# Large Language Models are Zero-Shot Rankers for Recommender Systems

Yupeng Hou[1,2], Junjie Zhang[1], Zihan Lin[3], Hongyu Lu[4], Ruobing Xie[4], Julian McAuley[2], and Wayne Xin Zhao[1(✉)]

[1] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
yphou@ucsd.edu, junjie.zhang@ruc.edu.cn, batmanfly@gmail.com
[2] UC San Diego, San Diego, USA
[3] School of Information, Renmin University of China, Beijing, China
[4] WeChat, Tencent, Shenzhen, China

**Abstract.** Recently, large language models (LLMs) (*e.g.,* GPT-4) have demonstrated impressive general-purpose task-solving abilities, including the potential to approach recommendation tasks. Along this line of research, this work aims to investigate the capacity of LLMs that act as the ranking model for recommender systems. We first formalize the recommendation problem as a conditional ranking task, considering sequential interaction histories as *conditions* and the items retrieved by other candidate generation models as *candidates*. To solve the ranking task by LLMs, we carefully design the prompting template and conduct extensive experiments on two widely-used datasets. We show that LLMs have promising zero-shot ranking abilities but (1) struggle to perceive the order of historical interactions, and (2) can be biased by popularity or item positions in the prompts. We demonstrate that these issues can be alleviated using specially designed prompting and bootstrapping strategies. Equipped with these insights, zero-shot LLMs can even challenge conventional recommendation models when ranking candidates are retrieved by multiple candidate generators. The code and processed datasets are available at https://github.com/RUCAIBox/LLMRank.

**Keywords:** Large Language Model · Recommender System

## 1 Introduction

In the literature of recommender systems, most existing models are trained with user behavior data from a specific domain or scenario [26,28,49], suffering from two major issues. Firstly, it is difficult to capture user preference by solely modeling historical behaviors, *e.g.,* clicked item sequences [28,33,81], limiting the expressive power to model more complicated but explicit user interests (*e.g.,* intentions expressed in natural language). Secondly, these models are essentially

Y. Hou and J. Zhang—Equal contribution.


N. Goharian et al. (Eds.): ECIR 2024, LNCS 14609, pp. 364–381, 2024.
https://doi.org/10.1007/978-3-031-56060-6_24

"*narrow experts*", lacking more comprehensive knowledge in solving complicated recommendation tasks that rely on background or commonsense knowledge [23].

To improve recommendation performance and interactivity, there have been increasing efforts that explore the use of pre-trained language models (PLMs) in recommender systems [21,30,62]. They aim to explicitly capture user preference in natural language [21] or transfer rich world knowledge from text corpora [29,30]. Despite their effectiveness, thoroughly fine-tuning the recommendation models on task-specific data is still a necessity, making it less capable of solving diverse recommendation tasks [30]. More recently, large language models (LLMs) have shown great potential to serve as zero-shot task solvers [52,64]. Indeed, there are some preliminary attempts that employ LLMs for solving recommendation tasks [13,20,40,59,60,73]. These studies mainly focus on discussing the possibility of building a capable recommender with LLMs. While promising, the insufficient understanding of the new characteristics when making recommendations using LLMs could hinder the development of this new paradigm.

In this paper, we conduct empirical studies to investigate what determines the capacity of LLMs that serve as recommendation models. Typically, recommender systems are developed in a pipeline architecture [10], consisting of *candidate generation* (retrieving relevant items) and *ranking* (ranking relevant items at a higher position) procedures. This work mainly focuses on the ranking stage of recommender systems, since LLMs are more expensive to run on a large-scale candidate set. Further, the ranking performance is sensitive to the retrieved candidate items, which is more suitable to examine the subtle differences in the recommendation abilities of LLMs.

To carry out this study, we first formalize the recommendation process of LLMs as a *conditional ranking* task. Given prompts that include sequential historical interactions as *"conditions"*, LLMs are instructed to rank a set of *"candidates"* (*e.g.,* items retrieved by candidate generation models), according to LLM's intrinsic knowledge. Then we conduct control experiments to systematically study the empirical performance of LLMs as rankers by designing specific configurations for "conditions" and "candidates", respectively. Overall, we attempt to answer the following key questions:

- What factors affect the zero-shot ranking performance of LLMs?
- What data or knowledge do LLMs rely on for recommendation?

Our empirical experiments are conducted on two public datasets for recommender systems. The results lead to several key findings that potentially shed light on how to develop LLMs as powerful ranking models for recommender systems. We summarize the key findings as follows:

- LLMs *struggle to perceive the order* of the given sequential interaction histories. By employing specifically designed promptings, *LLMs can be triggered to perceive the order*, leading to improved ranking performance.
- LLMs suffer from position bias and popularity bias while ranking, which can be alleviated by bootstrapping or specially designed prompting strategies.

