple weighting, minimal overhead)	Accuracy, coverage, pop. bias measures
Steck (2018)	+ (aligns recommendations with user's attribute distribution)	Post-Processing (re-ranking)	Divergence minimization (KL/JS)	– (not explicitly fairness-driven)	± (may indirectly reduce bias, but not specifically targeted)	+ (greedy re-ranking suitable for top-N)	Accuracy, calibration measures
Abdollahpouri et al. (2019a)	+ (custom re-ranking per user's pop. preference)	Post-Processing (re-ranking)	Personalized weighting of item popularity	– (not explicit group or producer fairness)	+ (actively mitigates pop. bias at user level)	+ (straightforward re-ranking process)	Accuracy, coverage, pop. bias measures
Yalcin and Bilge (2021)	– (focus on group-level preferences, not individual user calibration)	Post-Processing (re-ranking for groups)	Weighted re-ranking (Augmentative/Multiplicative strategies)	Group-level (aimed at balancing popular vs. niche for a group)	+ (specifically targets reducing pop. bias)	+ (re-ranking approach suitable for group recommendations)	Accuracy, coverage, pop. bias measures, group satisfaction
Dong et al. (2021)	– (focus on item fairness constraints, not individual user inclination)	Post-processing (re-ranking via min-cost max-flow)	Optimization-based (stock volume constraints on items)	Producer-side/item fairness (limits over-exposure)	+ (reduces pop. bias by capping item recommendations	± (optimization-based; feasible for moderate scale)	Accuracy, fairness, coverage
Abdollahpouri et al. (2021)	+ (focuses on user-level perspective)	N/A (evaluation framework, not a direct re-ranking method)	– (no specific debiasing; proposes user-based bias metrics)	– (emphasis on user bias analysis, not group fairness)	± (analyzes/quantifies bias rather than mitigating it)	+ (evaluation approach feasible for large datasets)	user-level popularity bias, coverage, diversity
Naghiaei et al. (2022b)	- (not tailored to individual tastes)	Post-Processing (re-ranking)	Constraint-based re-ranking	Consumer & producer fairness	- (Not specifically aimed at mitigating popularity bias)	- (LP-based optimization)	Accuracy, novelty, fairness
Abdollahpouri et al. (2023)	+ (Popularity inclination based on rating counts)	Post-Processing (re-ranking)	Divergence minimization via min-cost flow	- (primarily user satisfaction)	+ (align system with user's popularity inclination)	- (can be slower for large N)	Accuracy, fairness, calibration
Proposed *DUoR*	+ (explicitly measures each user's genuine interest in popular vs. niche items, using "better-than-average" filtering)	Post-processing (iterative re-ranking)	Stepwise (at each rank selection, it compares the user's desired pop. ratio to the ongoing recommendation list)	User-level (prioritizes individual taste alignment; indirectly benefits diversity/system fairness)	+ (dynamically penalizes or promotes items based on the user's personal pop. inclination)	+ (iterative greedy approach; efficient for large datasets)	Accuracy, diversity, calibration, fairness, novelty

Table 1: Comparative summary of most prominent recent studies on calibration and popularity bias

average popularity ratios. In doing so, all voted items in rating profiles are considered in general. The most critical limitation of both approaches is that users might consume a popular item for several reasons, such as social marketing or behavioral inducements of "word-of-mouth" (Liu et al., 2019); however, it does not necessarily mean that they enjoyed it. Indeed, people prefer popular items in most domains, such as books and movies, out of pure curiosity for bestseller lists. However, after consuming the product or service, users have a clear opinion about them and reflect on it by providing their explicit ratings. The *DUoR* method aims to neutralize misevaluations in determining user propensity for item popularity by adopting a recent approach, referred to as *Better-than-average* (Tacli et al., 2022), which filters out items with higher ratings than the average rating value of the user profile. This way, users' original tendencies on popularity are estimated based on more concrete and factual terms.

By employing the *Better-than-average* strategy, the *DUoR* method initiates an offline categorization of items based on their popularity, utilizing the widely recognized Pareto Principle (Sanders, 1987). Specifically, the item catalog is divided into two essential sets: the *Head* (H) set and the *Tail* (T) set. The *Head* set (H) comprises the most popular items that receive 20% of all provided votes, while the remaining less popular items form the *Tail* set (T). This division ensures a clear distinction between widely consumed items and those with limited interaction.

Next, for each user u, the method calculates the average rating value across their entire profile, denoted as μ_u. This average rating serves as a personalized threshold for determining which items the user concretely enjoys. To identify these "enjoyed" items, the set E_u is constructed by filtering out only the items that have been rated higher than the user's average rating, μ_u. By doing so, the method focuses on the items that truly reflect the user's preferences, rather than assuming all rated items contribute equally to the user's tendency toward popular content. Finally, the user's tendency toward popularity, denoted as Pop_u, is calculated by determining the proportion of items in the E_u set that belong to the H set. In other words, the popularity inclination of user u is measured as the ratio of popular items (from

the H set) within their personalized set of E_u. This can be expressed mathematically as:

$$Pop_u = \frac{|E_u \cap H|}{|E_u|} \tag{1}$$

This approach offers a refined measure of user popularity inclination by considering only the items that a user genuinely enjoys (i.e., those rated higher than their average rating), rather than all rated items. As a result, it provides a more accurate representation of how much a user leans toward consuming popular versus less popular items. By focusing on personalized thresholds and filtering out non-enjoyed items, the *DUoR* method enhances its ability to capture the true preferences of users with diverse tastes, offering a more tailored and meaningful assessment of their popularity tendencies. This approach mitigates the risk of overestimating a user's inclination toward popular items based solely on interaction data, thereby improving the overall effectiveness of the recommendation process.

Following the offline phase where items are categorized based on popularity, the *DUoR* receives any proper recommendation algorithm output in the form of prediction values for each item i for the user, i.e., p_i, and constructs user prediction vector $p_{1\times n}$ where n is the total number of items in the catalog. Later, it sorts items in descending order based on their calculated p_i scores and selects top-M ones as a candidate set of items that can be recommendable for the user, i.e., $C_{1\times M}$. Here, M is selected as $M >> N$, where N is the size of the final recommendation list to be produced. Note also that considering only pure prediction scores when constructing the C set helps to achieve a decent ranking-accuracy performance. Finally, it achieves the final top-N recommendation list for the user, i.e., *RecList*, by dynamically selecting N items as described in Algorithm 1.

Algorithm 1 The proposed *DUoR* method

Input: User prediction vector ($p_{1\times n}$), Candidate item set ($C_{1\times M}$), and Pop_u
Initialize: $RecList_{N\times 1} \leftarrow null$ ▷ *top-N recommendation list to be returned*
Constructing *RecList*:
1: **for** $k \leftarrow 1$ to N **do**
2: **if** $k = 1$ **then:**
3: $i \leftarrow \arg\max_{i\in C}(p_i)$ ▷ *Selecting the item with the highest prediction score in the C*
4: add i into *RecList* at its k. position
5: remove i from C
6: **else:**
7: $tempC \leftarrow null$ ▷ *temporary set of candidate items*
8: Compute $Pop_{RecList}$ ▷ *Ratio of popular items in the current RecList*
9: **for all item** i in C ($j \leftarrow 1$ to $M - k$) **do**
10: **if** $i \in H$ **then:**
11: **if** $Pop_{RecList} < Pop_u$ **then:**
12: add i into *tempC*
13: **end if**
14: **else if** $i \in T$ **then:**
15: **if** $Pop_{RecList} > Pop_u$ **then:**
16: add i into *tempC*
17: **end if**
18: **end if**
19: **end for**
20: $i \leftarrow \arg\max_{i\in tempC}(p_i)$ ▷ *Selecting the most recommendable item in the tempC*
21: add i into *RecList* at its k. position
22: remove i from C
23: **end if**
24: **end for**
return *RecList*

More specifically, when the *DUoR* constructs *RecList* for the user, it initially selects the item with the highest prediction score in C (i.e., the item at the top of the C) and injects it at the top of *RecList* (see Lines 2-6). This

item is also later removed from the C set. Then, the remaining items to be included in the *RecList* are determined by repeatedly performing the following steps: (*i*) computing the ratio of popular items in the current version of the *RecList*, i.e., $Pop_{RecList}$ (see Line 8), (*ii*) cloning items in the C into a temporary candidate set, i.e., *tempC*, by disregarding popular/head (resp. unpopular/tail) ones in C if $Pop_{RecList}$ is higher (resp. lower) than their original popularity inclination ratio, i.e., Pop_u (see Lines 9-19), (*iii*) selecting the item with the highest prediction score in the *tempC* and injecting it into the *RecList* (Lines 20-21), and finally (*iv*) removing it from the set C (Line 22). These processes are repeatedly performed until the *RecList* is filled with N items. Here, it can be demonstrated that our proposed *DUoR* is highly beneficial in achieving two important goals. The first is that more calibrated recommendations regarding the popularity tendency of users can be obtained as the final recommendation lists are adjusted by comparing the popularity ratio of the *RecList* with Pop_u at each iteration. The second is that the overall popularity bias level of the system significantly diminishes, as it harshly penalizes popular items in constructing recommendation lists, especially for catchy users who are more interested in less popular items.

3.1. Computational complexity analysis of the DUoR

The computational complexity of the proposed *DUoR* method can be analyzed based on the operations carried out in the Algorithm 1. Since the *DUoR* method consists of multiple steps, including an initial offline item classification based on the Pareto Principle, candidate set selection, popularity-based filtering, and final ranking, the complexity is calculated for each component in Table 2. We also explain these steps in detail in the following.

Step 1: The initial classification of items into Head (H) and Tail (T) sets using the Pareto Principle is an offline process. Sorting all items based on their popularity (e.g., the number of ratings) takes $O(n \log n)$, where n is the total number of items. This step is crucial to establish the popularity distribution but is performed once offline and not repeated during recommendation generation.

Step 2: The system receives predicted scores for each item and sorts them to identify the top-ranked items. Sorting the prediction scores requires $O(n \log n)$.

Step 3: The next step is selecting the top M items from the sorted predictions to form the candidate set C. This takes $O(M)$, where M is significantly larger than N.

Step 4: In the first iteration, the item with the highest prediction score is selected from C and added to the recommendation list, which involves $O(1)$ complexity.

Step 5: For each subsequent recommendation, the system filters items from C based on the user's popularity tendency (Pop_u). This involves iterating over the remaining items and checking their classification (Head or Tail), which takes $O((M - k) \times N)$, where k is the current iteration number.

Step 6: After filtering, the system selects the item with the highest prediction score from the temporary candidate set *tempC*, which requires $O(M - k)$ operations.

Step 7: Constructing the final recommendation list requires repeating the filtering and selection process for each of the N recommendations. Therefore, the total complexity is $O(N \times M)$, which dominates the online phase.

Step	Description	Complexity
1	*Offline item classification (Pareto-based)*: Categorizing items into H and T sets	$O(n \log n)$
2	*Initial prediction sorting*: Sorting the prediction scores for all n items	$O(n \log n)$
3	*Candidate set selection*: Selecting top M items from sorted predictions to form $C_{1\times M}$	$O(M)$
4	*First item selection*: Selecting the item with the highest score for the recommendation list	$O(1)$
5	*Popularity filtering loop*: Iterating over the remaining items, filtering based on popularity	$O((M - k) \times N)$
6	*Item selection from filtered set*: Selecting the most recommendable item from C	$O(M - k)$
7	*Total complexity for top-N construction*: Repeating the filtering process for N iterations	$O(N \times M)$

Table 2: Computational complexity of the proposed *DUoR* method.

The overall computational complexity of the *DUoR* method, calculated for a single user, includes both offline and online phases. The offline Pareto-based item classification incurs a one-time cost of $O(n \log n)$, applicable to all users. For the online recommendation generation phase for one user, the complexity is $O(N \times M)$, where M is the size of the candidate set, which is larger than the final recommendation list size N. Given that these steps are computed for a single user, the computational overhead remains manageable. Taking into account the offline pre-processing and efficient filtering mechanisms, the computational complexity is reasonable and scalable, even for larger datasets.

