Efficient and Responsible Adaptation of Large Language Models for Robust Top-k Recommendations*

Kirandeep Kaur[1], Chirag Shah[1]

[1]*University of Washington, Seattle, WA*

Abstract

Conventional recommendation systems (RSs) are typically optimized to enhance performance metrics uniformly across all training samples. This makes it hard for data-driven RSs to cater to a diverse set of users due to the varying properties of these users. The performance disparity among various populations can harm the model's robustness with respect to sub-populations. While recent works have shown promising results in adapting large language models (LLMs) for recommendation to address hard samples, long user queries from millions of users can degrade the performance of LLMs and elevate costs, processing times and inference latency. This challenges the practical applicability of LLMs for recommendations. To address this, we propose a hybrid task allocation framework that utilizes the capabilities of both LLMs and traditional RSs. By adopting a two-phase approach to improve robustness to sub-populations, we promote a strategic assignment of tasks for efficient and responsible adaptation of LLMs. Our strategy works by first identifying the weak and inactive users that receive a suboptimal ranking performance by RSs. Next, we use an in-context learning approach for such users, wherein each user interaction history is contextualized as a distinct ranking task and given to an LLM. We test our hybrid framework by incorporating various recommendation algorithms – collaborative filtering and learning-to-rank recommendation models – and two LLMs – both open and close-sourced. Our results on three real-world datasets show improved robustness of RSs to sub-populations ($\approx 12\%$) and overall performance without disproportionately escalating costs.

Keywords

Recommender Systems, Large Language Models, Robustness, Responsible AI

1. Introduction

Recommendation systems (RSs) have become an integral part of numerous online platforms, assisting users in navigating vast amounts of content to relieve information overload [1]. While Collaborative Filtering based RSs [2] primarily rely on user-item interactions to predict users' preferences for certain candidate items, the utilization of language in recommendations has been prevalent for decades in hybrid and content-based recommenders, mainly through item descriptions and text-based reviews [3]. Furthermore, conversational recommenders [4] have highlighted language as a primary mechanism for allowing users to naturally and intuitively express their preferences [5]. Deep recommendation models are trained under the Empirical Risk Minimization (ERM) framework that minimizes the loss function uniformly for all training samples. Such models, however, fail to cater to a diverse set of sub-populations, affecting robustness [6, 7, 8, 9, 10, 11, 12]. Empirical analysis conducted by Li et al. [13] shows that active users who have rated many items receive better recommendations on average than inactive users. This inadvertent disparity in recommendations requires careful scrutiny to ensure equitable recommendation experiences for all users [14].

On the other hand, Large Language Models (LLMs) like GPT [15], LLaMA [16], LaMDA [17], Mixtral [18] can effectively analyze and interpret textual data, thus enabling a better understanding of user preferences. These foundation models demonstrate remarkable versatility, adeptly tackling various tasks across multiple domains [19, 20, 21]. However, the field of recommendations is highly domain-specific and requires in-domain knowledge. Consequently, many researchers have sought to adapt LLMs for recommendation tasks [22, 23, 24, 25]. Authors in [25] outline four key stages in integrating LLMs into the recommendation pipeline: user interaction, feature encoding, feature engineering, and scoring/ranking. The purpose of using LLMs as a ranking function aligns closely with general-purpose recommendation models. The transition from traditional library-based book searches to evaluating various products, job applicants, opinions, and potential romantic partners signifies an important societal transformation, emphasizing the considerable responsibility incumbent upon ranking systems [26]. Existing works that deploy LLMs for ranking [5, 27, 28, 29, 30, 31, 32, 33, 34] have proven excellence of LLMs as zero-shot or few-shot re-rankers demonstrating their capabilities in re-ranking with frozen parameters. These works use traditional RSs as candidate item retrieval models to limit the candidate items that need to be ranked by LLM due to a limited context window. Furthermore, Xu et al. [28], Hou et al. [27] interpret user interaction histories as prompts for LLMs and show that LLMs perform well only when the interaction length is up to a few items, demonstrating the ability of LLMs for (near) cold-start users. Since adapting LLMs can raise concerns around economic and efficiency factors, most of these works train RS on entire datasets but randomly sample interaction histories of some users to evaluate the performance of LLMs, questioning the generalizability of results for all users. This leads us to two important research questions.

- **RQ1**: Though LLMs have shown remarkable ranking performance even in zero-shot settings, how can we reduce the high costs associated with adapting LLMs to support practical applicability?
- **RQ2**: Conventional recommendation systems are cost-effective and can perform well on most users, as shown by previous works; how can we prevent performance degradation on sub-populations?

To address these RQs, we propose a task allocation strategy that leverages LLM and RS's capabilities in a hybrid framework (Fig. 1). Our strategy operates in two phases based on the responsible and strategic selection of tasks for the cost-effective usage of LLMs. First, we identify the

The 1st Workshop on Risks, Opportunities, and Evaluation of Generative Models in Recommender Systems (ROEGEN@RecSys 2024), October 2024, Bari, Italy.

kaur13@cs.washington.edu (K. Kaur); chirags@uw.edu (C. Shah)
https://i-kiran.github.io/ (K. Kaur); https://chiragshah.org/ (C. Shah)
0009-0000-7128-1792 (K. Kaur); 0000-0002-3797-4293 (C. Shah)

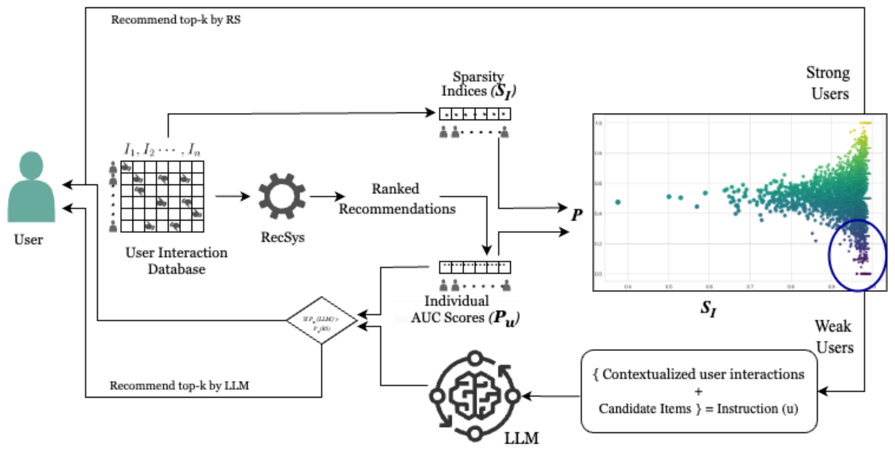


Figure 1: An overview of our framework that uses task allocation to adapt LLMs responsibly. We compute each user's sparsity index (S_I), evaluate recommendations retrieved from RS using performance metric ($P(u_m)$), and plot $P(u_m)$ against S_I. Interaction histories of highly sparse users with low $P(u_m)$ are contextualized and given to LLM for ranking. Strong users receive RS recommendations, while weak users get LLM recommendations if LLM outperforms RS.

users with highly sparse interaction histories on whom the ranking performance of RS is below a certain threshold t_p. All such users are termed as weak users. In the second phase, interaction histories of weak users are contextualized using in-context learning to demonstrate user preferences as instruction inputs for LLM. While the strong users receive the final recommendations retrieved by RS, weak users receive the recommendations ranked by LLM if the quality of the ranked list is better than the RS. We test our framework based on collaborative filtering and learning-to-rank recommendation models and our results show the efficacy of our strategy, both with open-source as well as closed-source LLMs, in boosting the model robustness to sub-population and data sparsity and improving the quality of recommendations. For reproducibility and to support research community, our code is available on https://anonymous.4open.science/r/resp-llmsRS/ and the link to the video is https://youtu.be/uqriMZHL-Ng In short, the following are our contributions in this paper.