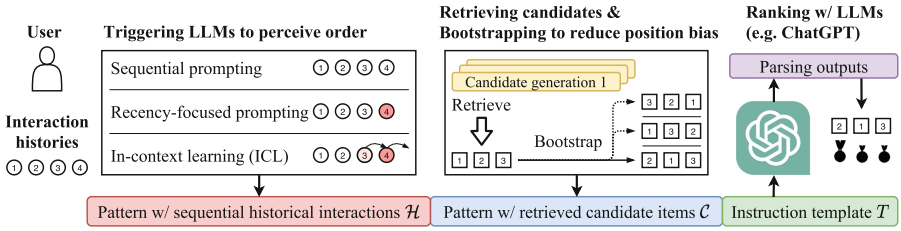



**Fig. 1.** An overview of the proposed LLM-based zero-shot ranking method.

– LLMs outperform existing zero-shot recommendation methods, showing promising zero-shot ranking abilities, especially on candidates retrieved by multiple candidate generation models with different practical strategies.

## 2 General Framework for LLMs as Rankers

To investigate the recommendation abilities of LLMs, we first formalize the recommendation process as a conditional ranking task. Then, we describe a general framework that adapts LLMs to solve the recommendation task.

### 2.1 Problem Formulation

Given the historical interactions $\mathcal{H} = \{i_1, i_2, \ldots, i_n\}$ of one user (in chronological order of interaction time) as *conditions*, the task is to rank the *candidate* items $\mathcal{C} = \{i_j\}_{j=1}^{m}$, such that the items of interest would be ranked at a higher position. In practice, the candidate items are usually retrieved by candidate generation models from the whole item set $\mathcal{I}$ ($m \ll |\mathcal{I}|$) [10]. Further, we assume that each item $i$ is associated with a descriptive text $t_i$ following [30].

### 2.2 Ranking with LLMs Using Natural Language Instructions

We use LLMs as ranking models to solve the above-mentioned task in an instruction-following paradigm [64]. Specifically, for each user, we first construct two natural language patterns that contain sequential interaction histories $\mathcal{H}$ (*conditions*) and retrieved candidate items $\mathcal{C}$ (*candidates*), respectively. Then these patterns are filled into a natural language template $T$ as the final instruction. In this way, LLMs are expected to understand the instructions and output the ranking results as the instruction suggests. The overall framework of the ranking approach by LLMs is depicted in Fig. 1. Next, we describe the detailed instruction design in our approach.

**Sequential Historical Interactions.** To investigate whether LLMs can capture user preferences from historical user behaviors, we include sequential historical interactions $\mathcal{H}$ into the instructions as inputs of LLMs. To enable LLMs

to be aware of the sequential nature of historical interactions, we propose three ways to construct the instructions:

- **Sequential prompting**: Arrange the historical interactions in chronological order. This way has also been used in prior studies [13]. For example, *"I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', ..."*.
- **Recency-focused prompting**: In addition to the sequential interaction records, we can add an additional sentence to emphasize the most recent interaction. For example, *"I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', .... Note that my most recently watched movie is Dead Presidents. ..."*.
- **In-context learning (ICL)**: ICL is a prominent prompting approach for LLMs to solve various tasks [78], where it includes demonstration examples in the prompt. For the personalized recommendation task, simply introducing examples of other users may introduce noises because users usually have different preferences. Instead, we introduce demonstration examples by augmenting the input interaction sequence itself. We pair the prefix of the input interaction sequence and the corresponding successor as examples. For instance, *" If I've watched the following movies in the past in order: '0. Multiplicity', '1. Jurassic Park', ..., then you should recommend Dead Presidents to me and now that I've watched Dead Presidents, then ..."*.

**Retrieved Candidate Items.** Typically, candidate items to be ranked are first retrieved by candidate generation models [10]. In this work, we consider a relatively small pool for the candidates, and keep 20 candidate items (*i.e.,* $m = 20$) for ranking. To rank these candidates with LLMs, we arrange the candidate items $\mathcal{C}$ in a sequential manner. For example, *"Now there are 20 candidate movies that I can watch next: '0. Sister Act', '1. Sunset Blvd', ..."*. Note that, following the classic candidate generation approach [10], there is no specific order for candidate items. As a result, We generate different orders for the candidate items in the prompts, which enables us to further examine whether the ranking results of LLMs are affected by the arrangement order of candidates, *i.e.,* position bias, and how to alleviate position bias via bootstrapping.

**Ranking with Large Language Models.** Existing studies show that LLMs can follow natural language instructions to solve diverse tasks in a zero-shot setting [64,78]. To rank using LLMs, we infill the patterns above into the instruction template $T$. An example instruction template can be given as: *" [pattern that contains sequential historical interactions $\mathcal{H}$] [pattern that contains retrieved candidate items $\mathcal{C}$] Please rank these movies by measuring the possibilities that I would like to watch next most, according to my watching history."*.

**Parsing the Output of LLMs.** Note that the output of LLMs is still in natural language text, and we parse the output with heuristic text-matching methods

**Table 1.** Statistics of the preprocessed datasets. "Avg. $|\mathcal{H}|$" denotes the average length of historical interactions. "Avg. $|t_i|$" denotes the average number of tokens in the item text.