3.2. An illustrative example regarding how our proposed DUoR works

In this section, we present a toy example to clarify how the proposed *DUoR* method achieves final recommendation lists for users. To this end, we first give a prediction matrix example in Table 3 containing the predictions of a user on a candidate set of pre-determined 15 items (C), which are computed utilizing a proper recommendation algorithm. The pre-determined class of the items regarding their popularity is also given in the last row of this table. Here, items are labeled as Popular/Head (H) or Unpopular/Tail (T) using the Pareto Principle based on their received ratings. Suppose also that the percentages of (H and (H sets within the original profile are 40% and 60%, respectively; thus, the value of Pop_u to be used in performing the *DUoR* is 0.4.

Candidate items (C)	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}
Prediction value	4.5	4.2	3.1	4.8	4.7	3.6	4.9	3.1	4.2	4.1	4.6	2.6	1.5	1.8	2.2
Pre-defined class	H	T	H	H	H	T	H	T	H	T	H	T	H	T	H

Table 3: Computed predictions and pre-defined classes of the items in the C for the toy example

Suppose we intend to generate a top-4 recommendation list for this user via the *DUoR* method detailed in Algorithm 1. The following summarizes the iterations of the *DUoR* to be performed.

1st iteration: It injects i_7 into the *RecList* and removes it from the C, as it has the highest prediction score across the items; hence, the current set of *RecList* is as $\{i_7\}$ and the ratio of its popular items, i.e., $Pop_{RecList}$, is now 1.

2nd iteration: Since the original popularity inclination of the user, i.e., Pop_u, is lower than $Pop_{RecList}$ of the current *RecList* (i.e., $0.4 < 1$), it only selects tail items by eliminating popular ones in the C set to achieve the *tempC* set; thus, the *tempC* set becomes $\{i_2, i_6, i_8, i_{10}, i_{12}, i_{14}\}$. Since i_2 has the highest prediction value among the items in the *tempC*, the *DUoR* method includes it into *RecList* and removes it from the C set. Thus, the set of the *RecList* becomes as $\{i_7, i_2\}$ and its current $Pop_{RecList}$ value is calculated as 0.5.

3rd iteration: Since Pop_u value is still lower than the $Pop_{RecList}$ value of the current *RecList* (i.e., $0.4 < 0.5$), the *tempC* is again constructed by filtering out the tail items in the C, i.e., $tempC = \{i_6, i_8, i_{10}, i_{12}, i_{14}\}$. Since i_{10} has the highest prediction value among the items in the *tempC*, this item is included in the *RecList* and removed from the C. Thus, the set of *RecList* becomes as $\{i_7, i_2, i_{10}\}$ and its current $Pop_{RecList}$ is calculated as around 0.34.

4th and final iteration: Since Pop_u value is not lesser than the $Pop_{RecList}$ value of the current *RecList* anymore (i.e., $0.4 \not< 0.34$), the *tempC* is constructed by selecting popular items in the C set; thus, the set *tempC* is now as $\{i_1, i_3, i_4, i_5, i_9, i_{11}, i_{13}, i_{15}\}$. The item with the highest prediction value among them is i_4; therefore, it is included in the *RecList* and removed from the C. Thus, the set of *RecList*, the final recommendation list to be produced, becomes as $\{i_7, i_2, i_{10}, i_4\}$, and its its current $Pop_{RecList}$ is calculated as 0.5.

As can be followed by the presented example, two of the four items recommended via the *DUoR* method are popular. In other words, the percentage of popular items in its generated top-4 recommendation is 50%. However, if only pure predictions computed by the recommendation algorithm were considered without applying the *DUoR* method, the $\{i_7, i_4, i_5, i_1\}$ set was to be recommended to the user. However, all of these four items belong to the H (i.e., popular) class, which leads to having a recommendation list whose ratio of popular items is 100%. In a nutshell, it can be concluded that our proposed *DUoR* method is highly beneficial in mitigating popularity bias in recommendations. Also, remember that the original popularity inclination of the user (i.e., Pop_u) is 0.4; another essential superiority of the *DUoR* method is that it can achieve more calibrated recommendations in terms of popularity inclination, which makes individuals more satisfied with their received referrals and consequently improves the efficiency of the system.

4. Experimental setup

We perform a comprehensive set of experimental studies to evaluate the performance of the proposed *DUoR* method. This section details the utilized datasets and evaluation metrics, applied experimentation methodology, and the considered baseline popularity-debiasing methods in the experiments.

4.1. Datasets

We use several benchmark real-world datasets collected for different application domains to provide generalizable findings in the experiments. These are well-known MovieLens-1M[2] (ML), Douban Book[3] (DB), and Yelp[4] datasets, including user preferences on movies, books, and reviews on local businesses, respectively. Provided ratings in all datasets are discrete and on a five-star scale (i.e., [1, 5]). They also have varying sizes and sparsity ratios, which enables us to observe how such characteristics of datasets affect the performance of the proposed *DUoR* method. Detailed properties of the ML, DB, and Yelp datasets are also given in Table 4.

Dataset	# of users	# of items	# of ratings	Sparsity (%)	Average rating	# of ratings per item	# of ratings per user
ML	6,040	3,952	1,000,209	95.8	3.58	253.1	165.6
DB	13,024	22,347	792,062	99.7	4.05	35.4	60.8
Yelp	16,239	14,284	198,397	99.9	3.77	13.9	12.2

Table 4: Detailed information about the datasets

4.2. Evaluation metrics

To comprehensively evaluate the performance of the proposed *DUoR* method, we consider several criteria regarding recommendation quality in the experiments, as in a highly related study (Yalcin and Bilge, 2022a). In doing so, we adopt four well-known statistical accuracy metrics, *Precision*, *Recall*, *F1-score*, and *nDCG*, and four beyond-accuracy metrics, *Average Percentage of Long Tail Items (APLT)*, *Entropy*, *Novelty*, and *Long-Tail Coverage (LTC)*. We also consider *Mean Rank Miscalibration (MRMC)* metric measures the fairness of recommendations by averaging the Jensen-Shannon divergence between a user's item preferences and the top-N recommended items. Furthermore, we propose a novel user-centric evaluation metric, *Root mean square error in Popularity Calibration (Rmse-PC)*, to concretely measure how much the produced recommendations calibrated with users' original inclination on item popularity. The following explains the proposed *Rmse-PC* and other nine evaluation metrics in detail.

Root mean square error in Popularity Calibration (Rmse-PC): Suppose that P_i is the popularity rate of an item i calculated as the proportion of the number of users voting item i to the total number of users in the system, and $PopN_u$ refers to the average popularity of the top-N items recommended to user u via any popularity-debiasing method and is calculated by averaging the popularity rate (P) values of each item in the top-N list. This metric first calculates the original inclination of each user u on item popularity, i.e., $OrgPop_u$, by calculating the proportion of *Head* items in the set of their enjoyed items set (E_u), which is constructed by only considering the items whose vote is higher than the average rating values on their profiles. Here, the set of *Head* items in the catalog is determined by the Pareto principle (Sanders, 1987), as explained in detail in Section 3. Under these settings, the overall *Rmse-PC* of the utilized popularity-debiasing method can be calculated using the formula given in Eq. 2.

$$Rmse-PC = \frac{1}{|U|}\sqrt{\sum_{u\in U}(PopN_u - OrgPop_u)^2} \tag{2}$$

[2]https://grouplens.org/datasets/movielens/1m/
[3]https://www.douban.com/
[4]https://www.yelp.com/

where U is the set of all users in the system.

Here, considering only the appreciated items rather than the entire set of rated items in the users' profiles helps better capture users' inclinations towards popularity; thus, the in/consistency between popularity exposed by the recommendations and users' real expectations from the system can be computed more correctly.

Mean Rank Miscalibration (MRMC): Steck (2018) introduced a metric to evaluate the calibration of recommendations, which was later generalized by (Abdollahpouri et al., 2019b) and (da Silva et al., 2021) as Miscalibration. In our work, we build upon these ideas to propose an enhanced version, the *Mean Rank Miscalibration (MRMC)*, designed to assess the fairness of recommendations in terms of item popularity. This approach focuses on re-ranking recommendations based on the user's historical interactions with popular (Head) and non-popular (Tail) items, ensuring that the recommended list aligns with the user's preferences. Rather than using the traditional Miscalibration (MC) metric, we compute the Jensen-Shannon divergence for each item in the recommendation list and accumulate the results. This method allows for a more granular evaluation of how well the recommendations reflect the user's engagement with different popularity groups.

For each user u, the Rank Miscalibration ($RMC(u)$) is calculated by analyzing the first N items in the recommendation list. At each position $j \in \{1, 2, \ldots, N\}$, the Jensen-Shannon divergence is computed between two distributions: p, which represents the user's historical interactions with popular (Head) and non-popular (Tail) items, and $q(R^*_j)$, which represents the popularity distribution of the recommended item at rank j. This calculation takes into account not only the item at rank j but also the cumulative distribution of item popularity for all preceding items in the recommendation list. For instance, at the 3rd position (i.e., top-3), the popularity distribution considers not only the 3rd item but also the categories of the first two items. Items are categorized using the Pareto principle (Sanders, 1987), as detailed in Section 3. These divergence values are cumulatively summed, and the total is then divided by N to provide an average miscalibration score for each user, as shown in Eq. 3.

$$RMC(u) = \frac{\sum_{j=1}^{N} \text{Divergence}(p, q(R^*_j))}{N} \tag{3}$$

Once the $RMC(u)$ is computed for each user, the final Mean Rank Miscalibration (MRMC) is calculated by averaging the results across all users (U), as in Eq. 4:

$$MRMC = \frac{\sum_{u \in U} RMC(u)}{|U|} \tag{4}$$

This metric evaluates the quality of the generated recommendations by considering the cumulative divergence across the top-N items in the list. The aim is to assess how well the recommended items align with the user's historical interactions, leading to a fairer evaluation of the recommendations.

***Precision, Recall, and F1-score*:** We use well-known *Precision*, *Recall*, and *F1-score* metrics to evaluate the accuracy quality of the top-N recommendation lists produced by the popularity-debiasing methods. The *Precision* score of a top-N list ($P@N_u$) for a user u is the percentage of the recommended N items suitable for the user. The *Recall* score of the corresponding recommendation list ($R@N_u$) is the ratio of the number of suggested suitable items to the number of all rated items in the user's profile. We set the threshold value for determining suitability as '3.5', as it is generally accepted in the literature that '4 'and '5' are affirmative ratings on a 5-star rating scale (Bobadilla et al., 2010). Finally, we calculate the *F1-score* of the considered top-N list ($F1@N_u$) as the harmonic mean of its $P@N_u$ and $R@N_u$ scores, as formulated in Eq. 5.

$$F1@N_u = 2 \times \frac{P@N_u \times R@N_u}{P@N_u + R@N_u} \tag{5}$$

***Normalized Discounted Cumulative Gain (nDCG)*:** The *nDCG* is another crucial statistical accuracy metric used in the experiments, which considers the genuine ratings of the recommended items and their positions in the

produced top-N list. If $r_{u,i}$ is user u's actual rating provided for item i, the $DCG^u_{N_u}$ and $nDCG^u_{N_u}$ values of the top-N recommendation list for u are calculated using Eqs. 6 and 7, respectively.

$$DCG^u_{N_u} = r_{u,i_1} + \sum_{n=2}^{N_u} \frac{r_{u,i_n}}{log_2(n)} \tag{6}$$

$$nDCG^u_{N_u} = \frac{DCG^u_{N_u}}{IDCG^u_{N_u}} \tag{7}$$

Here, the $IDCG^u_{N_u}$ value is the maximum possible gain for u and is obtained by ideally ordering N items.

***Average Percentage of Long Tail Items (APLT)*:** In addition to the previously mentioned statistical accuracy-based metrics, we also employ the *APLT* measure to evaluate the ranking-based recommendations. The *APLT* metric quantifies the percentage of long-tail items in the recommended list, indicating the ability of the algorithm to suggest niche items. Therefore, this metric is extremely helpful in measuring how a debiasing method is successful in featuring unpopular (i.e., tail) items in their produced recommendations. Here, the Pareto Principle determines the set of *tail* items in the whole catalog (Sanders, 1987). Given T as the set of assigned *tail* items, the *APLT* score of a top-N recommendation list produced for user u is computed as given in Eq. 8.

$$APLT_u = \frac{|i \in T \cap N_u|}{N} \tag{8}$$

It should be noted that although $APLT_u$ reflects the level of diversification for a particular user, the overall *APLT* score represents an average of such diversification levels. Hence, higher *APLT* scores do not necessarily indicate that the considered algorithm can provide high-quality referrals in terms of diversity. However, lower *APLT* scores certainly indicate a general lack of diversity in recommendation lists. Therefore, we also employ *Entropy* as an alternative metric to measure the level of diversification in all recommendation lists, which is explained in detail below.