- We introduce a **novel hybrid task allocation strategy** that combines the strengths of LLMs and traditional RSs to improve robustness to subpopulations and data sparsity.
- Our **unique method** for pinpointing weak users based upon two criteria (user activity and the received recommendation quality below a set threshold) facilitates interventions using LLMs for equitable recommendations.
- Our **proposed framework improves the robustness of traditional recommendation models** by reducing weak user count, enhancing recommendation quality, and addressing high costs associated with adapting LLMs.
- Our **experiments, both on closed-source and open-source LLMs**, show the efficacy of our framework in improving the model robustness to sub-populations by ($\approx 12\%$) for varying levels of sparsity and reducing the count of weak users significantly.

2. Related Work

Robustness in machine learning (ML) targets developing models capable of withstanding the challenges posed by imperfect data in diverse forms [35]. Within the paradigm of recommendations, some existing works developed models resilient to shifts in popularity distribution [36, 37, 38], distribution disparity in train and test datasets [39, 40], adversarial and data poisoning attacks [41, 42, 43, 44, 45]. Our work aims to tackle the recommendation model's robustness to data sparsity [46] and sub-populations [47].

In their research, Li et al. [13] illustrated that RSs excel in catering to active users but fall short in meeting the overall needs of inactive ones. To address this inequality, they proposed a re-ranking technique that reduced the disparity among active and inactive users. Their results depict that such post-processing techniques [48, 49, 50] can either harm the average performance on advantaged users to reduce the disparity or reduce the overall utility of models. Though the in-processing techniques [51, 52, 53] for improving equitable recommendations across various sub-populations can tackle fairness-utility trade-offs, simply adding regularizer term results in sub-optimal performance [54]. Most of these works have shown disparity and evaluated existing models by grouping users based on their activity, demographics, and preferences. Similarly, Wen et al. [55] developed a Streaming-Distributionally Robust Optimization (S-DRO) framework to enhance performance across user subgroups, particularly by accommodating their preferences for popular items. Different from these, our work first builds upon the existing literature that elicits the issue of performance disparities among active and inactive users and then indicates that though inactive users receive lower-quality recommendations on average, this degradation only affects a subset of inactive users rather than all inactive users. Unlike these works, our framework identifies weak users— inactive individuals whose preferences traditional recommendation systems struggle to capture effectively.

Many researchers have turned to LLMs to address some of these problems because, in recent years, LLMs have proven to be excellent re-rankers and have often outperformed existing SOTA recommendation models in zero-shot and few-shot settings without requiring fine-tuning. For example, Gao et al. [56] proposed an enhanced recommender system that integrates ChatGPT with traditional RS by synthesizing user-item history, profiles, queries, and dialogue to provide personalized explanations to the recommendations through iterative refinement based on user feedback. AgentCF [31], designed to rank items for users, involves treating users and items as agents and optimizing their interactions collaboratively. While user agents capture user preferences, item agents reflect item characteristics and potential adopters' preferences. They used collaborative memory-based optimization to ensure agents align better with real-world behaviours. While the retrieval-ranker framework in [29] remains similar to previous works, authors generate instructions with key values obtained from both users (e.g., gender, age, occupation) and items (e.g., title, rating, category).

Despite the excellence of LLMs as ranking agents, adapting LLMs can involve processing lengthy queries containing numerous interactions from millions of users. Furthermore, each query can raise various economic and latency concerns. Thus, all these works randomly select a few users from the original datasets to evaluate the performance of LLMs. In practice, this user base can involve many more users, which questions the practical applicability of large models for recommendations. However, some recent studies have shown the efficacy of large language models (LLMs) as re-ranking agents to cater to queries with shorter interaction histories compared to lengthy instructions that constitute hundreds of interactions.

For example, Hou et al. [27] trained recommendation systems to generate candidate item sets and then used user-item interactions to develop instructions. The authors sorted users' rating histories based on timestamps and used in-context learning to design recency-focused prompts. They prompted LLMs to re-rank the candidate items retrieved by the recommendation systems. Their analysis showed decreased performance of LLMs if the candidate item set had more than 20 items. ProLLM4Rec [28] adopted a unified framework for prompting LLMs for recommendation. The authors integrated existing recommendation systems and works that use LLMs for recommendations within a single framework. They provided a detailed comparison of the capabilities of LLMs and recommendation systems. Their empirical analysis showed that while state-of-the-art sequential recommendation models like SASRec [57] improve with a growing number of interactions, LLMs start to perform worse when the number of interactions grows. Furthermore, both of these works sampled some users to evaluate the performance of LLMs due to the high adaptation costs. To investigate the effectiveness of various prompting strategies Sanner et al. [5] focused on a (near) cold-start scenario where minimal interaction data is available. They used various prompting techniques to provide a natural language summary of preferences to enhance user satisfaction by offering a personalized experience. By exploiting rich positive and negative descriptive content and item preferences within a unified framework, they compared the efficacy of prompting paradigms with large language models against collaborative filtering baselines that rely solely on item ratings.

In summary, past works suggest that despite the high costs associated with adapting LLMs for recommendations, these models can outperform existing recommendation models significantly. Moreover, we acknowledge that the literature shows the contrasting capabilities of both RSs and LLMs – RSs fail to perform well on inactive users due to sparse interaction vectors, and in contrast, LLMs can be prompted to cater to inactive users in near cold-start settings without requiring any fine-tuning.

Building upon these crucial insights, our framework first aims to identify the weak users for whom RS finds it hard to capture their preferences accurately. We then use in-context learning to prompt LLMs to generate recommendations for such users. While past works like ProLLM4Rec by [28], dynamic reflection with divergent thinking within a retriever-reranked by [33], recency-focused prompting by [27] and aligning ChatGPT with conventional ranking techniques such as point-wise, pair-wise, and list-wise ranking by [58] are all different techniques to design prompts with different variations, our main contribution lies in the responsible task allocation within recommendation systems and all such techniques can be used within our framework for designing prompts. In the next section, we discuss our methodology in detail.

3. Methodology

We begin here by providing a formal definition of the existing problem. We then discuss our framework, which adopts a hybrid structure by leveraging the capabilities of both traditional RSs and LLMs. For this, we first identify users for whom RSs do not perform well and then leverage LLMs for these users to demonstrate user preferences using in-context learning.