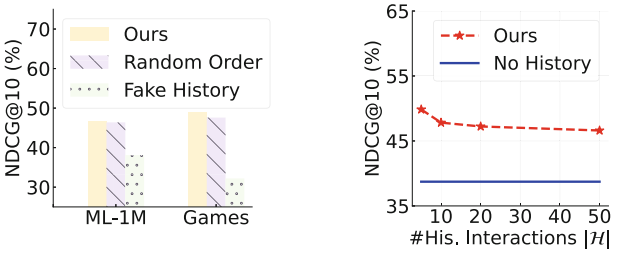
| Dataset | #Users | #Items | #Interactions | Sparsity | Avg. $\|\mathcal{H}\|$ | Avg. $\|t_i\|$ |
|---|---|---|---|---|---|---|
| ML-1M | 6,040 | 3,706 | 1,000,209 | 95.53% | 46.19 | 16.96 |
| Games | 50,547 | 16,859 | 389,718 | 99.95% | 7.02 | 43.31 |

and ground the recommendation results on the specified item set. In detail, we can directly perform efficient substring matching algorithms like KMP [35] between the LLM outputs and the text of candidate items. We also found that LLMs occasionally generate items that are not present in the candidate set. For GPT-3.5, such deviations occur in a mere 3% of cases. One can either reprocess the illegal cases or simply treat the out-of-candidate items as incorrect recommendations.

## 3 Empirical Studies

**Datasets.** The experiments are conducted on two widely-used public datasets for recommender systems: (1) the movie rating dataset *MovieLens-1M* [24] (in short, **ML-1M**) where user ratings are regarded as interactions, and (2) one category from the *Amazon Review* dataset [46] named **Games** where reviews are regarded as interactions. We filter out users and items with fewer than five interactions. Then we sort the interactions of each user by timestamp, with the oldest interactions first, to construct the corresponding historical interaction sequences. The movie/product titles are used as the descriptive text of an item. We use item titles in this study for two reasons: (1) to determine if LLMs can make recommendations based on their intrinsic world knowledge with minimal information provided, and (2) to conserve computational resources. Exploring how LLMs use more extensive textual features for recommendations will be the focus of our future work. Statistics of the preprocessed datasets are presented in Table 1

**Evaluation and Implementation Details.** Following existing works [30,33], we apply the leave-one-out strategy for evaluation. For each historical interaction sequence, the last item is used as the ground-truth item in test set. The item before the last one is used in the validation set (used for training baseline methods). We adopt the widely used metric NDCG@K (in short, N@K) to evaluate the ranking results over the given $m$ candidates, where $K \leq m$. To ease the reproduction of this work, our experiments are conducted using a popular open-source recommendation library RecBole [77]. The historical interaction sequences are truncated within a length of 50. We evaluate LLM-based methods on all users in ML-1M dataset and randomly sampled 6,000 users for Games dataset by default. Unless specified, the evaluated LLM is accessed by calling OpenAI's




(a) Perceive Order (b) Performance vs. $|\mathcal{H}|$

**Fig. 2.** Analysis of whether LLMs perceive the order of historical interactions.

API `gpt-3.5-turbo`. The hyperparameter temperature of calling LLMs is set to 0.2. *All the reported results are the average of at least three repeat runs to reduce the effect of randomness.*

### 3.1 Can LLMs Understand Prompts that Involve Sequential Historical User Behaviors?

In LLM-based methods, historical interactions are naturally arranged in an ordered sequence. By designing different configurations of $\mathcal{H}$, we aim to examine whether LLMs can leverage these historical user behaviors and perceive the sequential nature for making accurate recommendations.

**LLMs Struggle to Perceive the Order of Given Historical User Behaviors.** In this section, we examine whether LLMs can understand prompts with ordered historical interactions and give personalized recommendations. The task is to rank a candidate set of 20 items, containing one ground-truth item and 19 randomly sampled negatives. By analyzing historical behaviors, items of interest should be ranked at a higher position. We compare the ranking results of three LLM-based methods: (a) *Ours*, which ranks as we have described in Sect. 2.2. Historical user behaviors are encoded into prompts using the "sequential prompting" strategy. (b) *Random Order*, where the historical user behaviors will be randomly shuffled before being fed to the model, and (c) *Fake History*, where we replace all the items in original historical behaviors with randomly sampled items as fake historical behaviors. From Fig. 2(a), we can see that *Ours* has better performance than variants with fake historical behaviors. However, the performance of *Ours* and *Random Order* is similar, indicating that LLMs are not sensitive to the order of the given historical user interactions.

Moreover, in Fig. 2(b), we vary the number of latest historical user behaviors ($|\mathcal{H}|$) used for constructing the prompt from 5 to 50. The results show that increasing the number of historical user behaviors does not improve, but rather negatively impacts the ranking performance. We speculate that this phenomenon is caused by the fact that LLMs have difficulty understanding the order, but

consider all the historical behaviors equally. Therefore too many historical user behaviors (*e.g.,* $|\mathcal{H}| = 50$) may overwhelm LLMs and lead to a performance drop. In contrast, a relatively small $|\mathcal{H}|$ enables LLMs to concentrate on the most recently interacted items, resulting in better recommendation performance.

**Table 2.** Performance comparison on *randomly retrieved candidates*. Ground-truth items are included in the candidate sets. "full" denotes models that are trained on the target dataset, and "zero-shot" denotes models that are not trained on the target dataset but could be pre-trained. We highlight the best performance among zero-shot recommendation methods in **bold**.