***Entropy*:** The *Entropy* metric measures how often recommendation algorithms suggest each item in a catalog, as discussed in (Elahi et al., 2021). In other words, it quantifies the degree of inequality in the frequency distribution of suggested items. A higher *Entropy* value for an algorithm indicates greater homogeneity in item frequencies and diversity in recommendations, which is desirable for system administration to ensure fair competition in the market. To compute the *Entropy* value, we first construct the union of top-N lists for each individual, including duplicated items. Let N_{all} represent this union, and let the set of all items in the catalog be denoted by $i_1, i_2, ..., i_K$, where K is the total number of items. The formula for calculating the *Entropy* value for a recommendation algorithm is given by Eq. 9:

$$Entropy = -\sum_{i \in K} (\Pr(i)) \log_2(\Pr(i)) \tag{9}$$

Here, $\Pr(i)$ represents the relative frequency of item i appearing in the N_{all} set.

***Novelty*:** We also employ the *Novelty* metric as an alternative beyond-accuracy measure in our experiments, which evaluates the ability of recommendation algorithms to suggest items that users have not previously rated. The *Novelty* metric measures the percentage of previously unrated items in the top-N list, which indicates the algorithm's potential to offer novel alternatives to users. To calculate the *Novelty* score for a user u, we first define I_u as the set of items rated by user u. Then, the *Novelty* score for the top-N recommendations provided to user u, denoted by $Novelty_u$, can be computed using Eq. 10:

$$Novelty_u = \frac{|i, i \in N_u \text{ and } i \cap I_u = \emptyset|}{N} \tag{10}$$

Here, N_u represents the set of top-N items recommended to user u.

***Long-Tail Coverage (LTC)*:** In addition to the previously discussed beyond-accuracy metrics, we also use the *LTC* metric in our experiments to measure the extent to which recommendation algorithms cover the tail part of the item catalog, as defined in (Abdollahpouri, 2020). The *LTC* metric is calculated by first constructing a non-recurring aggregation of top-N lists produced for each user, denoted by $\mathbb{N}$, where duplicated items in individual top-N lists are eliminated. We then determine the intersection item set, $I_{\mathbb{N}\cap T}$, by considering items that appear both in $\mathbb{N}$ and the tail item set T determined by the Pareto Principle. Finally, the *LTC* score for a recommendation algorithm is calculated as the ratio of the size of $I_{\mathbb{N}\cap T}$ to the size of T, as shown in Eq. 11. Higher *LTC* values indicate that the algorithm provides better coverage of the less popular or tail items in the recommendations.

$$LTC = \frac{|I_{\mathbb{N}\cap T}|}{|T|} \tag{11}$$

It should be noted that higher values of all the discussed metrics, except for *Rmse-PC*, indicate less biased and more qualified recommendations in terms of both accuracy and beyond-accuracy perspectives.

4.3. Experimentation methodology

In the experiments, we selected two prominent collaborative filtering algorithms, *Hierarchical Poisson Factorization* (*HPF*) (Gopalan et al., 2015) and *Spherical k-means* (*SKM*) (Salah et al., 2016), as base recommenders before applying any popularity-debiasing methods. The primary reason for selecting both HPF and SKM is that they are known to exhibit strong popularity biases, making them excellent candidates for evaluating our debiasing approach (Yalcin and Bilge, 2022a; Gulsoy et al., 2023; Yalcin and Bilge, 2023). We can directly observe how effectively our proposed method addresses such a bias and its adverse effects on recommendation calibration by employing algorithms that favor popular items. Furthermore, HPF and SKM are widely recognized in the RSs literature, ensuring that our findings align with commonly referenced baselines in the field. Additionally, both algorithms are implemented using the Python-based comparative framework Cornac[5] (Salah et al., 2020), which provides a robust and scalable environment for conducting reliable and consistent experiments. This combination of factors ensures that *HPF* and *SKM* serve as effective baseline algorithms for demonstrating the improvements introduced by our debiasing techniques.

More specifically, the *HPF* is a probabilistic model developed for providing high-quality recommendations based on implicit feedback. Unlike traditional methods, *HPF* excels in capturing the long-tail distribution of user activity and item popularity by modeling user interactions with finite attention budgets. It assumes that user behavior follows a Poisson distribution, where each user is represented by a vector of latent preferences and each item by a vector of latent attributes. These vectors are constrained to be non-negative, and the model places Gamma priors over them to encourage sparse representations, effectively modeling user heterogeneity and item popularity. This enables HPF to outperform classical matrix factorization methods, especially in dealing with sparse, implicit feedback data, without needing to generate negative examples artificially. The model also scales efficiently, making it suitable for large datasets in real-world recommendation systems (Gopalan et al., 2015).

On the other hand, the *SKM* algorithm is a clustering approach designed for high-dimensional data where the data points lie on a unit hypersphere. The *SKM* algorithm iteratively partitions the data into clusters by maximizing the sum of cosine similarities between data points and their corresponding cluster centroids. This ensures that points with similar directions are grouped together. In recommender systems, *SKM* is used to cluster users or items based on their interaction data, making it easier to generate recommendations by associating users with clusters of similar preferences. The online version of *SKM*, known as OSK-means, updates centroids incrementally with each new data point, allowing it to handle dynamic environments where new users, items, or ratings are continuously introduced. This characteristic makes it highly efficient for large-scale collaborative filtering tasks, providing scalability without a significant increase in computation time. In our experiments, we configured both *HPF* (Gopalan et al., 2015) and *SKM* (Salah et al., 2016) with the default hyperparameter settings recommended in their respective original publications and implemented in the Cornac library. By adhering to these default parameters, we not only maintain consistency with established practices and align with the canonical usage of these algorithms in the literature, but also enhance the reproducibility of our results.

[5] https://cornac.preferred.ai/

For the experimentation methodology, we employ the widely-used leave-one-out strategy (Wong, 2015). In this approach, each user in the dataset is treated as the test user in turn, while the remaining users form the training set. The system first predicts the ratings or preferences for the test user across all items using either the *HPF* or *SKM* algorithms. From these predictions, the top M items (in our case, $M = 100$) that receive the highest predicted scores are selected to form the candidate set (C) for further processing.

Next, the proposed *DUoR* method is applied to this candidate set to generate a top-N recommendation list for the test user, where $N = 10$. This means that, from the candidate set of 100 items, *DUoR* refines the list and selects the 10 most appropriate recommendations for the user based on its algorithm.

This process is repeated for every user in the system, ensuring that each individual is tested as the test user once. The proposed *DUoR* method, along with other benchmark methods, is applied to generate the top-10 recommendation lists for all users. In the final step, the quality of these recommendation lists is assessed using the evaluation metrics described in the previous section.

This thorough process allows for a comprehensive assessment of each method's performance across the entire dataset, ensuring that the evaluation is robust and representative of real-world recommendation scenarios.

4.4. Benchmark methods and parameter-tuning

Although several attempts relying on different strategies (i.e., pre-, in-, and post-processing) to cope with the popularity bias problem exist, we consider the most prominent examples of post-processing ones in the experiments to achieve a fair comparison as the proposed *DUoR* method is also a post-processing user-centric approach. To this end, to compare with the *DUoR* method, we select eight benchmark methods, namely *FA*IR, Discrepancy Minimization (DM)*, *Popularity-aware item weighting (Paiw)*, *Augmentative (Aug)*, *Multiplicative (Mul)*, *eXplicit Query aspect diversification (xQuad)*, *Minimum Cost Maximum Flow (MCMF)*, and *Calibrated Popularity (CP)* which are also explained in detail in the following. Note that all these approaches except *MCMF* follow a re-ranking strategy but significantly differ in how they generate synthetic ranking scores based on their purposes.

***FA*IR* (Zehlike et al., 2017):** This method aims to achieve a balanced representation of two item groups in recommendations: protected and unprotected. In this case, protected items refer to long-tail items ($M \cup T$), whereas unprotected items are head items (H). The algorithm operates by constructing separate queues for both protected and unprotected items and then merging them based on normalized scores. This ensures that protected items receive greater exposure in the recommendation process, effectively addressing the underrepresentation of long-tail items.

***Discrepancy Minimization (DM)* (Antikacioglu and Ravi, 2017):** In this method, the primary objective is to enhance the total number of unique recommended items, commonly known as aggregate diversity. To achieve this, the minimum-cost network flow method is utilized to identify recommendation sub-graphs that optimize diversity. The authors of this work define a target distribution for item exposure (i.e., the desired frequency of each item appearing in recommendations), which serves as a constraint in their objective function. Consequently, the aim is to minimize the deviation between the actual recommendation frequency of each item and its target distribution.

***Popularity-aware item weighting (Paiw)* (Abdollahpouri et al., 2018):** It first calculates the logarithmic inverse weight for items based on their popularity, where tail items get higher weights while head ones receive lower ones. Then, it combines such weights and computed predictions for items to achieve ranking scores to be used in achieving final recommendations. It uses a controller parameter λ to adjust their importance when integrating them. Here, we set the value of λ to 0.5 to make a fair balance between such two aspects of the recommendations.

***EXplicit Query Aspect Diversification (xQuad)* (Abdollahpouri et al., 2019a):** This method is one of the most prominent popularity-debiasing methods and is used to provide final recommendations by re-ranking items based on their popularity into two categories: *head* or *tail*. The primary objective of this method is to enhance the diversity of final recommendations by balancing the *head* and *tail* items, depending on the original propensities of users towards these categories. To achieve this, the *xQuad* uses a controlling parameter, denoted by λ, to adjust the balance between recommendation accuracy and diversity. The *xQuad* is applied on a candidate set of

items as in our proposed *DUoR* method. Note that the size of the candidate set is set as 100 for both *xQuad* and *DUoR* to achieve fair competition. We also set the value of λ to 0.5 to balance accuracy and diversity aspects.

***Augmentative (Aug)* (Yalcin and Bilge, 2021):** This method combines pure prediction scores with the inverse weights of items' popularity levels to calculate synthetic ranking scores. However, in its combination strategy, prediction scores are employed as the driving force, and item weights are utilized as an additive factor; thus, ranking accuracy is more critical when mitigating popularity bias.

***Multiplicative (Mul)* (Yalcin and Bilge, 2021):** As in the *Aug* method, a re-ranking strategy penalizing popular items in final recommendations is utilized in the *Mul* method. However, its combination strategy differs significantly from the *Aug*. More specifically, utilizing the item weights as a factoring coefficient when computing ultimate ranking scores, the goal of *Mul* is to penalize popular items in the recommendation lists harshly; thus, its primary aim is to recommend ranking lists where a maximum possible number of unpopular items included.

***Minimum Cost Maximum Flow (MCMF)* (Dong et al., 2021):** This method aims to provide exposure fairness and simultaneously reduce popularity bias in recommendations. For this purpose, it utilizes stock volume constraints on items to restrict the maximum allowable recommended occurrences to be directly proportional to their historical interaction frequencies; thus, the Matthew Effect on item popularity in final recommendations is mitigated. Furthermore, it is based on a Minimum Cost Maximum Flow model to address the optimal user-item matching issue with constraints.