3.1. Problem Formulation

Consider a recommendation dataset $\mathcal{D}$ with k data points. Let $U = \{u_1, u_2, \ldots, u_M\}$ be the set of users and $|U| = M$ represents the number of users in $\mathcal{D}$. Let $I = \{i_1, i_2, \ldots, i_N\}$ be the set of all the items and $|I| = N$ represents the number of items in $\mathcal{D}$.

$$\mathcal{D} = \{(u_m, i_n, r_{mn}) : m = 1, 2, \ldots, M; n = 1, 2, \ldots, N\} \tag{1}$$

Here, the triplet $d_{mn} = (u_m, i_n, r_{mn})$ represents one data point where a user u_m provided a rating of r_{mn} to an item i_n. Now, if a user u_m has rated a set of items, then let $[r_{mn}]_{n=1}^{N}$ denote the rating vector consisting of explicit rating values ranging from 1 to 5 if a user provided a rating and 0 otherwise. Additionally, θ^r represents the conventional recommendation model. The first step to solving the problem includes determining different criteria to categorize a user as weak. This includes ranking users based on the RS performance on each one of them. Then, the goal is to understand user characteristics to categorize extremely weak users. For each weak user, we contextualize interaction history as a distinct recommendation task and finally allocate these tasks to LLM.

3.2. Identifying Weak Users

We consider two criteria for identifying weak users for recommendation model θ^r. First, given K users and their associated rating vectors, we evaluate how well the model could rank the relevant user items, often termed as *positive items* above the irrelevant or *negative items*. Let r denote the rank of the relevant item, and r' be the rank of irrelevant items. Then,

$$\delta(r < r') \tag{2}$$

denotes an indicator function that outputs one if the rank of the relevant item r is higher than that of the irrelevant item r'. Let N denote the total number of items and $|R|$ be the set of all relevant items. Then, similar to Rendle et al. [59], we use AUC measure to evaluate how hard it was for θ^r to rank items preferred by a certain user, given by

$$\mathcal{P}(\mathrm{u}) = \frac{1}{|R|(N - |R|)} \sum_{r \in R} \sum_{r' \in \{1,\ldots,N\} \setminus R} \delta(r < r') \tag{3}$$

Here, $|R|(n - |R|)$ denotes all possible pairs of relevant and irrelevant items.

We acknowledge that various metrics like NDCG, F1, precision and recall have been used to measure the quality of ranking ability of recommendation models. However, these metrics place significant importance on the outcomes of the top-k items in the list and completely ignore the tail. For identifying weak users, we require a metric consistent under sampling i.e. if a recommendation model tends to give better recommendations than another on average across all the data, it should still tend to do so even if we only look at a smaller part of the data. The aim of our framework is a clear task distribution. The performance of top-k metrics varies with k, and this might raise uncertainty as k varies with varying users and platforms. Nevertheless, AUC is the only metric which remains consistent under-sampling, and as k reduces, all top-k metrics collapse to AUC. For more details, we refer the readers to [60].

Past works [13] have shown that *active* users that provide more ratings receive better recommendations than the *inactive* users on average. However, only a few inactive users might receive irrelevant recommendations individually (Fig. 3). Thus, we evaluate each user's activity. Let a user u rated $|R|$ items out of a total of N items. Then sparsity index $\mathcal{S}_I$ associated with a given user u can be calculated as:

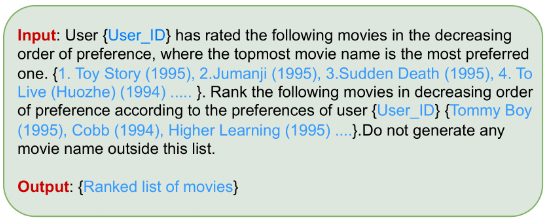
$$\mathcal{S}_I(u) = \frac{|R|}{N} \tag{4}$$

If this value falls above a certain threshold t_s, the user is considered as inactive. Combining with the weak user identification, we obtain,

Definition 1. *Given dataset $\mathcal{D}$ and a recommendation model θ^r, we say that a user u_m is extremely weak if the likelihood of θ^r being able to rank the relevant items above the irrelevant items is below t_p and the rating vector $[r_{mn}]$ has extremely high sparsity. i.e., above t_s*

$$\mathcal{P}(u_m) \leq t_p \quad \&\& \quad \mathcal{S}_I(u) > t_s \tag{5}$$

It is important to note that a higher AUC value implies better performance, and the value always lies between 0 to 1. Further, we use $t_s = avg(\mathcal{S}_I(D))$, the average sparsity of all users in $\mathcal{D}$ i.e.,
$avg(\mathcal{S}_I(D)) = 1/m * \sum_{j=1}^{m} \mathcal{S}_I(u_j)$ for determining this threshold.


Input: User {User_ID} has rated the following movies in the decreasing order of preference, where the topmost movie name is the most preferred one. {1. Toy Story (1995), 2.Jumanji (1995), 3.Sudden Death (1995), 4. To Live (Huozhe) (1994) ..... }. Rank the following movies in decreasing order of preference according to the preferences of user {User_ID} {Tommy Boy (1995), Cobb (1994), Higher Learning (1995) ....}.Do not generate any movie name outside this list.

Output: {Ranked list of movies}

Figure 2: Instruction template for contextualizing interaction histories of weak users.

3.3. Designing Natural Language Instructions for Ranking

Closest to our work, Hou et al. [27] formalized the recommendation problem as a *conditional* ranking task considering sequential interaction histories as conditions and uses the items retrieved by traditional RS as $candidate$ items. While we aim to design the conditional ranking tasks, our approach differs significantly from theirs as instead of using LLMs as a re-ranking agent for all users; we instruct LLM with the preferences of weak users preferences (*sorted in descending order of decreased preference*). This technique is detailed below.

For each user, we use in-context learning to instruct LLM about user preferences *conditions* and assign the task of ranking the *candidate* items. For a user u, let $\mathcal{H}_u = \{i_1, i_2, \ldots, i_n\}$ depict the user interaction histories sorted in decreasing order of preference and $\mathcal{C}_u = \{i_1, i_2, \ldots, i_j\}$ be the candidate items to be ranked. Then, each instruction can be generated as a sum of conditions and candidate items, i.e.,

$$\mathcal{I}_u = \mathcal{H}_u + \mathcal{C}_u \tag{6}$$

In-context learning: We use in-context learning to provide a demonstration of the user preferences to LLM using certain examples. As suggested by Hou et al. [27], providing examples of other users may introduce extra noise if a user has different preferences. Therefore, we sort every weak user's preferences based on explicit user ratings. For example, *"User $\{user\_id\}$ liked the following movies in decreasing order of preference where the topmost item is the most preferred one: 1. Harry Potter, 2. Jurassic Park . . . "*. This forms the condition part of the instruction. We then select items which served as test items for recommendation models as candidate items and instruct LLM to rank them in decreasing order of preference as *"Now, rank the following items in decreasing order of preference such that the top most movie should be the most preferred one: Multiplicity, Dune . . . "*.

It is important to note that while the presentation order in conditions plays a significant role in demonstrating user preferences to LLM, we deliberately shuffle the candidate items to test the ability of LLM to rank correctly. Since LLMs can generate items out of the set, we specially instruct to restrict recommendations to the candidate set. Fig. 2 shows the final template of the instruction given to LLM for a particular user. We use the same template for all identified weak users to contextualize their past interactions into a ranking task.