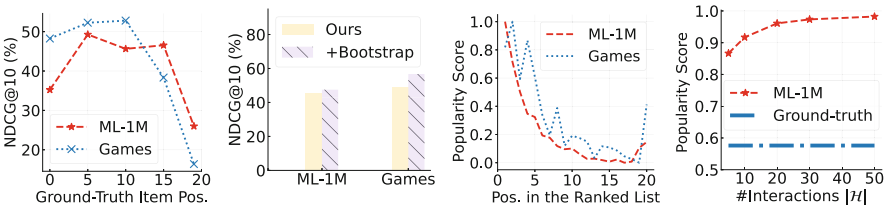
| | Method | ML-1M | | | | Games | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | N@1 | N@5 | N@10 | N@20 | N@1 | N@5 | N@10 | N@20 |
| full | Pop | 22.91 | 45.16 | 52.33 | 55.36 | 28.35 | 47.42 | 52.96 | 57.45 |
| | BPRMF [49] | 34.60 | 59.87 | 64.29 | 65.39 | 44.92 | 62.33 | 66.27 | 68.94 |
| | SASRec [33] | 61.39 | 76.39 | 78.89 | 79.79 | 56.90 | 73.19 | 75.92 | 77.14 |
| zero-shot | BM25 [50] | 4.70 | 12.68 | 17.88 | 33.19 | 13.92 | 28.81 | 34.61 | 44.35 |
| | UniSRec [30] | 7.37 | 18.80 | 26.67 | 37.93 | 18.95 | 33.99 | 40.71 | 48.42 |
| | VQ-Rec [29] | 5.98 | 15.48 | 23.74 | 35.85 | 7.28 | 18.28 | 26.21 | 37.62 |
| | Sequential | 18.28 | 36.35 | 42.85 | 49.02 | 30.28 | 45.48 | 50.57 | 56.55 |
| | Recency-Focused | 19.57 | 37.73 | 44.23 | 50.01 | **34.03** | **48.77** | **53.50** | **59.01** |
| | In-Context Learning | **21.77** | **39.59** | **45.83** | **51.62** | 33.95 | 48.44 | 53.10 | 58.92 |

**Triggering LLMs to Perceive the Interaction Order.** Based on the above observations, we find it difficult for LLMs to perceive the order in interaction histories by a default prompting strategy. As a result, we aim to elicit the order-perceiving abilities of LLMs, by proposing two alternative prompting strategies and emphasizing the recently interacted items. Detailed descriptions of the proposed strategies have been given in Sect. 2.2. In Table 2, we can see that both recency-focused prompting and in-context learning can generally improve the ranking performance of LLMs, though the best strategy may vary on different datasets. The above results can be summarized as the following key observation:

**Observation 1.** LLMs *struggle to perceive the order* of the given sequential interaction histories. By employing specifically designed promptings, *LLMs can be triggered to perceive the order* of historical user behaviors, leading to improved ranking performance.

### 3.2 Do LLMs Suffer from Biases While Ranking?

The biases and debiasing methods in conventional recommender systems have been widely studied [5]. For LLM-based recommendation models, both the input and output are natural language texts and will inevitably introduce new biases. In this section, we discuss two kinds of biases that LLM-based recommendation models suffer from. We also make discussions on how to alleviate these biases.




(a) Position Bias (b) Bootstrapping (c) Popularity Bias (d) Popularity of top-1 item *w.r.t.* $|\mathcal{H}|$

**Fig. 3.** Biases and debiasing methods in the ranking of LLMs. (a) The position of candidates in the prompts influences the ranking results. (b) Bootstrapping alleviates position bias. (c) LLMs tend to recommend popular items. (d) Focusing on historical interactions reduces popularity bias.

**The Order of Candidates Affects the Ranking Results of LLMs.** For conventional ranking methods, the order of retrieved candidates usually will not affect the ranking results [28,33]. However, for the LLM-based approach that is described in Sect. 2.2, the candidates are arranged in a sequential manner and infilled into a prompt. It has been shown that LLMs are generally sensitive to the order of examples in the prompts for NLP tasks [44,79]. As a result, we also conduct experiments to examine whether the order of candidates affects the ranking performance of LLMs. We follow the experimental settings adopted in Sect. 3.1. The only difference is that we control the order of these candidates in the prompts, by making the ground-truth items appear at a certain position. We vary the position of ground-truth items at $\{0, 5, 10, 15, 19\}$ and present the results in Fig. 3(a). We can see that the performance varies when the ground-truth items appear at different positions. Especially, the ranking performance drops significantly when the ground-truth items appear at the last few positions. The results indicate that LLM-based rankers are affected by the order of candidates, *i.e., position bias*, which may *not* affect conventional recommendation models.

**Alleviating Position Bias Via Bootstrapping.** A simple strategy to alleviate position bias is to bootstrap the ranking process. We may rank the candidate set repeatedly for $B$ times, with candidates randomly shuffled at each round. In this way, one candidate may appear in different positions. We then merge the results of each round to derive the final ranking. From Fig. 3(b), we follow the setting in Sect. 3.1 and apply the bootstrapping strategy to *Ours*. Each candidate set will be ranked for 3 times. We can see that bootstrapping improves the ranking performance on both datasets.