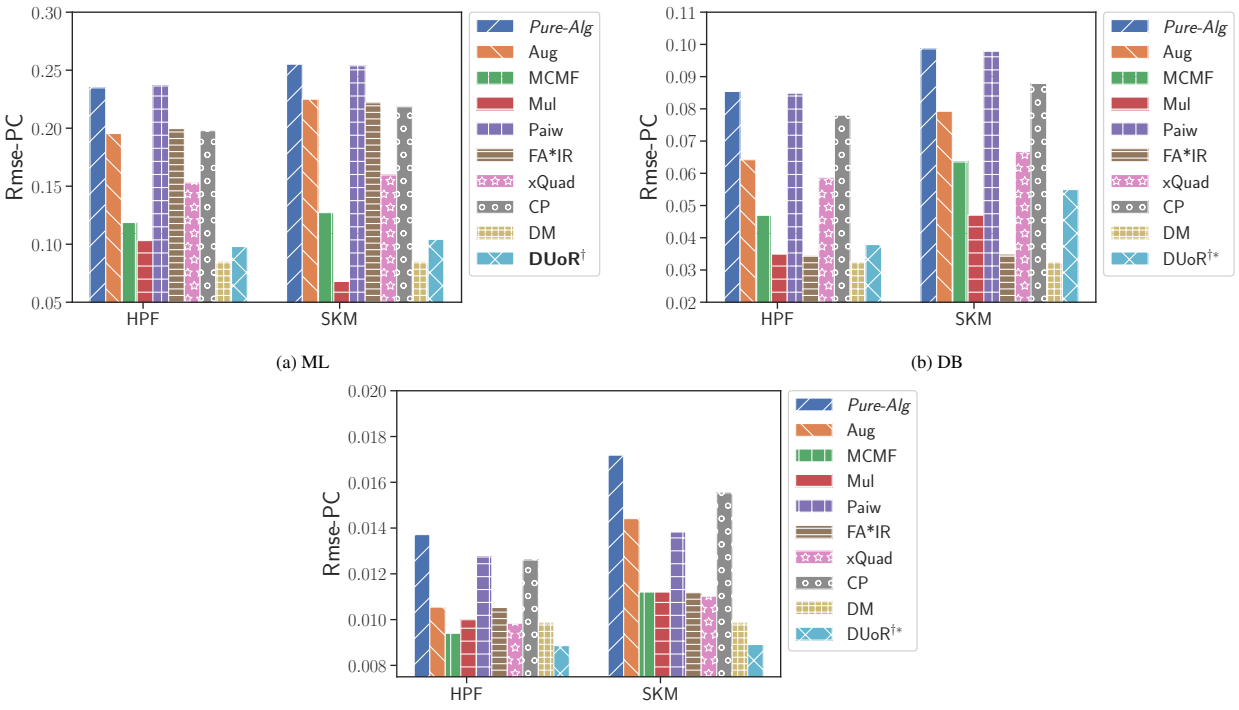
***Calibrated Popularity (CP)* (Abdollahpouri et al., 2021):** This method aims to reduce popularity bias by personalizing recommendations based on the user's past interactions with items of varying popularity. Inspired by Steck's calibrated recommendation approach (Steck, 2018), CP re-ranks an initial list of recommendations to match the user's historical engagement with popular, moderately popular, and non-popular items. The algorithm uses Jensen–Shannon divergence to measure the difference between the item's popularity distribution in the user's profile and the recommended list, minimizing this divergence to achieve better alignment. As one of the few user-centric approaches in the literature, this method uniquely focuses on tailoring recommendations to each individual user's preferences rather than applying a global optimization. The balance between relevance and popularity calibration is controlled by the parameter λ, ensuring recommendations reflect the user's preferences across different popularity groups. In our experiments, we set the size of the candidate set is set as 100 for both *CP* and *DUoR* to achieve fair competition and set λ to 0.9, as this value provides an optimal balance for achieving diversified recommendations for the CP.

5. Experimental results

This section presents the outcomes of conducted experiments to comprehensively evaluate the performance of popularity-debiasing methods, including our proposed *DUoR*. In these experiments, we compare the *DUoR* method with eight well-known treatment methods, *FA*IR*, *DM*, *Paiw*, *Mul*, *Aug*, *xQuad*, *MCMF*, and *CP* in terms of accuracy (via *Precision*, *Recall*, *F1-score*, and *nDCG*) and beyond-accuracy recommendation quality such as diversity (via *APLT*), inequal representation of the items (via *Entropy*), Novelty (via *Novelty*), and coverage (via *LTC*). We also compare the proposed *DUoR* model with the benchmarks via *Rmse-PC* to analyze variations in their performance for producing calibrated user recommendations and with *MRMC* to evaluate the fairness of the generated recommendations. Before performing any popularity-debiasing method in these experiments, we apply the *HPF* and *SKM* as the base recommenders, and we denote their pure performance as *Pure-Alg* in the presented results. This enables us to how much the adverse effects of the popularity bias are resolved with the debiasing methods. Note also that we perform paired *t*-tests to assess the statistical significance of the improvements at the 95% confidence level. In all the figures below, $*$ symbol indicates that *DUoR* significantly outperforms the following best debiasing method for the HPF algorithm, while † symbol denotes a significant improvement over the following best debiasing method for the SKM algorithm.

Lower *Rmse-PC* and *MRMC* values indicate more accurate recommendations in terms of popularity calibration. From the results in Fig. 1 and Fig. 2, we can conclude that *DUoR* consistently ranks as one of the most effective methods for delivering recommendations that align with users' preferences regarding item popularity. As shown

in these figures, *DUoR* achieves lower *Rmse-PC* and *MRMC* values compared to the eight benchmark methods, particularly for the ML and Yelp datasets, regardless of the recommendation algorithm used. Also, the superiority of our *DUoR* method regarding both *Rmse-PC* and *MRMC* seems statistically significant for almost all cases, as can be followed by Figs. [1 - 2]. Since real-world datasets tend to be large and sparse, and given that the Yelp dataset is larger than both ML and DB (see Table 4), the empirical results demonstrate that *DUoR* performs relatively well in producing recommendations consistent with users' popularity preferences. The strength of *DUoR* lies in its ability to dynamically adjust top-*N* recommendation lists by considering users' popularity preferences in each iteration, resulting in finely calibrated recommendations.



(a) ML (b) DB

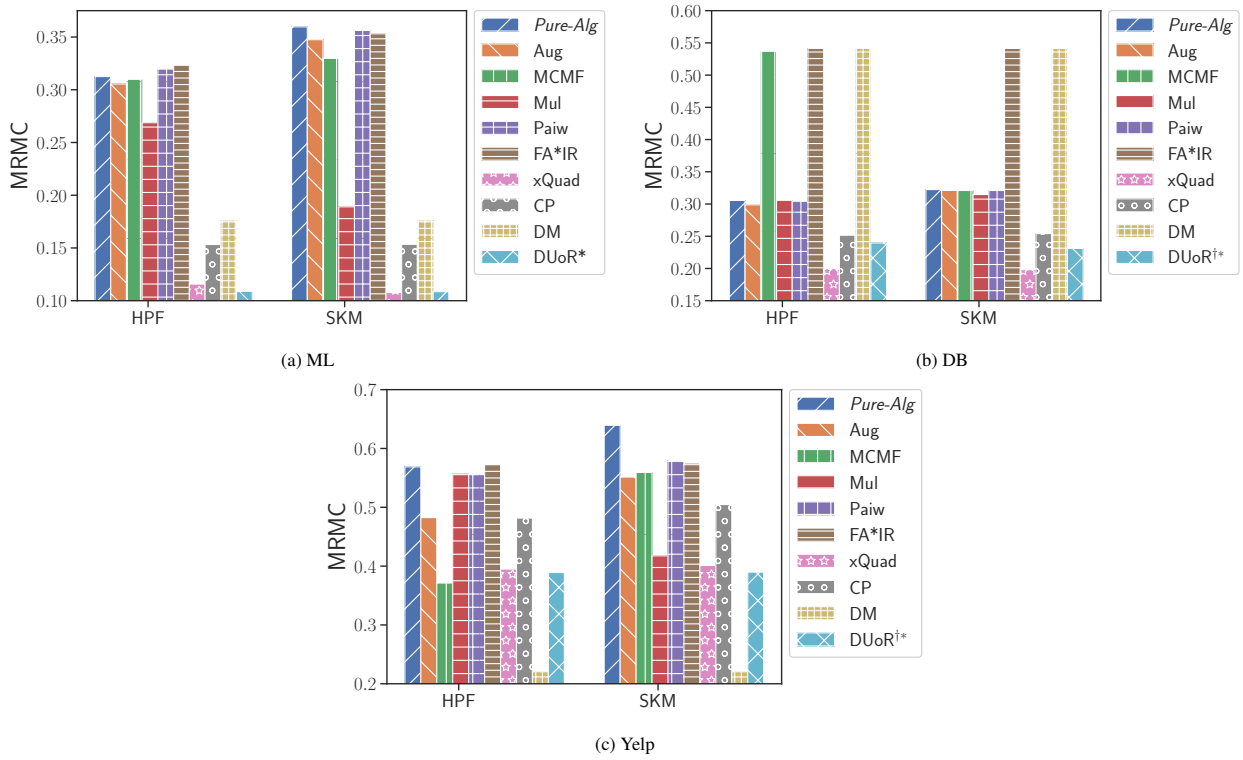
(c) Yelp

Figure 1: *Rmse-PC* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets. (*Rmse-PC* improvements compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

The *y*-axis of Figs. [1a - 1c] shows that the lowest *Rmse-PC* values are achieved in the Yelp dataset, followed by DB and ML, regardless of the recommendation algorithm or debiasing method used. This can be attributed to the fact that Yelp has the fewest ratings per item, followed by DB and ML (see Table 4). This results in a lower popularity ratio for items, which naturally leads to lower *Rmse-PC* values in sparser datasets.

Although we expect the *Rmse-PC* and *MRMC* values of debiasing methods to be higher than those of *Pure-Alg*, Fig. 2 shows that some debiasing methods, especially for the DB dataset, exhibit even higher *MRMC* scores than *Pure-Alg*. This highlights a critical issue: while these debiasing methods are inherently designed to lower the overall popularity levels of recommended items, they unfortunately generate recommendations that are poorly calibrated to users' profiles in terms of item popularity. As a result, these methods lead to recommendations that misalign with user preferences, causing a decrease in the calibration of recommendations and ultimately resulting in a poorer user experience.

To compare the accuracy performances of the popularity-debiasing methods, we also present their achieved *Precision* and *Recall* values in Fig. 3 and *F1-score* and *n*DCG values in Fig. 4. As can be followed by both, the highest accuracy results are usually obtained for the pure version of the recommenders (i.e., before any debiasing method is applied), with some exceptions. In other words, all debiasing methods usually lead to notable decreases in overall



(a) ML

(b) DB

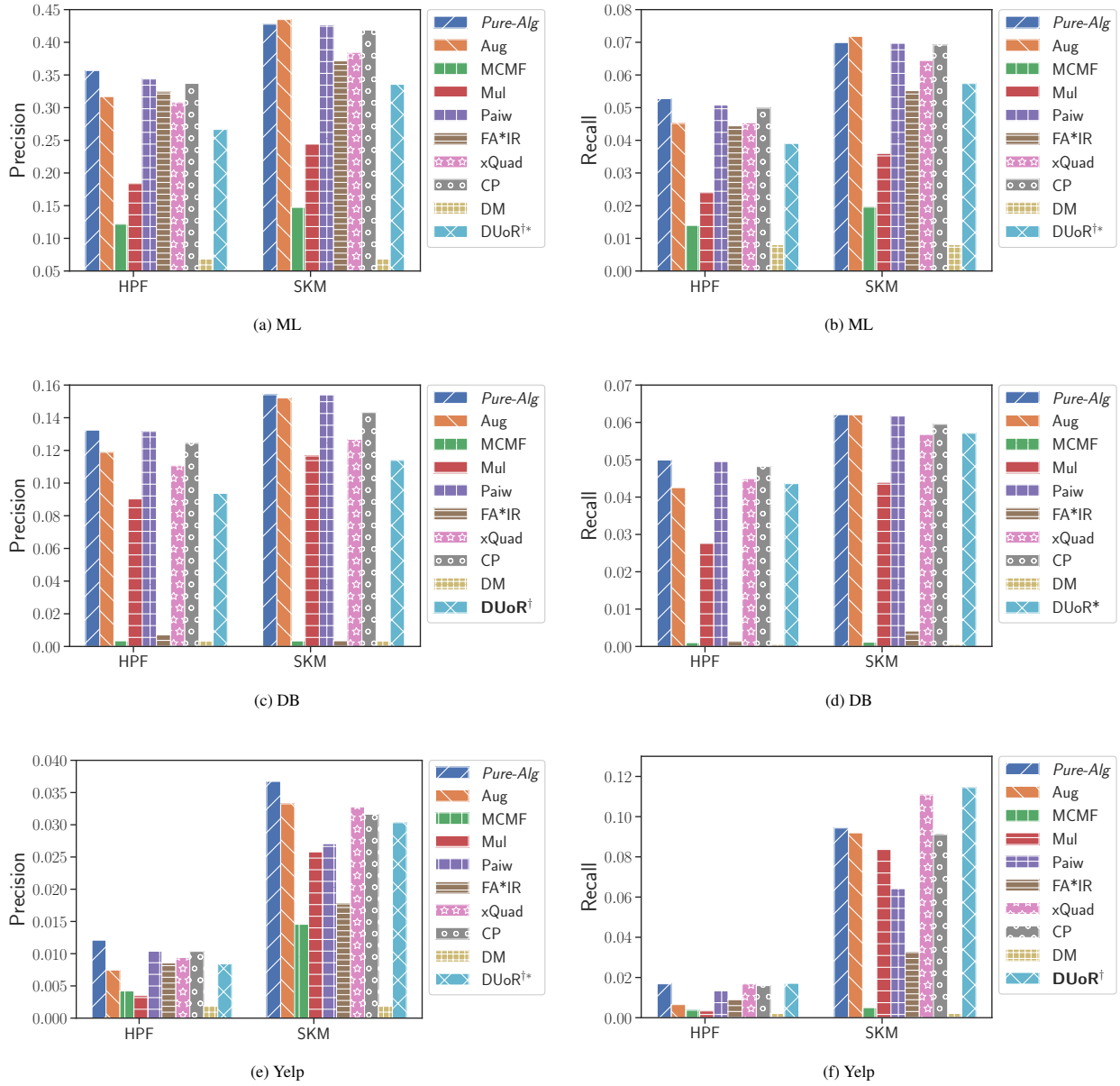
(c) Yelp

Figure 2: *MRMC* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets. (*MRMC* improvements compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