Algorithm 1 Hybrid LLM-RecSys Algorithm for Ranking

```
1:  Input: $\mathcal{D}_{train}$: training dataset; $\mathcal{D}_{test}$: test dataset; $\mathcal{U}$: set of users; $\mathcal{S}_I$: Sparsity index for all users; $\theta^r$: recommendation algorithm; $\theta^l$: large language model, $t_s$: sparsity threshold, $t_p$: performance threshold.
2:  Output: $ranked\_pred_{strong}$: ranked lists of items for strong users, $ranked\_pred_{weak}$: ranked lists of items for weak users.
3:  $ranked\_pred \leftarrow \theta^r(\mathcal{D}_{train})$
4:  for each user $u_m \in \mathcal{U}$ do
5:      Calculate $\mathcal{P}(u_m)$ using Eq. 3
6:      Calculate $\mathcal{S}(u_m)$ using Eq. 4
7:      if $\mathcal{P}(u_m) < t_p$ && $\mathcal{S}_I(u_m)$ then
8:          $\mathcal{U}_{weak} \leftarrow u_m$
9:      else
10:         $\mathcal{U}_{strong} \leftarrow u_m$
11:         $ranked\_pred_{strong} \leftarrow ranked\_pred[u_m]$
12:     end if
13: end for
14: for each $u_i \in \mathcal{U}_{weak}$ do
15:     Generate instruction $\mathcal{I}_{u_i}$ using Eq. 6
16:     $ranked\_list_{u_i} = \theta^L(\mathcal{I}_u)$
17:     $ranked\_list_{weak} \leftarrow ranked\_list_{u_i}$
18: end for
```

3.4. Our Framework

This section discusses the workflow adopted by our framework as depicted in Fig. 1 and corresponding algorithm 1. Initially, the model takes input as the training $\mathcal{D}_{train}$ and test dataset $\mathcal{D}_{test}$, a set of users $\mathcal{U}$, a recommendation model θ^r, large language model θ^l and two thresholds: sparsity threshold t_s and performance threshold t_p which depict the minimum sparsity and performance values for user to be classified as strong user. It is important to note that splitting data will not yield a mutually exclusive set of users in both sets, but item ratings for each user in $\mathcal{D}_{train}$ will differ from those in $\mathcal{D}_{test}$.

The algorithm begins by training the recommendation model θ^r on the training set $\mathcal{D}_{train}$ and provides ranked items for all users. Using $\mathcal{D}_{test}$, we test the ranking ability of the model for each user by evaluating $\mathcal{P}(u_m)$ using Eq. 3. Further, each user is also assigned a sparsity score $\mathcal{S}_I(u)$ evaluated using Eq. 4. If $\mathcal{P}(u)$ has a value less than t_p and the sparsity index $\mathcal{S}_I(u)$ for a particular user falls below t_s, the user is termed as a weak user. While previous works have shown that, on average, inactive users receive poor performance, we pinpoint weak users by evaluating both the sparsity and performance.

For all such weak users, we convert rating histories from $\mathcal{D}_{train}$ as conditions $\mathcal{H}_u$ using in-context learning and use test items as candidate items $\mathcal{C}_u$ for testing purposes. However, in practice, these candidate items can be replaced by unobserved items. The final instructions are generated by combining conditions and candidate items as depicted by (Eq.6). These instructions are given to the LLM, which provides a ranked list of items for each user. For all the strong users, the recommendations presented are the ones ranked by the conventional recommendation model. However, the weak users receive final ranked lists generated by the LLM.

Table 1
Datasets statistics

	ML-1M	ML-100k	Book-Crossing
# Users	6,041	943	6,810
# Items	3,952	1,682	9,135
# Interactions	1,000,209	100,000	114,426
Sparsity	95.81%	93.7%	99.82%
Domain	Movies	Movies	Books

4. Experiments

This section discusses our experimental setup with details of the datasets and models used, followed by the implementation details of all these models and various metrics used. We finally present empirical results and a comparative analysis of various recommendation models and LLMs.

4.1. Experimental Setup

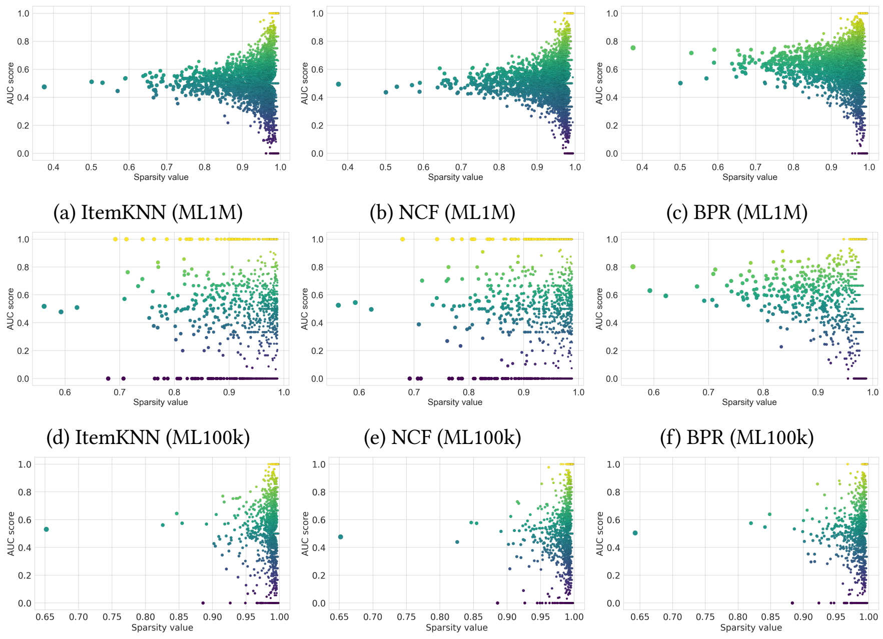
4.1.1. Datasets.

To test the effectiveness of our framework, we conducted experiments on three real-world datasets: **ML-1M**[1], **ML100k**[2], and **Book-Crossing (B-C)**[3]. Both ML100k and ML1M are movie-rating datasets, and book-crossing is a book-rating dataset. We select three datasets with varying levels of sparsity for evaluating robustness to data sparsity- ML100k has the least sparsity, and Book-Crossing has the highest sparsity (for exact values, refer to table 1). All these datasets have explicit user preference in the form of ratings ranging from $0 - 5$ for movie ratings and $0 - 10$ for book ratings dataset. We do not filter out users from ML1M and ML100k as each user has rated at least 20 movies in both these datasets. For consistency, we filter out users with less than 20 ratings from the Book-Crossing dataset. While both movie-ratings datasets have clustering based on sensitive attributes like age and gender, this paper aims to boost performance on all weak users irrespective of the sensitive features. Thus, following the protocol adopted by [13], we divided users based on their activity or the number of items rated. Any user who has rated items below a certain threshold t_s is termed an inactive user, and all those above this threshold are active users. We calculated the number of items rated on an average by all the users and used this average value as a threshold; this threshold can always vary and be set to different values per application. Table 1 presents the statistics of all three datasets.