**Popularity Degrees of Candidates Affect Ranking Results of LLMs.** For popular items, the associated text may also appear frequently in the pre-training corpora of LLMs. For example, a best-selling book would be widely discussed on the Web. Thus, we aim to examine whether the ranking results are

**Table 3.** Zero-shot ranking performance comparison. We highlight the best performance in **bold**. Due to limited budget, we evaluate each LLM only once on 200 sampled users **only** for experiments corresponding to this table.

| Method | ML-1M | | | | Games | | | |
|---|---|---|---|---|---|---|---|---|
| | N@1 | N@5 | N@10 | N@20 | N@1 | N@5 | N@10 | N@20 |
| BM25 [50] | 4.70 | 12.68 | 17.88 | 33.19 | 13.92 | 28.81 | 34.61 | 44.35 |
| UniSRec [30] | 7.37 | 18.80 | 26.67 | 37.93 | 18.95 | 33.99 | 40.71 | 48.42 |
| Alpaca-7B [55] | 4.00 | 13.92 | 23.09 | 31.54 | 5.50 | 14.16 | 21.67 | 28.68 |
| Vicuna-13B [9] | 6.50 | 14.75 | 22.64 | 33.42 | 7.00 | 17.73 | 24.30 | 31.22 |
| LLaMA-2-70B-Chat [57] | 8.00 | 25.42 | 31.19 | 34.52 | 21.50 | 32.30 | 37.83 | 41.97 |
| ChatGPT (GPT-3.5) | **23.33** | **42.07** | **48.80** | **53.73** | 23.83 | 45.69 | 50.31 | 55.45 |
| GPT-4 | 15.50 | 40.65 | 46.74 | 48.42 | **39.50** | **58.22** | **62.88** | **65.25** |

affected by the popularity of candidates. However, it is difficult to directly measure the popularity of item text. Here, we hypothesize that the text popularity can be indirectly measured by item frequency in one recommendation dataset. In Fig. 3(c), we report the item popularity score (measured by the normalized item frequency of appearance in the training set) at each position of the ranked item lists. We can see that popular items tend to be ranked at higher positions.

**Making LLMs Focus on Historical Interactions Helps Reduce Popularity Bias.** We assume that if LLMs focus on historical interactions, they may give more personalized recommendations but not more popular ones. From Fig. 2(b), we know that LLMs make better use of historical interactions when using less historical interactions. From Fig. 3(d), we compare the popularity scores of the best-ranked items varying the number of historical interactions. It can be observed that as $|\mathcal{H}|$ decreases, the popularity score decreases as well. This suggests that one can reduce the effects of popularity bias when LLMs focus more on historical interactions. From the above experiments, we can conclude the following:

> **Observation 2.** LLMs suffer from position bias and popularity bias while ranking, which can be alleviated by bootstrapping or specially designed prompting strategies.

### 3.3 How Well Can LLMs Rank Candidates in a Zero-Shot Setting?

We further evaluate LLM-based methods on candidates with hard negatives that are retrieved by different strategies to further investigate what the ranking of LLMs depends on. Then, we present the ranking performance of different methods on candidates retrieved by multiple candidate generation models to simulate a more practical and difficult setting.

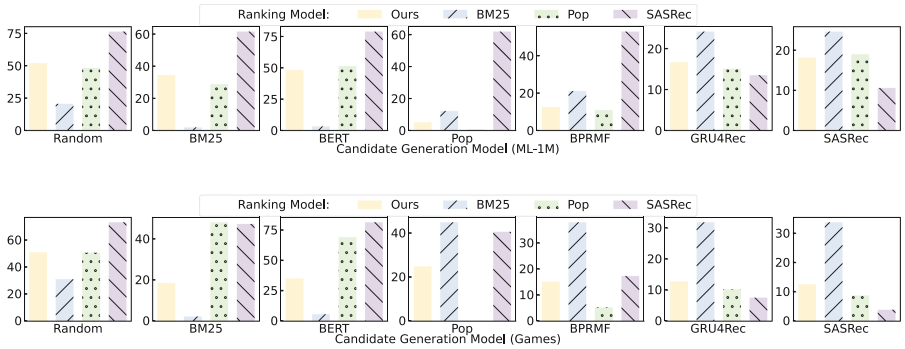



**Fig. 4.** Ranking performance measured by NDCG@10 (%) on hard negatives.

**LLMs have Promising Zero-Shot Ranking Abilities.** In Table 2, we conduct experiments to compare the ranking abilities of LLM-based methods with existing methods. We follow the same setting in Sect. 3.1 where $|\mathcal{C}| = 20$ and candidate items are randomly retrieved. We include three conventional models that are trained on the training set, *i.e.,* Pop (recommending according to item popularity), BPRMF [49], and SASRec [33]. We also evaluate three zero-shot recommendation methods that are not trained on the target datasets, including BM25 [50] (rank according to the textual similarity between candidates and historical interactions), UniSRec [30], and VQ-Rec [29]. For UniSRec and VQ-Rec, we use their publicly available pre-trained models. We do not include ZESRec [15] because there is no pre-trained model released. In addition, we compare the zero-shot ranking performance of different LLMs in Table 3. "Recency-Focused" prompting strategy is used for LLM-based rankers.

From Table 2 and 3, we can see that LLMs with more parameters generally perform better. The best LLM-based methods outperform existing zero-shot recommendation methods by a large margin, showing promising zero-shot ranking abilities. We would highlight that it is difficult to conduct zero-shot recommendations on the ML-1M dataset, due to the difficulty in measuring the similarity between movies merely by the similarity of their titles. However, LLMs can use their intrinsic knowledge to measure the similarity between movies and make recommendations. We would emphasize that the goal of evaluating zero-shot recommendation methods is not to surpass conventional models. The goal is to demonstrate the strong recommendation capabilities of pre-trained base models, which can be further adapted and transferred to downstream scenarios.