ranking accuracy. This observation is expected due to the nature of debiasing methods and the well-known trade-off between accuracy and beyond-accuracy recommendation quality. The treatment methods proposed for the popularity bias issue focus on improving beyond-accuracy recommendation quality by featuring unpopular items more in recommendations, which inevitably leads to having lower accuracy performance in the offline experimental evaluation as accuracy metrics evaluate top-N lists by considering actual ratings of suggested items. Since the total number of ratings provided for popular items are significantly higher than for unpopular ones, recommending unpopular items decreases accuracy performance, even if improving beyond-accuracy aspects such as coverage, novelty, or diversity.

Paiw, *Aug*, and *CP* consistently provide the most accurate recommendations among the debiasing methods regarding *Precision*, *Recall* and *F1-score* metrics particularly for the ML and DB datasets. This is expected, as their higher *Rmse-PC* values in Fig. 1 suggest that recommending more popular items improves overall ranking accuracy. However, our proposed *DUoR* method demonstrates comparable accuracy to these methods across most settings. This is especially evident in larger and sparser datasets, as reflected in the outcomes of the Yelp dataset. Notably, *DUoR* achieves the highest *Recall* scores among all methods when the *SKM* algorithm is applied to the Yelp dataset, as shown in Fig. 3f.

The highest *n*DCG values for both the *HPF* and *SKM* algorithms on the Yelp dataset are achieved with the *xQuad*, *Aug*, and *DUoR* methods, with no significant differences observed between *DUoR* and the other two benchmark methods, as can be followed from Fig. 4. This finding is crucial, as *n*DCG is widely regarded as one of the most effective metrics for assessing recommendation accuracy, given that it evaluates both the relevance of recommended items and their positions in the top-N lists. Therefore, considering the earlier *Rmse-PC* outcomes, we can conclude that our proposed *DUoR* method is a highly effective popularity-debiasing approach, offering well-calibrated recommendations with strong accuracy. This is due to its item selection mechanism, which takes into account both popularity calibration and the highest prediction value at each iteration. Moreover, its advantage becomes more pronounced in large-scale datasets, which more closely resemble real-world recommendation scenarios.



(a) ML (b) ML

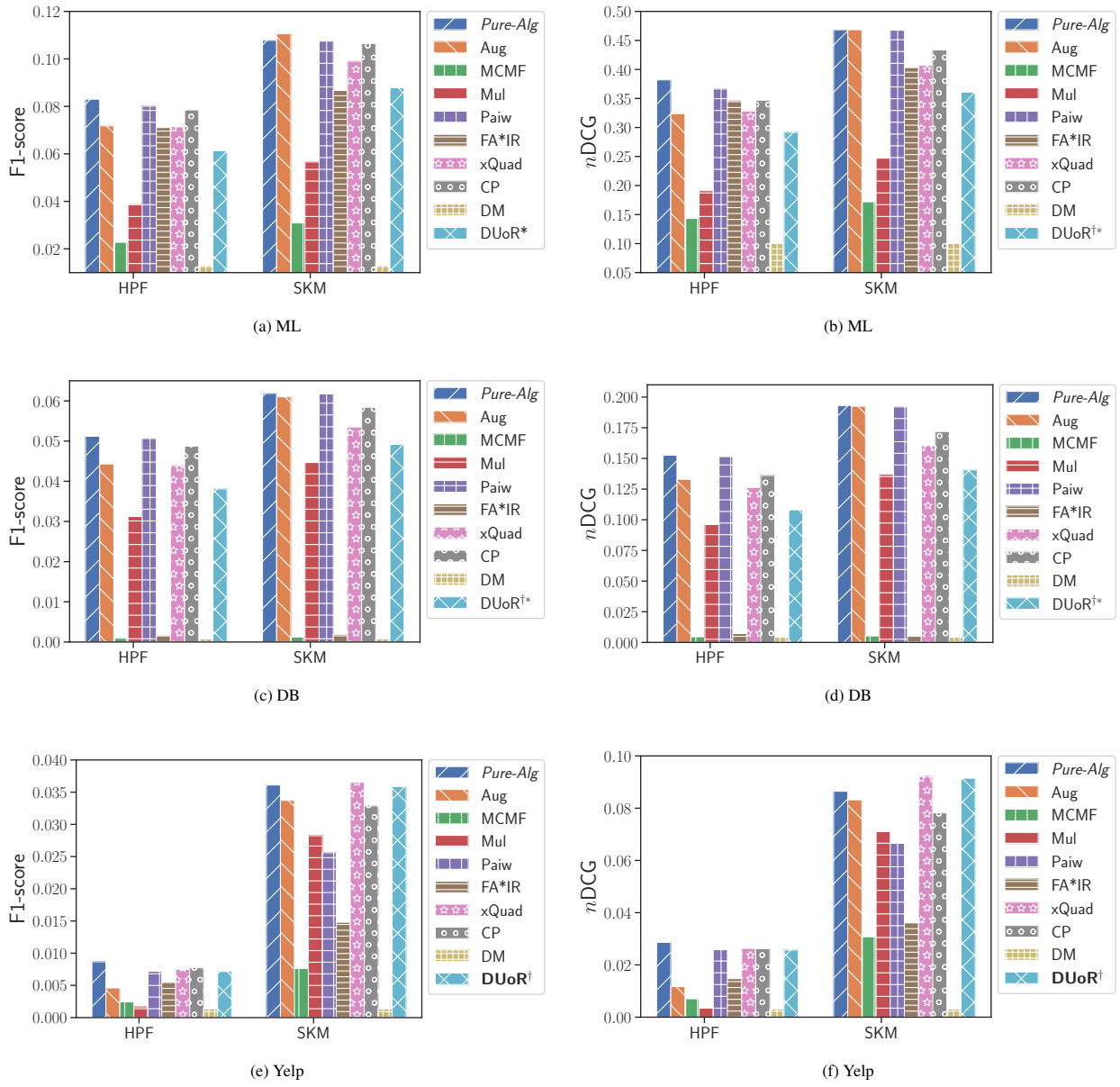
(c) DB (d) DB

(e) Yelp (f) Yelp

Figure 3: *Precision* and *Recall* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a-b) ML, (c-d) DB, and (e-f) Yelp datasets. (The improvements for corresponding metric compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

We also compare the proposed *DUoR* with the benchmark methods regarding their ability to feature unpopular items in final recommendations and improve diversity performance. To this end, we present their *APLT* scores in Fig. 5 to analyze the level of their ability to include tail items in their produced recommendation lists. Also, we compare them in terms of *Entropy* in Fig. 6 to assess to what extent they equally represent the items in the catalog within their recommendations.

As shown in Fig. 5, the proposed *DUoR* method achieves the highest *APLT* scores in all cases except when the *HPF* algorithm is applied to the Yelp dataset. The strength of *DUoR* lies in its ability to balance calibration while incorporating a large number of tail items into the recommendation lists, resulting in more diverse and well-rounded recommendations. Furthermore, *DUoR* performs exceptionally well in the *Entropy* results, as shown in



(a) ML (b) ML

(c) DB (d) DB

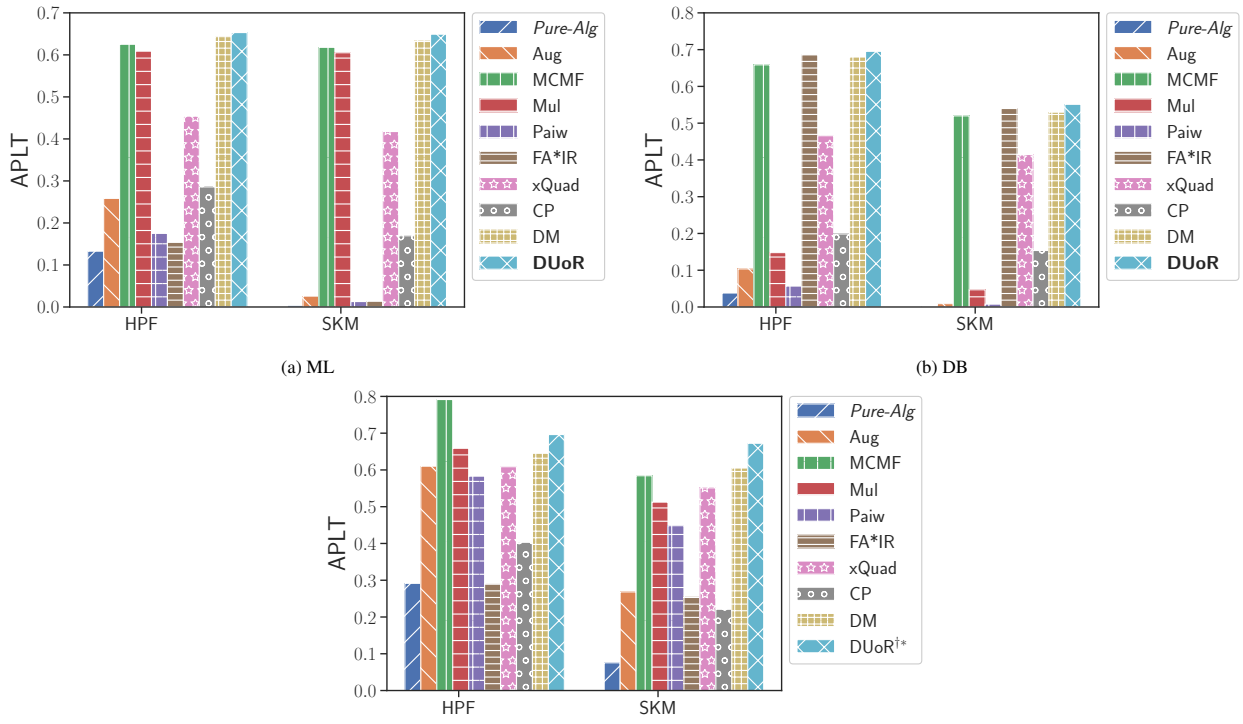
(e) Yelp (f) Yelp

Figure 4: *F1-score* and *n*DCG results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a-b) ML, (c-d) DB, and (e-f) Yelp datasets. (The improvements for corresponding metric compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

Fig. 6. Although it ranks second in *Entropy* behind the *Mul* method for the ML and DB datasets, it achieves the highest *Entropy* values for both algorithms on the Yelp dataset. These results highlight *DUoR*'s effectiveness as a popularity-debiasing method, offering greater opportunities for tail items to appear in recommendations and ensuring fairer representation across the catalog.

In addition, *DUoR* achieves promising *Novelty* scores, consistently ranking among the top-4 best-performing algorithms, as seen in Fig. 7. In conclusion, adopting *DUoR* over other methods can provide system providers with significant improvements in delivering high-quality recommendations that offer greater diversity and acceptable levels of novelty.