4.1.2. Baselines and Models.

Our hybrid framework uses both traditional recommendation systems and LLMs. Thus, we include two different types of recommendation models: (i) Collaborative-filtering based: Neural Collaborative Filtering (NCF) [61] as well as ItemKNN [62]; and (ii) Learning-to-rank model- Bayesian Personalized Ranking (BPR) [59]. While these models identify weak users and generate candidate items, LLMs are further deployed to improve the performance of such users. We use both open (Mixtral-8x-7b-instruct) and closed-sourced (GPT-3.5-turbo)to test the capability of the proposed framework. It is important to note that the Collaborative Filtering models are mostly used to capture the long-term preferences

[1]https://grouplens.org/datasets/movielens/1m/
[2]https://grouplens.org/datasets/movielens/100k/
[3]http://www2.informatik.uni-freiburg.de/ cziegler/BX/



(a) ItemKNN (ML1M) (b) NCF (ML1M) (c) BPR (ML1M)

(d) ItemKNN (ML100k) (e) NCF (ML100k) (f) BPR (ML100k)

(g) ItemKNN (B-C) (h) NCF (B-C) (i) BPR (B-C)

Figure 3: AUC vs Sparsity scatter plots for illustrating the performance (measured using AUC- x-axis) for all users in ML1M, ML100k and Book-Crossing (B-C) dataset on three different algorithms.

of users. We acknowledge that existing works (refer to Section 2) have used sequential recommendation models for comparing the performance of LLMs. These works also use recommendation models as candidate retrieval models and then use LLMs to rerank the candidate items. However, sequential models are used to predict the next item according to the recently bought items. We test our framework mainly on long-term user preferences and use collaborative filtering models not only for candidate item retrieval but also for recommending top-k items to strong users. However, we believe that any existing model adopting this retrieval-reranker strategy can adopt our framework. For space constraints, we present an evaluation of our framework only on NCF, ItemKNN and BPR. In line with existing literature [27, 28], we design instructions by randomly sampling 20 rated items to demonstrate the user's preferences to the LLM. Furthermore, existing works do not discuss the responsible adaptation of LLMs, and the underlying task of retrieval-reranker of such models remains consistent and can thus use our framework.

4.1.3. Implementation details.

For ease of reproducibility, we use the open-source recommendation library RECBOLE [63] for implementing all recommendation models and API calls for access to LLMs [45]. Each dataset is split into train (80%), test (10%) and validation set (10%). We carefully use the validation set to tune all recommendation models' hyperparameters. For BPR, we search for optimal learning rate in $[5e-5, 1e-4, 5e-4, 7e-4, 1e-3, 5e-3, 7e-3]$ and in $[5e-7, 1e-6, 5e-6, 1e-5, 1e-4, 1e-3]$ for NCF. Additionally, we use $[64, 32, 16]$ as MLP hidden size for all layers and search optimal dropout probability within $[0.0, 0.1, 0.3]$ for NCF. Two hyperparameter for ItemKNN involve k (neighborhood size) in $[10, 50, 100, 200, 250, 300, 400]$ and $shrink$ (normalization parameter to calculate cosine distance) in $[0.0, 0.1, 0.5, 1, 2]$. We adopt the protocol presented by a recently released toolkit RGRecSys [64] for evaluating robustness to sub-population using NDCG and AUC. We emphasize that the use of AUC to measure the hardness associated with each user for a given recommendation model is because of the consistency property of AUC. We use the popular CatBoost[6] library that offers AUC implementation for ranking and also report final NDCG@10 scores. Furthermore, we set the temperature to 0 in GPT-3.5-turbo to minimize the generation of out-of-list items and hallucinations. However, as per our observations, setting the temperature to 0 in Mixtral-8x7b-instruct, the model outputs the list in the same order in which it was given input to it. Hence, we set the temperature to 1 and removed the items which were not originally present in the candidate list. We now discuss our empirical inferences as we conduct experiments following these details.

4.2. Empirical Evaluation

4.2.1. Comparative analysis.

The *first phase* begins by identifying the inactive users. For this, we calculate the average sparsity of all users in the dataset and identify users above this threshold as inactive users. However, one can use different values for this threshold like [13] used only top 20% of the sparse users for annotating the inactive users. We then evaluate the AUC score using equation 3 to measure the performance of RS on all these users. In line with the findings of Li et al. [13], RS performs significantly well on active users as compared to inactive users. For instance-by-instance analysis of every user, we then plot AUC scores against the sparsity index as

[4]https://platform.openai.com/docs/api-reference
[5]https://www.llama-api.com/
[6]https://github.com/catboost/

Table 2
Tabular illustration of the overall comparison of results in terms of ranking quality measured using AUC and NDCG@10 for two collaborative filtering based (Neural Collaborative Filtering, ItemKNN) and one learning-to-rank models in comparison to their usage within our framework along with one open-sourced LLM (GPT-3.5-turbo) and one close-sourced LLM (Mixtral-8x7b-instruct).

	ML1M				ML100K				Book-Crossing			
	AUC	AUC (Weak Users)	NDCG@10	NDCG@10 (Weak Users)	AUC	AUC (Weak Users)	NDCG@10	NDCG@10 (Weak Users)	AUC	AUC (Weak Users)	NDCG@10	NDCG@10 (Weak Users)
ItemKNN	0.47032	0.23776	0.66792	0.58226	0.45616	0.24778	0.66792	0.58226	0.43309	0.25909	0.75197	0.65098
ItemKNN + GPT-3.5-turbo	**0.58142**	**0.51776**	**0.82643**	**0.70352**	0.59781	0.51953	**0.82643**	**0.82598**	**0.61629**	**0.49713**	**0.86212**	**0.77101**
ItemKNN + Mixtral-8x7b-instruct	0.56035	0.51708	0.70147	0.69276	**0.59972**	**0.52327**	0.70147	0.82438	0.55203	0.47215	0.85904	0.76183
NCF	0.47805	0.22945	0.78795	0.59801	0.48311	0.25182	0.78795	0.67734	0.51852	0.29004	0.78370	0.66148
NCF + GPT-3.5-turbo	**0.58935**	**0.52122**	**0.80317**	**0.71723**	0.60831	0.50814	**0.80317**	**0.82603**	**0.61946**	**0.50513**	**0.88219**	**0.78133**
NCF + Mixtral-8x7b-instruct	0.57211	0.52100	0.79178	0.70741	**0.61174**	**0.50903**	0.79178	0.82306	0.59901	0.49897	0.86254	0.78001
BPR	0.57957	0.37824	0.88833	0.73998	0.51629	0.17020	0.88833	0.71944	0.53310	0.24426	0.80405	0.70289
BPR + GPT-3.5-turbo	**0.65397**	**0.51997**	**0.90098**	**0.82117**	0.6387	0.51435	**0.90098**	**0.82452**	**0.62145**	**0.50998**	**0.88173**	**0.81933**
BPR + Mixtral-8x7b-instruct	0.64174	0.51972	0.89742	0.82998	**0.64910**	**0.52135**	0.89742	0.81761	0.61625	0.50081	0.87798	0.80284

shown in Fig. 3 for all three datasets using three different recommendation algorithms: ItemKNN, NCF, and BPR. While ItemKNN and NCF are collaborative filtering algorithms, the overall scatterplot for BPR shows better AUC scores as compared to the other two algorithms. This is because of the inherent nature of learning-to-rank models like BPR, which rank user preferences better. This figure shows that though RS performs poorly on inactive users on average, not all inactive users receive poor-quality recommendations. We thus use our definition 5 to identify such users and mark them as weak. It might be interesting to explore why not all inactive users receive poor performance. We leave this exploratory study for future work.