**LLMs Rank Candidates Based on Item Popularity, Text Features as Well as User Behaviors.** To further investigate how LLMs rank the given candidates, we evaluate LLMs on candidates that are retrieved by different candidate generation methods. These candidates can be viewed as hard negatives for ground-truth items, which can be used to measure the ranking ability of LLMs for specific categories of items. We consider two categories of strategies to retrieve

**Table 4.** Performance comparison on *candidates retrieved by multiple candidate generation models*. Ground-truth items are *not* guaranteed to be included in the candidate sets. "full" denotes models that are trained on the target dataset, and "zero-shot" denotes models that are not trained on the target dataset but could be pre-trained. We highlight the best and second-best performance among *all* recommendation methods in **bold**.

| | Method | ML-1M | | | | Games | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | N@1 | N@5 | N@10 | N@20 | N@1 | N@5 | N@10 | N@20 |
| full | Pop | 0.08 | 1.20 | 4.13 | 5.79 | 0.13 | 1.00 | 2.27 | 2.62 |
| | BPRMF [49] | 0.26 | 1.69 | 4.41 | 6.04 | 0.55 | 1.98 | **2.96** | **3.19** |
| | SASRec [33] | **3.76** | **9.79** | **10.45** | **10.56** | **1.33** | **3.55** | **4.02** | **4.11** |
| zero-shot | BM25 [50] | 0.26 | 0.87 | 2.32 | 5.28 | 0.18 | 1.07 | 1.80 | 2.55 |
| | UniSRec [30] | 0.88 | 3.46 | 5.30 | 6.92 | 0.00 | 1.86 | 2.03 | 2.31 |
| | VQ-Rec [29] | 0.20 | 1.60 | 3.29 | 5.73 | 0.20 | 1.21 | 1.91 | 2.64 |
| | Ours | **1.74** | **5.22** | **6.91** | **7.90** | **0.90** | **2.26** | 2.80 | 3.08 |

the candidates: (1) *content-based methods* like *BM25* [50] and *BERT* [14] retrieve candidates based on the text feature similarities, and (2) *interaction-based methods*, including *Pop*, *BPRMF* [49], *GRU4Rec* [28], and *SASRec* [33], retrieve items using neural networks trained on user-item interactions. Given candidates, we compare the ranking performance of the LLM-based model (*Ours*) and representative methods.

From Fig. 4, we can see that the ranking performance of the LLM-based method varies on different candidate sets and different datasets. (1) On ML-1M, LLM-based method cannot rank well on candidate sets that contain popular items (*e.g., Pop* and *BPRMF*), indicating the LLM-based method recommend items largely depend on item popularity on ML-1M dataset. (2) On Games, we can observe that *Ours* has similar performance both on popular candidates and textual similar candidates, showing that item popularity and text features contribute similarly to the ranking of LLMs. (3) On both two datasets, the performance of *Ours* is affected by hard negatives retrieved by interaction-based candidate generation models, but not as severe as those interaction-based rankers like *SASRec*. The above results demonstrate that LLM-based methods not only consider one single aspect for ranking, but make use of item popularity, text features, and even user behaviors. On different datasets, the weights of these three aspects to affect the ranking performance may also vary.

**LLMs Can Effectively Rank Candidates Retrieved by Multiple Candidate Generation Models.** For real-world recommender systems [10], the items to be ranked are usually retrieved by multiple candidate generation models. As a result, we also conduct experiments in a more practical and difficult setting. We use the above-mentioned seven candidate generation models to retrieve items. The top-3 best items retrieved by each candidate generation model will

be merged into a candidate set containing a total of 21 items. As a more practical setting, we do not complement the ground-truth item to each candidate set. Note that the experiments here were conducted under the implicit preference setup [76], indicating that implicit positive instances (not explicitly labeled) may exist among the retrieved items. A more faithful evaluation might require a human study, which we intend to explore in our future work. For *Ours*, we summarize the experiences gained from Sect. 3.1 and 3.2. We use the recency-focused prompting strategy to encode $|\mathcal{H}| = 5$ sequential historical interactions into prompts and use a bootstrapping strategy to repeatedly rank for 3 rounds.

From Table 4, we can see that the LLM-based model (*Ours*) yields the second-best performance over the compared recommendation models on most metrics. The results show that LLM-based zero-shot ranker even beats the conventional recommendation model *Pop* and *BPRMF* that has been trained on the target datasets, further demonstrating the strong zero-shot ranking ability of LLMs. We assume that LLMs can make use of their intrinsic world knowledge to rank the candidates comprehensively considering popularity, text features, and user behaviors. In comparison, existing models (as *narrrow experts*) may lack the ability to rank items in a complicated setting. The above findings can be summarized as:

**Observation 3.** LLMs have promising zero-shot ranking abilities, especially on candidates retrieved by multiple candidate generation models with different practical strategies.