As a final comparison of the popularity-debiasing methods, we present their *LTC* results in Fig. 8. For the



(a) ML (b) DB

(c) Yelp

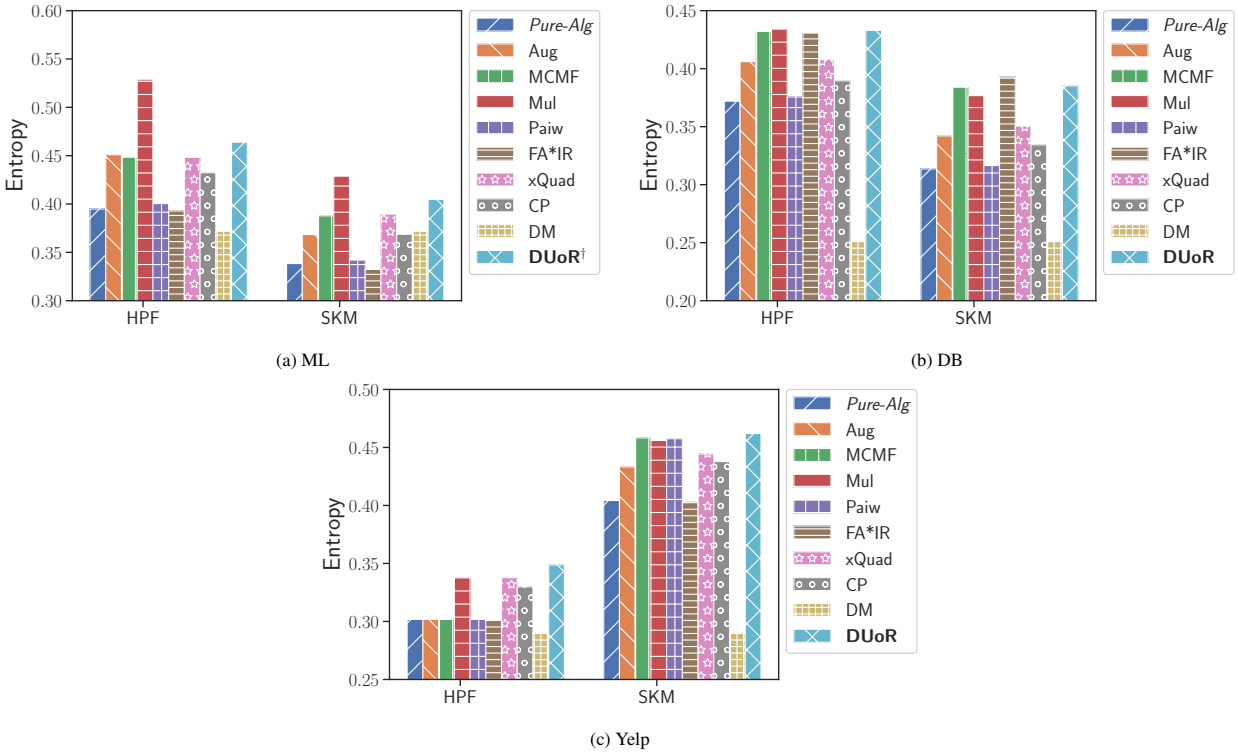
Figure 5: *APLT* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets. (*APLT* improvements compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

ML dataset, *MCMF* consistently achieves the highest *LTC* scores, with significant gaps compared to other methods. However, this advantage is less pronounced for the other two datasets. It's important to note that *MCMF* performs poorly in terms of accuracy (see Figs. 3 and 4), suggesting that its lower accuracy may contribute to its superior catalog coverage. In contrast, the proposed *DUoR* generally ranks around fourth or fifth when popularity-debiasing methods are sorted by their *LTC* scores. While *DUoR* achieves comparable *LTC* values to most methods (excluding *MCMF*), it is not as strong in catalog coverage as it is in the other metrics previously discussed. It is also worth noting that the range of *LTC* values for the ML dataset is relatively larger than that of the other two datasets. This can be attributed to the fact that ML has a smaller number of items, which results in higher *LTC* values for the recommendations.

5.1. Overall evaluation of the popularity-debiasing methods

In the experiments, we compare the proposed *DUoR* against benchmark methods regarding many recommendation qualities, such as calibration, accuracy, diversity, coverage, and novelty. However, we observe in the previous section that the popularity-debiasing methods can show different performances for such aspects, which makes it challenging to infer which method is the best-performing overall. To provide an overall ranking of the popularity-debiasing methods by considering all these recommendation aspects as a whole, we perform additional experiments relying on the *Borda Count* (Yalcin et al., 2021), which is a highly effective strategy in determining the best among a set of candidates by combining their *Borda Ranking Scores (BRSs)* generated for each considered criterion.

The *Borda Count* method is a well-established voting system used to rank a set of candidates by aggregating their rankings across multiple criteria (Regenwetter, 2009). It assigns a score to each candidate based on their ranking position for each criterion. Specifically, for each criterion, the top-ranked method receives the highest score (equal to the total number of methods), while lower-ranked methods receive progressively smaller scores. In our case, we use this approach to combine the rankings across multiple evaluation metrics, such as accuracy, diversity, coverage, and novelty. By aggregating these scores, we generate an overall ranking that reflects the comprehensive performance



(a) ML (b) DB

(c) Yelp

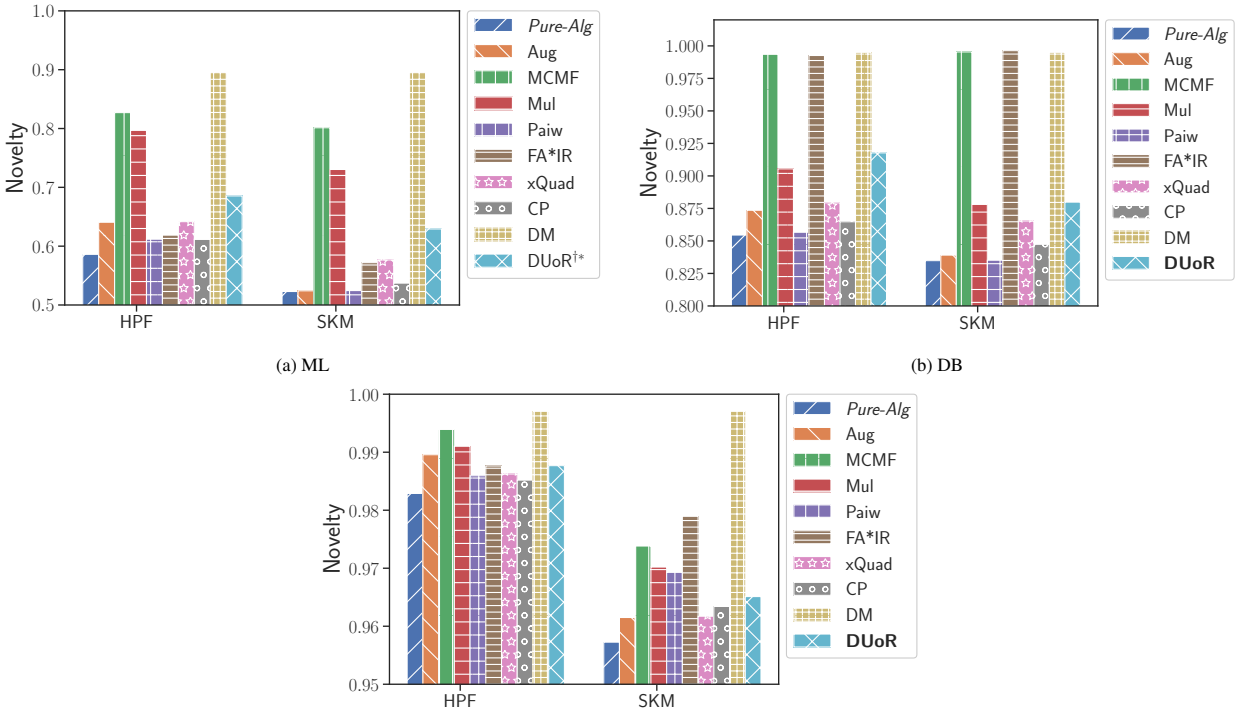
Figure 6: *Entropy* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets. (*Entropy* improvement compared to the next best-performing debiasing technique, with significance at 95%, is marked with † for SKM.)

of each method, allowing us to determine the best-performing one. This ensures that conflicting results in individual metrics are balanced out, providing a fair comparison of methods.

We adopt the *Borda Count* strategy in the experiments as in the following: the popularity-debiasing methods are sorted in descending order based on their achieved results for a particular evaluation metric, and the method at the top of this list indicates the method showing the best performance, and its *BRS* is assigned as '1'. Then, the next method in the list refers to the second best-performing one, and its *BRS* is assigned as '2', and so on. In the case of ties, however, the average ranks of corresponding methods are assigned. Here, note that the methods are sorted in ascending order if lower values are more desired for the considered metric. In our experiments, the only two metrics for this case are *Rmse-PC* and *MRMC*. Once these processes are performed for each considered metric separately, the *Final Borda Scores (FBSs)* for the popularity-debiasing methods are computed by summing their *BRSs* determined for each metric. Finally, the methods are sorted in ascending order based on their computed *FBSs*; thus, the method at the top of this list is the best-performing one overall, and its *FinalRank* is assigned as '1'; the second method in the list is considered second-best performing one overall and its *FinalRank* is as assigned '2', and so on. Under these settings, we present the computed *BRSs*, *FBSs*, and *FinalRanks* of the popularity-debiasing methods for each considered case for ML, DB, and Yelp datasets in Tables 5, 6, and 7, respectively.

As can be followed by *BRS* values in Tables [5 - 7], the proposed *DUoR* puts on the first place in terms of the considered metric for many cases (see cases highlighted with ⋆). The obtain findings conclude that the superiority of the *DUoR* is particularly noticeable in the case of Yelp, and across the metrics *Rmse-PC*, *MRMC*, *APLT*, and *Entropy*. These results highlight the model's effectiveness in delivering more calibrated and diverse recommendations, as previously discussed. More importantly, the computed *FinalRank* values clearly demonstrate that *DUoR* outperforms all other popularity debiasing methods across all scenarios, irrespective of the dataset or collaborative filtering algorithm used. Its superiority is particularly evident for the Yelp dataset, as reflected in both the overall *FinalRank* and the *BRSs* calculated for each individual metric.

Therefore, as an overall evaluation, we can conclude that the proposed *DUoR* is highly beneficial in producing



(a) ML

(b) DB

(c) Yelp

Figure 7: *Novelty* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets. (*Novelty* improvements compared to the next best-performing debiasing technique, with significance at 95%, are marked with $*$ for HPF and † for SKM.)