For the *second phase*, we design instructions for these weak users using the approach discussed inSection 3. Our results in Table 2 show that LLMs perform significantly well on these users. Using LLM and base RS models yields the best results for all three datasets. Our results show improvement in both AUC and NDCG@10 for weak users, thus demonstrating improved robustness to the sub-population of weak users. This further leads to an overall improved ranking quality. We also highlight that previous works like [34, 27] show that close-source models perform much better than open-source. However, understanding the properties of users for which LLMs inherently perform well (like we provide a mechanism of finding weak users) and responsibly assigning tasks to large models improve performance using open-source as well as closed-source models. Our results show that Mixtral-8x-7b-instruct can perform almost equally well on weak users in all the datasets and base models. Furthermore, as observed for the ML100k dataset, this open-source model can outperform GPT-3.5-turbo when evaluated on AUC. It should be noted that AUC mainly evaluates the discriminatory ability of model to rank positive items over negative and NDCG focuses on the user's satisfaction with the ranked list, considering both relevance and position. The reason for this can be associated with dataset sparsity. As shown in Table 1, the sparsity of the ML100k dataset is lesser ($\approx 93\%$) compared to the other two datasets. While Mixtral is a good choice when datasets are small and more dense, GPT-3.5-turbo performs well for extremely sparse datasets. Yet, it is important to note that in either case, the margin of the performance of both these LLMs is not significant, and thus, even open-source models can give a comparable performance. Nevertheless, usage of GPT model yields best NDCG@10 scores for all datasets.

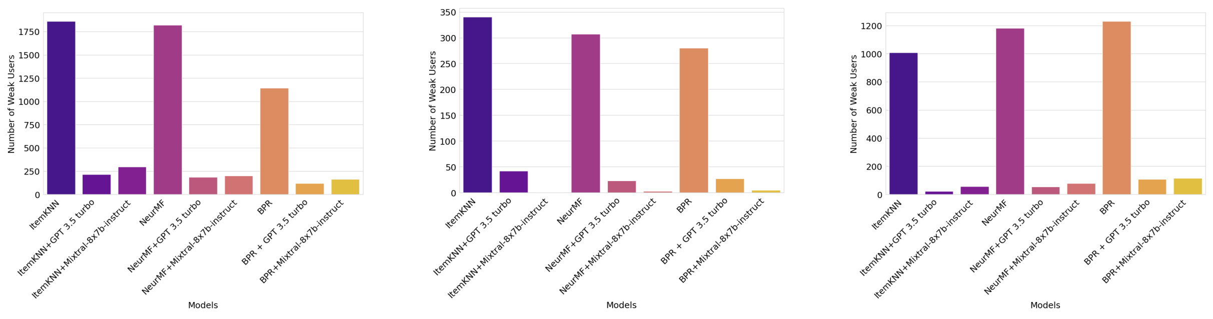
4.2.2. Reduction in weak user count.

For analyzing the variations in the count of weak users, we counted a number of weak users identified in the first phase whose interactions were contextualized and given to LLMs as the base for comparison with LLMs using the same threshold t_p. We evaluated AUC on the rankings obtained by LLM for these users. It was noted that a few users continued being hard even for LLM if the AUC lied below t_p. Fig. 4 shows that when used with large models, the count of weak users in recommendation systems drops significantly. In highly sparse datasets like that of Book-Crossing and ML1M, GPT-3.5-turbo reduced the number of weak users by $\approx 87\%$ and Mixtral-8x7b-instruct by $\approx 85\%$. On the contrary, when the dataset is dense like ML100k, Mixtral-8x7b-instruct can reduce the count by $\approx 99\%$ and the closed-source model by $\approx 88\%$. While the reduction ability of GPT-3.5-turbo remains consistent over all datasets, the open-source models yield better performance for less sparse datasets yet improve the robustness of RS to sub-populations.

In addition, we noted that a single query takes ≈ 8 seconds in GPT-3.5-turbo and ≈ 11 seconds in Mixtral-8x7b-instruct. This shows each user query's high processing times and inference latency. Thus, it is crucial to use these models responsibly by identifying what they are good at. Considering the example of the smallest dataset of ML100k, which consists of 943 users. Our strategy for identifying weak users results in only 330 weak users (worst case by ItemKNN), which leads to an overhead of $2,640$ seconds in using GPT-3.5-turbo in addition to the training time of base RS models, which is significantly less than the $7,544$ seconds if used for all the users.

5. Discussion

In this work, we implemented a novel approach for responsible adaptation of LLMs for ranking tasks. As suggested by Burnell et al. [65], the involvement of AI in high-stake decision-based applications (like ranking models for job recommendations) requires instance-by-instance evaluation instead of aggregated metrics for designing responsible AI models. Our results in Fig. 3 show that many inactive users receive recommendations of poor quality by traditional RS. Some inactive users still receive recommendations comparable to the active users, but this might be due to high similarity scores with active users that existing models can still capture their preferences effectively. We leave this as an exploratory study for future work. However, the overall performance scores on active users remain better than those of inactive users. Building upon these weak instances, our framework emphasizes on instance-by-instance evaluation of users. While we group users based on activity and then evaluate the performance of inactive users, our approach



(a) ML1M (b) ML100k (c) Book-Crossing

Figure 4: Comparative analysis of reduction in the count of weak users

pinpoints to weak users whose preferences remain hard for traditional RS to capture effectively. We believe that our framework inherently addresses the issue of group fairness. Though we do not group users based on demographics, our framework can also be extended to these scenarios where instead of activity, user demographics can be used to group users and then within the marginalized groups, the interaction histories of users who receive poor performance can be contextualized and given to LLM. However, irrespective of the demographics, our framework mainly addressed the robustness to data sparsity and sub-population of weak users, which inherently tackles the fairness issue.

This framework further helped reduce the number of queries which needed to be given to the LLM. Since most existing works (refer to Section 2) randomly select a few users out of all the users present in the dataset to evaluate the performance of LLMs, our framework provides a systematic way of selecting users for which LLMs can be used. Leveraging the capabilities of LLMs for weak users, our work emphasizes the importance of low-cost traditional RSs as well. We also observed that in some cases (Fig. 4), LLM might not be able to perform well on every weak user. This opens up new research opportunities for understanding similarities and differences within the identified weak users on which LLM does and does not perform well. Further, one can think of various prompting strategies to prompt the model to capture the preferences of extremely weak users effectively. Past works have developed various prompting strategies, which can all be tested to observe which strategies remain effective for which types of users. Nevertheless, the main goal of this paper remains to emphasise the importance of responsible adaptation of LLMs by strategically selecting tasks for which these models inherently perform well. It is also important to note that for weak users, we still obtain candidate items from traditional RS, as has been done in most past works (refer to Section 2). This helped in reducing the candidate set from thousands of unrated items to a few, which were given to the LLM to then rank. While this approach ensures that the results obtained by RS for weak users are utilized for generating candidate items instead of discarding such results directly, thus maximizing the usage of RSs even for weak users, it has a limitation. Traditional RSs perform worse on these users, and the candidate items might not capture the true preference of weak users. When we give these candidate items to LLM for ranking, the results might deviate further from true preferences. This issue can be the one reason that LLMs might not perform well on all weak users. One can, thus, further investigate the relation of candidate items to the performance of LLMs on certain users. If the candidate items are already non-preferred items by users, LLM might inherently find it difficult to perform well.