## 4 Related Work

**Transfer Learning for Recommender Systems.** As recommender systems are mostly trained on data collected from a single source, people have sought to transfer knowledge from other domains [45,70,75,82,84,85], markets [3,51], or platforms [4,19]. Typical transfer learning methods for recommender systems rely on anchors, including shared users/items [7,8,45,68,69,83] or representations from a shared space [11,18,38]. However, these anchors are usually sparse among different scenarios, making transferring difficult for recommendations [84]. More recently, there are studies aiming to transfer knowledge stored in language models by adapting them to recommendation tasks via tuning [1,12,21,53] or prompting [37,39,74]. In this paper, we conduct zero-shot recommendation experiments to examine the potential to transfer knowledge from LLMs.

**Large Language Models for Recommender Systems.** The design of recommendation models, especially sequential recommendation models, has been long inspired by the design of language models, from word2vec [2,22,25] to recent neural networks [28,33,54,81]. In recent years, with the development of pre-trained language models (PLMs) [14], people have tried to transfer knowledge stored in PLMs to recommendation models, by either representing

items using their text features or representing behavior sequences in the format of natural language [16,21,42,58,67]. Very recently, large language models (LLMs) have been shown superior language understanding and generation abilities [6,17,47,56,66,78]. Studies have been made to make recommender systems more interactive by integrating LLMs along with conventional recommendation models [20,27,36,43,48,59,61,65] or fine-tuned with specially designed instructions [1,12,21,31,80]. There are also early explorations showing LLMs have zero-shot recommendation abilities [13,34,41,59,60,63,71,72]. Despite being effective to some extent, few works have explored what determines the recommendation performance of LLMs.

## 5 Conclusion

In this work, we investigated the capacities of LLMs that act as the zero-shot ranking model for recommender systems. To rank with LLMs, we constructed natural language prompts that contain historical interactions, candidates, and instruction templates. We then propose several specially designed prompting strategies to trigger the ability of LLMs to perceive orders of sequential behaviors. We also introduce bootstrapping and prompting strategies to alleviate the position bias and popularity bias issues that LLM-based ranking models may suffer.

Extensive empirical studies indicate that LLMs have promising zero-shot ranking abilities. The empirical studies demonstrate the strong potential of transferring knowledge from LLMs as powerful recommendation models. We aim at shedding light on several promising directions to further improve the ranking abilities of LLMs, including (1) better perceiving the order of sequential historical interactions and (2) alleviating the position bias and popularity bias. For future work, we consider developing technical approaches to solve the above-mentioned key challenges when deploying LLMs as recommendation models. We also would like to develop LLM-based recommendation models that can be efficiently tuned on downstream user behaviors for effective personalized recommendations.

## 6 Limitations

In most experiments in this paper, ChatGPT is used as the primary target LLM for evaluation. However, being a closed-source commercial service, ChatGPT might integrate additional techniques with its core large language model to improve performance. While there are open-source LLMs available, such as LLaMA 2 [57] and Mistral [32], they exhibit a notable performance disparity compared to ChatGPT (e.g., LLaMA-2-70B-Chat vs. ChatGPT in Table 3). This gap makes it difficult to evaluate the emergent abilities of LLMs on the recommendation tasks using purely open-source models. In addition, we should note that the observations might be biased by specific prompts and datasets.

**Acknowledgements.** This work was partially supported by National Natural Science Foundation of China under Grant No. 62222215, Beijing Natural Science Foundation under Grant No. L233008 and 4222027.

## References

1. Bao, K., Zhang, J., Zhang, Y., Wang, W., Feng, F., He, X.: Tallrec: an effective and efficient tuning framework to align large language model with recommendation. arXiv preprint arXiv:2305.00447 (2023)
2. Barkan, O., Koenigstein, N.: ITEM2VEC: neural item embedding for collaborative filtering. In: Palmieri, F.A.N., Uncini, A., Diamantaras, K.I., Larsen, J. (eds.) 26th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2016, Vietri sul Mare, Salerno, Italy, 13–16 September 2016, pp. 1–6. IEEE (2016). https://doi.org/10.1109/MLSP.2016.7738886
3. Bonab, H.R., Aliannejadi, M., Vardasbi, A., Kanoulas, E., Allan, J.: Cross-market product recommendation. In: Demartini, G., Zuccon, G., Culpepper, J.S., Huang, Z., Tong, H. (eds.) CIKM, pp. 110–119. ACM (2021). https://doi.org/10.1145/3459637.3482493
4. Cao, D., He, X., Nie, L., Wei, X., Hu, X., Wu, S., Chua, T.: Cross-platform app recommendation by jointly modeling ratings and texts. ACM Trans. Inf. Syst. **35**(4), 37:1–37:27 (2017). https://doi.org/10.1145/3017429
5. Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., He, X.: Bias and debias in recommender system: a survey and future directions. CoRR abs/2010.03240 (2020). https://arxiv.org/abs/2010.03240
6. Chen, J., et al.: When large language models meet personalization: perspectives of challenges and opportunities. arXiv preprint arXiv:2307.16376 (2023)
7. Chen, L., Yuan, F., Yang, J., He, X., Li, C., Yang, M.: User-specific adaptive fine-tuning for cross-domain recommendations. IEEE Trans. Knowl. Data Eng. **35**(3), 3239–3252 (2023). https://doi.org/10.1109/TKDE.2021.3119619
8. Cheng, M., Yuan, F., Liu, Q., Xin, X., Chen, E.: Learning transferable user representations with sequential behaviors via contrastive pre-training. In: Bailey, J., Miettinen, P., Koh, Y.S., Tao, D., Wu, X. (eds.) ICDM, pp. 51–60. IEEE (2021). https://doi.org/10.1109/ICDM51629.2021.00015
9. Chiang, W.L., et al.: Vicuna: an open-source chatbot impressing gpt-4 with 90%* chatgpt quality (2023). https://vicuna.lmsys.org/. Accessed 14 Apr 2023
10. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: RecSys, pp. 191–198 (2016)
11. Cui, Q., Wei, T., Zhang, Y., Zhang, Q.: Herograph: a heterogeneous graph framework for multi-target cross-domain recommendation. In: Vinagre, J., Jorge, A.M., Al-Ghossein, M., Bifet, A. (eds.) RecSys. CEUR Workshop Proceedings, vol. 2715. CEUR-WS.org (2020). https://ceur-ws.org/Vol-2715/paper6.pdf
12. Cui, Z., Ma, J., Zhou, C., Zhou, J., Yang, H.: M6-rec: generative pretrained language models are open-ended recommender systems. arXiv preprint arXiv:2205.08084 (2022)
13. Dai, S., et al.: Uncovering chatgpt's capabilities in recommender systems. arXiv preprint arXiv:2305.02182 (2023)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
15. Ding, H., Ma, Y., Deoras, A., Wang, Y., Wang, H.: Zero-shot recommender systems. arXiv:2105.08318 (2021)