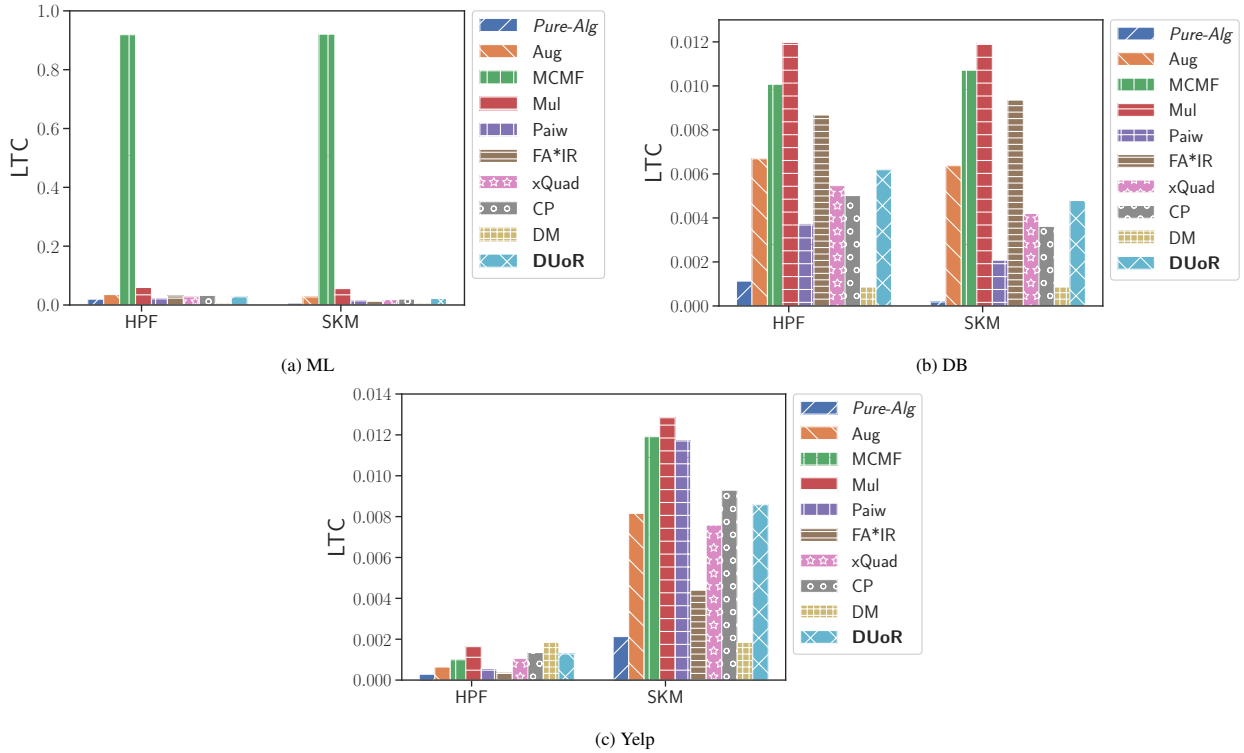
more calibrated and diverse recommendations. Also, when many aspects of the recommendations are considered simultaneously, the proposed *DUoR* performs relatively better overall than other benchmark popularity-debiasing methods, especially for sparser and larger datasets resembling more realistic scenarios of real-world applications. In a nutshell, we highlight the superiority of the proposed *DUoR* method regarding the following concrete aspects:

1. While our *DUoR* exhibits lower accuracy compared to most debiasing methods for the ML and DB datasets, it achieves comparable accuracy with other methods on the Yelp dataset. In fact, it outperforms all others in terms of the *Precision* metric when applied with both the *HPF* and *SKM* algorithms.

2. Our *DUoR* excels in minimizing error when aligning with users' original propensity towards item popularity, as also reflected by the proposed *Rmse-PC* metric (see Fig. 1), which is a core motivation of this study. Notably, *DUoR* surpasses all other methods on the Yelp dataset using both the *HPF* and *SKM* algorithms. Furthermore, the *MRMC* results underscore *DUoR* as one of the best-performing debiasing methods in delivering highly calibrated recommendations.

3. Our *DUoR* outperforms all other debiasing methods in terms of *APLT* across all scenarios and *Entropy* for the Yelp dataset, demonstrating its effectiveness in delivering not only calibrated recommendations but also in mitigating popularity bias, especially by incorporating a wider range of items from the tail set in the final recommendations.

4. It consistently stands out as the best-performing method in the overall evaluation when accuracy, beyond-accuracy, and calibration metrics are given equal importance.



(a) ML

(b) DB

(c) Yelp

Figure 8: *LTC* results of the popularity-debiasing methods for the *HPF* and *SKM* algorithms on the (a) ML, (b) DB, and (c) Yelp datasets.

5. It is the best-performing method in more realistic large and sparse datasets, especially for Yelp.

6. Limitations

While the *DUoR* approach demonstrates promising performance in offline experiments, several potential limitations may arise, particularly when transitioning from controlled offline environments to dynamic real-world settings. One possible limitation could occur in highly dynamic domains, such as music streaming, e-commerce, or social media, where products or items might rapidly gain or lose popularity as their ratings fluctuate. In these environments, items may experience swift changes in popularity due to transient trends, viral content, or seasonal spikes, which could challenge *DUoR*'s static classification system. Since *DUoR* relies on an offline Pareto-based classification to determine popular and niche items, its ability to make optimal recommendations might diminish when items shift categories quickly. The static nature of the model may not fully capture these real-time shifts, potentially leading to suboptimal recommendations in fast-paced environments where item popularity is in constant flux.

Additionally, the increasing adoption of privacy-preserving recommendation systems could pose another limitation for the *DUoR* method. These systems are designed to protect user data by obfuscating or encrypting individual interactions and ratings, which may restrict access to crucial information about user-item interactions. This lack of visibility into specific interactions could make it more challenging to accurately classify items as popular or niche, as *DUoR* depends on such data for its popularity-based classifications. Without access to precise user behavior data, the initial Pareto-based classification step of *DUoR* might become less reliable, potentially hindering its performance in privacy-preserved environments.

Moreover, cold-start scenarios might present additional challenges for *DUoR*, affecting both new users and items. For new users, *DUoR* relies on historical data to infer preferences towards popular and niche items. However, without sufficient interaction history, the system may struggle to personalize recommendations effectively, resulting in less

Recommendation Algorithm	Debiasing Method	*BRS*										*FBS*	***Final Rank***
		Rmse-PC	*MRMC*	*Precision*	*Recall*	*F1-score*	*APLT*	*Novelty*	*nDCG*	*LTC*	*Entropy*		
HPF	*Pure-Alg*	9	8	1	1	1	10	10	1	9	8	58	7
	Aug	6	6	5	5	4	7	6	6	3	3	51	5
	MCMF	4	7	9	9	9	3	2	9	1	4	57	6
	Mul	3	5	8	8	8	4	3	8	2	1	50	4
	Paiw	10	9	2	2	2	8	8	2	8	7	58	8
	*FA*IR*	8	10	4	6	6	9	7	3	5	9	67	9
	xQuad	5	2	6	4	5	5	5	5	7	5	49	3
	CP	7	3	3	3	3	6	9	4	4	6	48	2
	DM	1	4	10	10	10	2	1	10	10	10	68	10
	DUoR	**2**	1*	**7**	**7**	**7**	1*	**4**	**7**	**6**	**2**	**44**	(1)
SKM	*Pure-Alg*	10	10	2	2	2	10	10	1	10	9	66	9
	Aug	8	7	1	1	1	7	9	2	3	7	46	3
	MCMF	4	6	9	9	9	3	2	9	1	4	56	6
	Mul	1	5	8	8	8	4	3	8	2	1	48	4
	Paiw	9	9	3	3	3	9	8	3	7	8	62	7
	*FA*IR*	7	8	6	7	7	8	6	6	8	10	73	10
	xQuad	5	1	5	5	5	5	5	5	6	3	45	2
	CP	6	3	4	4	4	6	7	4	5	6	49	5
	DM	2	4	10	10	10	2	1	10	9	5	63	8
	DUoR	**3**	**2**	**7**	**6**	**6**	1*	**4**	**7**	**4**	**2**	**42**	(1)

$\star$: The case where the ***DUoR*** best-performs for the corresponding metric

Table 5: Computed *FinalRanks* of the debiasing methods via the *Borda Count* strategy for the ML dataset

accurate and possibly suboptimal suggestions. Similarly, for new or recently introduced items, the lack of user ratings or interactions may make it difficult for *DUoR* to classify them accurately. This dual cold-start problem might limit the model's ability to offer personalized recommendations for both new users and new items until sufficient data is collected.

In summary, while *DUoR* shows potential, several challenges remain. Adapting to dynamic environments with rapidly shifting item popularity is a key issue, as static classifications may limit recommendation accuracy. The cold-start problem for new users and items also affects personalization. Additionally, operating in privacy-preserving systems, where access to detailed user interactions is restricted, poses difficulties in accurately classifying items. Addressing these limitations can improve *DUoR*'s effectiveness in real-world applications.

7. Conclusion and possible future directions

This paper addresses the popularity bias problem in recommender systems, which can result in a homogenized user experience and perpetuate existing inequalities. We propose an efficient post-processing treatment method that can be applied to the output of any recommendation approach to mitigate the popularity bias. Our method incorporates a practical popularity inclination measuring strategy better to capture users' unique propensities towards item popularity, leading to more calibrated recommendations. We also introduce a novel popularity-aware evaluation metric to measure the effectiveness of our method. Experimental results on three benchmark datasets demonstrate that our proposed method significantly mitigates the popularity bias and provides better user-oriented recommendations than existing benchmark methods.

Our proposed method has important implications for improving the user experience in recommender systems, as it can provide more diverse and personalized recommendations that align with users' original propensities towards item popularity. By mitigating the popularity bias, our method can promote a broader range of items, including less popular but potentially more relevant items, leading to a more diverse information landscape. Furthermore, our method can help reduce existing inequalities by providing recommendations not solely based on popularity but also considering users' unique preferences toward popular items.

Furthermore, we conducted additional experiments using the Borda Count strategy to provide an overall ranking of the popularity-debiasing methods by considering multiple recommendation qualities such as accuracy, diversity, coverage, and novelty. The computed Borda Ranking Scores and Final Ranks demonstrate that the proposed

Recommendation Algorithm	Debiasing Method	*BRS*										*FBS*	***Final Rank***
		Rmse-PC	*MRMC*	*Precision*	*Recall*	*F1-score*	*APLT*	*Novelty*	*nDCG*	*LTC*	*Entropy*		
HPF	*Pure-Alg*	10	6	1	1	1	10	10	1	9	9	58	8
	Aug	7	4	4	6	4	8	7	4	4	6	54	5
	MCMF	5	8	9	9	9	4	2	9	2	3	60	9
	Mul	3	7	7	7	7	7	5	7	1	1	52	4
	Paiw	9	5	2	2	2	9	9	2	8	8	56	6
	*FA*IR*	2	10	8	8	8	2	3	8	3	4	56	7
	xQuad	6	1	5	4	5	5	6	5	6	5	48	2
	CP	8	3	3	3	3	6	8	3	7	7	51	3
	DM	1	9	10	10	10	3	1	10	10	10	74	10
	DUoR	**4**	**2**	**6**	**5**	**6**	1*	**4**	**6**	**5**	**2**	**41**	(1)
SKM	*Pure-Alg*	10	8	1	1	1	10	10	1	10	9	61	9
	Aug	7	7	3	2	3	8	8	3	4	6	51	4
	MCMF	5	6	9	9	9	4	2	9	2	3	58	8
	Mul	3	4	6	7	7	7	5	7	1	4	51	5
	Paiw	9	5	2	3	2	9	9	2	8	8	57	7
	*FA*IR*	2	9	8	8	8	2	1	8	3	1	50	2
	xQuad	6	1	5	6	5	5	6	5	6	5	50	3
	CP	8	3	4	4	4	6	7	4	7	7	54	6
	DM	1	10	10	10	10	3	3	10	9	10	76	10
	DUoR	**4**	**2**	**7**	**5**	**6**	1*	**4**	**6**	**5**	**2**	**42**	(1)

$\star$: The case where the ***DUoR*** best-performs for the corresponding metric

Table 6: Computed *FinalRanks* of the debiasing methods via the *Borda Count* strategy for the DB dataset

popularity-debiasing strategy outperforms other benchmark methods in many cases, indicating its superiority in providing calibrated and diverse recommendations. In conclusion, our proposed method contributes to the ongoing efforts to address the popularity bias problem in recommender systems and provides a practical and effective solution to improve the user-orientedness of recommendations.

However, several future directions remain to be explored, especially to address the limitations identified during this study. Key areas for future research include:

- *Trend detection and response*: Future work could explore incorporating online learning algorithms or adaptive mechanisms to update the Pareto-based classification and user profiles in real time. This would address the challenge of rapidly shifting item popularity in domains such as music streaming, e-commerce, and social media, ensuring that the system remains accurate and responsive in dynamic environments.

- *Handling privacy-preserving systems*: As privacy-preserving recommendation systems continue to grow, future work could focus on developing ways for *DUoR* to function effectively without direct access to user-item interactions and ratings. Approaches such as synthetic data generation, where anonymized data can be simulated based on statistical patterns, or collaborative filtering with aggregated, non-identifiable user data could allow the system to maintain recommendation quality while respecting privacy constraints. Additionally, techniques that infer item popularity from implicit feedback (e.g., clicks, browsing time) instead of explicit ratings might also be explored to overcome the lack of detailed interaction data.

- *Cold-start scenarios*: To enhance performance in cold-start scenarios—both for new users and new items—future research could investigate the integration of hybrid approaches or transfer learning models. These methods could help provide more personalized recommendations in early interactions, even when little or no historical data is available, thereby improving the system's effectiveness in sparse data environments.