Our work, thus, represents a foundational step towards responsibly adapting LLMs while emphasizing the importance of traditional models, particularly focusing on addressing the challenges posed by sub-populations with sparse interaction histories. Our instance-by-instance evaluation approach, inspired by the imperative highlighted by recent studies in high-stakes decision-based AI applications, underscores the necessity of a nuanced understanding of individual user needs and preferences. While our framework emphasizes the importance of leveraging traditional recommendation systems alongside LLMs, we acknowledge the need to further explore the performance variations among weak users and the impact of candidate item selection on LLM effectiveness. Moving forward, our work lays the groundwork for continued research into refining the adaptation of LLMs, ensuring their responsible deployment across diverse user populations and application scenarios.

6. Conclusion

In this paper, we presented a hybrid framework that aims to improve the robustness of RS to sub-populations by leveraging LLMs. Our approach first utilized user activity to identify inactive users and then measured the performance of various RSs on these users to pinpoint weak users on which RSs find it hard to perform well. In doing so, this paper represents a novel stride towards improving robustness to sub-populations (irrespective of sensitive attributes) in RSs through efficient and responsible adaptation of LLMs without requiring any fine-tuning. While we evaluated our framework for the long-term preference of users using various collaborative filtering and learning-to-rank models, our model can be extended to various recommendation models. Our work particularly examines the importance of evaluating various user properties in the responsible adaptation of generative models within the recommendation domains. This paper opens numerous research directions for further exploration including developing prompting strategies for extremely weak users on which LLMs can not perform well. We further aim to explore other factors that can aid in the responsible adaptation of large models and improve the robustness and fairness of the challenges within recommendation systems.

References

[1] J. Jacoby, Perspectives on information overload, Journal of consumer research 10 (1984) 432–435.

[2] H. Papadakis, A. Papagrigoriou, C. Panagiotakis, E. Kosmas, P. Fragopoulou, Collaborative filtering recommender systems taxonomy, Knowledge and Information Systems 64 (2022) 35–74.

[3] P. Lops, M. De Gemmis, G. Semeraro, Content-based recommender systems: State of the art and trends, Recommender systems handbook (2011) 73–105.

[4] Y. Sun, Y. Zhang, Conversational recommender system, in: The 41st international acm sigir conference on research & development in information retrieval, 2018, pp. 235–244.

[5] S. Sanner, K. Balog, F. Radlinski, B. Wedin, L. Dixon, Large language models are competitive near cold-start recommenders for language- and item-based preferences, in: Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 890–896. URL: https://doi.org/10.1145/3604915.3608845. doi:10.1145/3604915.3608845.

[6] Y. Geifman, R. El-Yaniv, Deep active learning over the long tail, arXiv preprint arXiv:1711.00941 (2017).

[7] Z. Wang, J. Liu, H. Zou, X. Zhang, Y. He, D. Liang, P. Cui, Exploring and exploiting data heterogeneity in recommendation, arXiv preprint arXiv:2305.15431 (2023).

[8] A. J. Chaney, B. M. Stewart, B. E. Engelhardt, How algorithmic confounding in recommendation systems increases homogeneity and decreases utility, in: Proceedings of the 12th ACM conference on recommender systems, 2018, pp. 224–232.

[9] A. Beutel, E. H. Chi, Z. Cheng, H. Pham, J. Anderson, Beyond globally optimal: Focused learning for improved recommendations, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 203–212.

[10] S. Yao, Y. Halpern, N. Thain, X. Wang, K. Lee, F. Prost, E. H. Chi, J. Chen, A. Beutel, Measuring recommender system effects with simulated users, arXiv preprint arXiv:2101.04526 (2021).

[11] Y. Zhang, D. Z. Cheng, T. Yao, X. Yi, L. Hong, E. H. Chi, A model of two tales: Dual transfer learning framework for improved long-tail item recommendation, in: Proceedings of the web conference 2021, 2021, pp. 2220–2231.

[12] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi, et al., Fairness in recommendation ranking through pairwise comparisons, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2212–2220.

[13] Y. Li, H. Chen, Z. Fu, Y. Ge, Y. Zhang, User-oriented fairness in recommendation, in: Proceedings of the web conference 2021, 2021, pp. 624–632.

[14] A. Gunawardana, G. Shani, S. Yogev, Evaluating recommender systems, in: Recommender systems handbook, Springer, 2012, pp. 547–601.

[15] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., Gpt-4 technical report, arXiv preprint arXiv:2303.08774 (2023).

[16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).

[17] E. Collins, Z. Ghahramani, Lamda: our breakthrough conversation technology, Google AI Blog (2021).

[18] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al., Mixtral of experts, arXiv preprint arXiv:2401.04088 (2024).

[19] Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu, et al., Exploring the potential of large language models (llms) in learning on graphs, arXiv preprint arXiv:2307.03393 (2023).

[20] R. Bansal, B. Samanta, S. Dalmia, N. Gupta, S. Vashishth, S. Ganapathy, A. Bapna, P. Jain, P. Talukdar, Llm augmented llms: Expanding capabilities through composition, arXiv preprint arXiv:2401.02412 (2024).

[21] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, X. Hu, Harnessing the power of llms in practice: A survey on chatgpt and beyond, ACM Transactions on Knowledge Discovery from Data (2023).

[22] I. Mohanty, Recommendation systems in the era of llms, in: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation, 2023, pp. 142–144.

[23] C. Huang, T. Yu, K. Xie, S. Zhang, L. Yao, J. McAuley, Foundation models for recommender systems: A survey and new perspectives, arXiv preprint arXiv:2402.11143 (2024).

[24] W. Fan, Z. Zhao, J. Li, Y. Liu, X. Mei, Y. Wang, J. Tang, Q. Li, Recommender systems in the era of large language models (llms), arXiv preprint arXiv:2307.02046 (2023).

[25] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, X. Li, C. Zhu, H. Guo, Y. Yu, R. Tang, et al., How can recommender systems benefit from large language models: A survey, arXiv preprint arXiv:2306.05817 (2023).

[26] A. Singh, T. Joachims, Fairness of exposure in rankings, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 2219–2228.

[27] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, W. X. Zhao, Large language models are zero-shot rankers for recommender systems, in: European Conference on Information Retrieval, Springer, 2024, pp. 364–381.

[28] L. Xu, J. Zhang, B. Li, J. Wang, M. Cai, W. X. Zhao, J.-R. Wen, Prompting large language models for recommender systems: A comprehensive framework and empirical analysis, arXiv preprint arXiv:2401.04997 (2024).

[29] D. Wang, X. Hou, X. Yang, B. Zhang, R. Chen, D. Xue, Multiple key-value strategy in recommendation systems incorporating large language model, arXiv preprint arXiv:2310.16409 (2023).

[30] P. Ghosh, V. Sadaphal, Jobrecogpt–explainable job recommendations using llms, arXiv preprint arXiv:2309.11805 (2023).

[31] J. Zhang, Y. Hou, R. Xie, W. Sun, J. McAuley, W. X. Zhao, L. Lin, J.-R. Wen, Agentcf: Collaborative learning with autonomous language agents for recommender systems, arXiv preprint arXiv:2310.09233 (2023).

[32] M. S. Tamber, R. Pradeep, J. Lin, Scaling down, litting up: Efficient zero-shot listwise reranking with seq2seq encoder-decoder models, arXiv preprint arXiv:2312.16098 (2023).
[33] Y. Wang, Z. Liu, J. Zhang, W. Yao, S. Heinecke, P. S. Yu, Drdt: Dynamic reflection with divergent thinking for llm-based sequential recommendation, arXiv preprint arXiv:2312.11336 (2023).
[34] Z. Yue, S. Rabhi, G. d. S. P. Moreira, D. Wang, E. Oldridge, Llamarec: Two-stage recommendation using large language models for ranking, arXiv preprint arXiv:2311.02089 (2023).
[35] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, A. Artussi, M. Shafique, S. Garg, Building robust machine learning systems: Current progress, research challenges, and opportunities, in: Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–4.
[36] A. Zhang, J. Zheng, X. Wang, Y. Yuan, T.-S. Chua, Invariant collaborative filtering to popularity distribution shift, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 1240–1251.
[37] A. Zhang, W. Ma, J. Zheng, X. Wang, T.-S. Chua, Robust collaborative filtering to popularity distribution shift, ACM Transactions on Information Systems 42 (2024) 1–25.
[38] J. Zhao, W. Wang, X. Lin, L. Qu, J. Zhang, T.-S. Chua, Popularity-aware distributionally robust optimization for recommendation system, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 4967–4973. URL: https://doi.org/10.1145/3583780.3615492. doi:10.1145/3583780.3615492.
[39] Z. Yang, X. He, J. Zhang, J. Wu, X. Xin, J. Chen, X. Wang, A generic learning framework for sequential recommendation with distribution shifts, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 331–340.
[40] B. Wang, J. Chen, C. Li, S. Zhou, Q. Shi, Y. Gao, Y. Feng, C. Chen, C. Wang, Distributionally robust graph-based recommendation system, 2024. arXiv:2402.12994.
[41] J. Jia, Y. Liu, Y. Hu, N. Z. Gong, {PORE}: Provably robust recommender systems against data poisoning attacks, in: 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 1703–1720.
[42] R. Burke, M. P. O'Mahony, N. J. Hurley, Robust collaborative recommendation, Recommender systems handbook (2015) 961–995.
[43] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, Influence-driven data poisoning for robust recommender systems, IEEE Transactions on Pattern Analysis and Machine Intelligence (2023).
[44] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, T.-S. Chua, Adversarial training towards robust multimedia recommender system, IEEE Transactions on Knowledge and Data Engineering 32 (2019) 855–867.
[45] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, S. Yuan, Fight fire with fire: towards robust recommender systems via adversarial poisoning training, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1074–1083.
[46] J.-y. Song, B. Suh, Data augmentation strategies for improving sequential recommender systems, arXiv preprint arXiv:2203.14037 (2022).
[47] Z. Ovaisi, S. Heinecke, J. Li, Y. Zhang, E. Zheleva, C. Xiong, Rgrecsys: A toolkit for robustness evaluation of recommender systems, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1597–1600.
[48] K. Yang, J. Stoyanovich, Measuring fairness in ranked outputs, CoRR abs/1610.08559 (2016). URL: http://arxiv.org/abs/1610.08559. arXiv:1610.08559.
[49] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, R. Baeza-Yates, Fa*ir: A fair top-k ranking algorithm, CoRR abs/1706.06368 (2017). URL: http://arxiv.org/abs/1706.06368. arXiv:1706.06368.
[50] L. E. Celis, D. Straszak, N. K. Vishnoi, Ranking with fairness constraints, CoRR abs/1704.06840 (2017). URL: http://arxiv.org/abs/1704.06840. arXiv:1704.06840.
[51] T. Kamishima, S. Akaho, H. Asoh, J. Sakuma, Recommendation independence, in: S. A. Friedler, C. Wilson (Eds.), Proceedings of the 1st Conference on Fairness, Accountability and Transparency, volume 81 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 187–201. URL: https://proceedings.mlr.press/v81/kamishima18a.html.
[52] M. Zehlike, C. Castillo, Reducing disparate exposure in ranking: A learning to rank approach, CoRR abs/1805.08716 (2018). URL: http://arxiv.org/abs/1805.08716. arXiv:1805.08716.
[53] S. Yao, B. Huang, Beyond parity: Fairness objectives for collaborative filtering, CoRR abs/1705.08804 (2017). URL: http://arxiv.org/abs/1705.08804. arXiv:1705.08804.
[54] Y. Wang, W. Ma, M. Zhang, Y. Liu, S. Ma, A survey on the fairness of recommender systems, ACM Transactions on Information Systems 41 (2023) 1–43.
[55] H. Wen, X. Yi, T. Yao, J. Tang, L. Hong, E. H. Chi, Distributionally-robust recommendations for improving worst-case user experience, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 3606–3610.
[56] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, J. Zhang, Chat-rec: Towards interactive and explainable llms-augmented recommender system, ArXiv abs/2303.14524 (2023). URL: https://api.semanticscholar.org/CorpusID:257766541.
[57] W.-C. Kang, J. McAuley, Self-attentive sequential recommendation, in: 2018 IEEE international conference on data mining (ICDM), IEEE, 2018, pp. 197–206.
[58] S. Dai, N. Shao, H. Zhao, W. Yu, Z. Si, C. Xu, Z. Sun, X. Zhang, J. Xu, Uncovering chatgpt's capabilities in recommender systems, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1126–1132.
[59] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618 (2012).
[60] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 1748–1757.
[61] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.
[62] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based

collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285–295.

[63] W. X. Zhao, Y. Hou, X. Pan, C. Yang, Z. Zhang, Z. Lin, J. Zhang, S. Bian, J. Tang, W. Sun, Y. Chen, L. Xu, G. Zhang, Z. Tian, C. Tian, S. Mu, X. Fan, X. Chen, J. Wen, Recbole 2.0: Towards a more up-to-date recommendation library, in: CIKM, ACM, 2022, pp. 4722–4726.

[64] Z. Ovaisi, S. Heinecke, J. Li, Y. Zhang, E. Zheleva, C. Xiong, Rgrecsys: A toolkit for robustness evaluation of recommender systems, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 1597–1600. URL: https://doi.org/10.1145/3488560.3502192. doi:10.1145/3488560.3502192.

[65] R. Burnell, W. Schellaert, J. Burden, T. D. Ullman, F. Martinez-Plumed, J. B. Tenenbaum, D. Rutar, L. G. Cheke, J. Sohl-Dickstein, M. Mitchell, D. Kiela, M. Shanahan, E. M. Voorhees, A. G. Cohn, J. Z. Leibo, J. Hernandez-Orallo, Rethink reporting of evaluation results in ai, Science 380 (2023) 136–138. URL: https://www.science.org/doi/abs/10.1126/science.adf6369. doi:10.1126/science.adf6369. arXiv:https://www.science.org/doi/pdf/10.1126/science.adf6369.