In conclusion, *DUoR* contributes meaningfully to the mitigation of popularity bias in recommender systems. By addressing its current limitations and expanding the proposed future research directions, further developments in this field can lead to more user-centric and adaptive recommendation systems.

Recommendation Algorithm	Debiasing Method	*BRS*										*FBS*	***Final Rank***
		Rmse-PC	*MRMC*	*Precision*	*Recall*	*F1-score*	*APLT*	*Novelty*	*nDCG*	*LTC*	*Entropy*		
HPF	*Pure-Alg*	10	9	1	2	1	10	10	1	10	9	63	8
	Aug	7	6	7	7	7	5	4	7	7	8	65	9
	MCMF	2	2	8	8	8	1	2	8	6	7	52	4
	Mul	5	8	9	9	9	3	3	9	2	2	59	5
	Paiw	9	7	2	5	4	7	8	4	8	6	60	6
	*FA*IR*	6	10	5	6	6	9	6	6	9	5	68	10
	xQuad	3	4	4	3	3	6	7	2	5	3	40	2
	CP	8	5	3	4	2	8	9	3	3	4	49	3
	DM	4	1	10	10	10	4	1	10	1	10	61	7
	DUoR	1⋆	**3**	**6**	1⋆	**5**	**2**	**5**	**5**	**4**	1⋆	**33**	**(1)**
SKM	*Pure-Alg*	10	10	1	3	2	10	10	3	9	8	66	8
	Aug	8	6	2	4	4	7	9	4	6	7	57	4
	MCMF	5	7	9	9	9	3	3	9	2	2	58	5
	Mul	4	4	7	6	6	5	4	6	1	4	47	3
	Paiw	7	9	6	7	7	6	5	7	3	3	60	7
	*FA*IR*	6	8	8	8	8	8	2	8	8	9	73	10
	xQuad	3	3	3	2	1	4	8	1	7	5	37	2
	CP	9	5	4	5	5	9	7	5	4	6	59	6
	DM	2	1	10	10	10	2	1	10	10	10	66	9
	DUoR	1⋆	**2**	**5**	1⋆	**3**	1⋆	**6**	**2**	**5**	1⋆	**27**	**(1)**

⋆: The case where the ***DUoR*** best-performs for the corresponding metric

Table 7: Computed *FinalRanks* of the debiasing methods via the *Borda Count* strategy for the Yelp dataset

Acknowledgment

This study is supported in part by the Scientific and Technical Research Council of Turkey (TUBITAK) under grant number 122E040.

References

Abdollahpouri, H., 2020. Popularity bias in recommendation: A multi-stakeholder perspective. Ph.D. thesis. University of Colorado at Boulder.

Abdollahpouri, H., Burke, R., Mobasher, B., 2017. Controlling popularity bias in learning-to-rank recommendation, in: Proceedings of the eleventh ACM conference on recommender systems, pp. 42–46.

Abdollahpouri, H., Burke, R., Mobasher, B., 2018. Popularity-aware item weighting for long-tail recommendation. arXiv preprint arXiv:1802.05382 .

Abdollahpouri, H., Burke, R., Mobasher, B., 2019a. Managing popularity bias in recommender systems with personalized re-ranking. arXiv preprint arXiv:1901.07555 .

Abdollahpouri, H., Mansoury, M., Burke, R., Mobasher, B., 2019b. The impact of popularity bias on fairness and calibration in recommendation. arXiv preprint arXiv:1910.05755 .

Abdollahpouri, H., Mansoury, M., Burke, R., Mobasher, B., 2019c. The unfairness of popularity bias in recommendation. arXiv preprint arXiv:1907.13286 .

Abdollahpouri, H., Mansoury, M., Burke, R., Mobasher, B., Malthouse, E., 2021. User-centered evaluation of popularity bias in recommender systems, in: Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization, pp. 119–129.

Abdollahpouri, H., Nazari, Z., Gain, A., Gibson, C., Dimakopoulou, M., Anderton, J., Carterette, B., Lalmas, M., Jebara, T., 2023. Calibrated recommendations as a minimum-cost flow problem, in: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, pp. 571–579.

Ahanger, A.B., Aalam, S.W., Bhat, M.R., Assad, A., 2022. Popularity bias in recommender systems-a review, in: Emerging Technologies in Computer Engineering: Cognitive Computing and Intelligent IoT: 5th International Conference, ICETCE 2022, Jaipur, India, February 4–5, 2022, Revised Selected Papers, Springer. pp. 431–444.

Antikacioglu, A., Ravi, R., 2017. Post processing recommender systems for diversity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 707–716.

Bobadilla, J., Serradilla, F., Bernal, J., 2010. A new collaborative filtering metric that improves the behavior of recommender systems. Knowledge-Based Systems 23, 520–528.

Boratto, L., Fenu, G., Marras, M., 2019. The effect of algorithmic bias on recommender systems for massive open online courses, in: Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41, Springer. pp. 457–472.

Boratto, L., Fenu, G., Marras, M., 2021. Connecting user and item perspectives in popularity debiasing for collaborative recommendation. Information Processing & Management 58, 102387.

Borges, R., Stefanidis, K., 2021. On mitigating popularity bias in recommendations via variational autoencoders, in: Proceedings of the 36th annual ACM symposium on applied computing, pp. 1383–1389.

Chen, C., Zhang, M., Liu, Y., Ma, S., 2018. Missing data modeling with user activity and item popularity in recommendation, in: Information Retrieval Technology: 14th Asia Information Retrieval Societies Conference, AIRS 2018, Taipei, Taiwan, November 28-30, 2018, Proceedings 14, Springer. pp. 113–125.

Deldjoo, Y., Jannach, D., Bellogin, A., Difonzo, A., Zanzonelli, D., 2024. Fairness in recommender systems: research landscape and future directions. User Modeling and User-Adapted Interaction 34, 59–108.

Dong, Q., Xie, S.S., Li, W.J., 2021. User-item matching for recommendation fairness. IEEE Access 9, 130389–130398.

Elahi, M., Kholgh, D.K., Kiarostami, M.S., Saghari, S., Rad, S.P., Tkalčič, M., 2021. Investigating the impact of recommender systems on user-based and item-based popularity bias. Information Processing & Management 58, 102655.

Fabbri, F., Croci, M.L., Bonchi, F., Castillo, C., 2022. Exposure inequality in people recommender systems: The long-term effects, in: Proceedings of the International AAAI Conference on Web and Social Media, pp. 194–204.

Gopalan, P., Hofman, J.M., Blei, D.M., 2015. Scalable recommendation with hierarchical poisson factorization., in: UAI, pp. 326–335.

Gulsoy, M., Yalcin, E., Bilge, A., 2023. Robustness of privacy-preserving collaborative recommenders against popularity bias problem. PeerJ Computer Science 9, e1438.

Hou, L., Pan, X., Liu, K., 2018. Balancing the popularity bias of object similarities for personalised recommendation. The European Physical Journal B 91, 1–7.

Jannach, D., Kamehkhosh, I., Bonnin, G., 2016. Biases in automated music playlist generation: A comparison of next-track recommending techniques, in: Proceedings of the 2016 conference on user modeling adaptation and personalization, pp. 281–285.

Jannach, D., Lerche, L., Kamehkhosh, I., Jugovac, M., 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. User Modeling and User-Adapted Interaction 25, 427–491.

Kamishima, T., Akaho, S., Asoh, H., Sakuma, J., 2014. Correcting popularity bias by enhancing recommendation neutrality. RecSys Posters 805.

Klimashevskaia, A., Jannach, D., Elahi, M., Trattner, C., 2024. A survey on popularity bias in recommender systems. User Modeling and User-Adapted Interaction 34, 1777–1834.

Kowald, D., Schedl, M., Lex, E., 2020. The unfairness of popularity bias in music recommendation: A reproducibility study, in: Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42, Springer. pp. 35–42.

Lesota, O., Melchiorre, A., Rekabsaz, N., Brandl, S., Kowald, D., Lex, E., Schedl, M., 2021. Analyzing item popularity bias of music recommender systems: are different genders equally affected?, in: Proceedings of the 15th ACM Conference on Recommender Systems, pp. 601–606.

Li, L., Chen, J., Raghunathan, S., 2018. Recommender system rethink: Implications for an electronic marketplace with competing manufacturers. Information Systems Research 29, 1003–1023.

Liu, Q., Zhang, X., Zhang, L., Zhao, Y., 2019. The interaction effects of information cascades, word of mouth and recommendation systems on online reading behavior: An empirical investigation. Electronic Commerce Research 19, 521–547.

Liu, Z., Fang, Y., Wu, M., 2023. Mitigating popularity bias for users and items with fairness-centric adaptive recommendation. ACM Transactions on Information Systems 41, 1–27.

Naghiaei, M., Rahmani, H.A., Dehghan, M., 2022a. The unfairness of popularity bias in book recommendation, in: Advances in Bias and Fairness in Information Retrieval: Third International Workshop, BIAS 2022, Stavanger, Norway, April 10, 2022, Revised Selected Papers, Springer. pp. 69–81.

Naghiaei, M., Rahmani, H.A., Deldjoo, Y., 2022b. Cpfair: Personalized consumer and producer fairness re-ranking for recommender systems, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 770–779.

Park, Y.J., Tuzhilin, A., 2008. The long tail of recommender systems and how to leverage it, in: Proceedings of the 2008 ACM conference on Recommender systems, pp. 11–18.

Regenwetter, M., 2009. Perspectives on preference aggregation. Perspectives on Psychological Science 4, 403–407.

Ricci, F., Rokach, L., Shapira, B., 2015. Recommender systems: introduction and challenges. Recommender systems handbook , 1–34.

Salah, A., Rogovschi, N., Nadif, M., 2016. A dynamic collaborative filtering system via a weighted clustering approach. Neurocomputing 175, 206–215.

Salah, A., Truong, Q.T., Lauw, H.W., 2020. Cornac: A comparative framework for multimodal recommender systems. J. Mach. Learn. Res. 21.

Sánchez, P., 2019. Exploiting contextual information for recommender systems oriented to tourism, in: Proceedings of the 13th ACM conference on recommender systems, pp. 601–605.

Sanders, R., 1987. The pareto principle: its use and abuse. Journal of Services Marketing 1, 37–40.

Seymen, S., Abdollahpouri, H., Malthouse, E.C., 2021. A unified optimization toolbox for solving popularity bias, fairness, and diversity in recommender systems., in: MORS@ RecSys.

da Silva, D.C., Manzato, M.G., Durão, F.A., 2021. Exploiting personalized calibration and metrics for fairness recommendation. Expert Systems with Applications 181, 115112.

Steck, H., 2018. Calibrated recommendations, in: Proceedings of the 12th ACM conference on recommender systems, pp. 154–162.

Su, X., Li, P., Zhu, X., 2023. The influence of herd mentality on rating bias and popularity bias: A bi-process debiasing recommendation model based on matrix factorization. Behavioral Sciences 13, 63.

Tacli, Y., Yalcin, E., Bilge, A., 2022. Novel approaches to measuring the popularity inclination of users for the popularity bias problem, in: 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), IEEE. pp. 555–560.

Wong, T.T., 2015. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. Pattern Recognition 48, 2839–2846.

Yalcin, E., Bilge, A., 2021. Investigating and counteracting popularity bias in group recommendations. Information Processing & Management 58, 102608.

Yalcin, E., Bilge, A., 2022a. Evaluating unfairness of popularity bias in recommender systems: A comprehensive user-centric analysis. Information Processing & Management 59, 103100.

Yalcin, E., Bilge, A., 2022b. Treating adverse effects of blockbuster bias on beyond-accuracy quality of personalized recommendations. Engineering Science and Technology, an International Journal 33, 101083.

Yalcin, E., Bilge, A., 2023. Popularity bias in personality perspective: An analysis of how personality traits expose individuals to the unfair recommendation. Concurrency and Computation: Practice and Experience 35, e7647.

Yalcin, E., Ismailoglu, F., Bilge, A., 2021. An entropy empowered hybridized aggregation technique for group recommender systems. Expert Systems with Applications 166, 114111.

Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., Baeza-Yates, R., 2017. Fa* ir: A fair top-k ranking algorithm, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1569–1578.

Zhang, Y., Feng, F., He, X., Wei, T., Song, C., Ling, G., Zhang, Y., 2021. Causal intervention for leveraging popularity bias in recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 11–20.