16. Ding, H., Ma, Y., Deoras, A., Wang, Y., Wang, H.: Zero-shot recommender systems. arXiv preprint arXiv:2105.08318 (2021)
17. Fan, W., et al.: Recommender systems in the era of large language models (llms). arXiv preprint arXiv:2307.02046 (2023)
18. Fu, J., et al.: Exploring adapter-based transfer learning for recommender systems: Empir. Stud. Pract. Insights. CoRR abs/2305.15036 (2023). https://doi.org/10.48550/arXiv.2305.15036
19. Gao, C., Lin, T., Li, N., Jin, D., Li, Y.: Cross-platform item recommendation for online social e-commerce. TKDE **35**(2), 1351–1364 (2023). https://doi.org/10.1109/TKDE.2021.3098702
20. Gao, Y., Sheng, T., Xiang, Y., Xiong, Y., Wang, H., Zhang, J.: Chat-rec: towards interactive and explainable llms-augmented recommender system. arXiv preprint arXiv:2303.14524 (2023)
21. Geng, S., Liu, S., Fu, Z., Ge, Y., Zhang, Y.: Recommendation as language processing (RLP): a unified pretrain, personalized prompt & predict paradigm (P5). In: RecSys (2022)
22. Grbovic, M., Cheng, H.: Real-time personalization using embeddings for search ranking at airbnb. In: Guo, Y., Farooq, F. (eds.) Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 August 2018, pp. 311–320. ACM (2018). https://doi.org/10.1145/3219819.3219885
23. Guo, Q., et al.: A survey on knowledge graph-based recommender systems. TKDE **34**(8), 3549–3568 (2020)
24. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. TIIS **5**(4), 1–19 (2015)
25. He, R., Kang, W.C., McAuley, J.: Translation-based recommendation. In: RecSys (2017)
26. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: simplifying and powering graph convolution network for recommendation. In: SIGIR (2020)
27. He, Z., et al.: Large language models as zero-shot conversational recommenders. In: CIKM (2023)
28. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR (2016)
29. Hou, Y., He, Z., McAuley, J., Zhao, W.X.: Learning vector-quantized item representation for transferable sequential recommenders. In: WWW (2023)
30. Hou, Y., Mu, S., Zhao, W.X., Li, Y., Ding, B., Wen, J.: Towards universal sequence representation learning for recommender systems. In: KDD (2022)
31. Hua, W., Xu, S., Ge, Y., Zhang, Y.: How to index item ids for recommendation foundation models. arXiv preprint arXiv:2305.06569 (2023)
32. Jiang, A.Q., et al.: Mistral 7b. arXiv preprint arXiv:2310.06825 (2023)
33. Kang, W., McAuley, J.: Self-attentive sequential recommendation. In: ICDM (2018)
34. Kang, W.C., et al.: Do llms understand user preferences? evaluating llms on user rating prediction. arXiv preprint arXiv:2305.06474 (2023)
35. Knuth, D.E., Morris, J.H., Jr., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**(2), 323–350 (1977)
36. Li, J., Zhang, W., Wang, T., Xiong, G., Lu, A., Medioni, G.: GPT4Rec: a generative framework for personalized recommendation and user interests interpretation (2023)
37. Li, L., Zhang, Y., Chen, L.: Personalized prompt learning for explainable recommendation. TOIS **41**(4), 1–26 (2023)

38. Li, R., Deng, W., Cheng, Y., Yuan, Z., Zhang, J., Yuan, F.: Exploring the upper limits of text-based collaborative filtering using large language models: discoveries and insights. CoRR abs/2305.11700 (2023). https://doi.org/10.48550/arXiv.2305.11700
39. Li, X., Zhang, Y., Malthouse, E.C.: PBNR: prompt-based news recommender system. arXiv preprint arXiv:2304.07862 (2023)
40. Lin, G., Zhang, Y.: Sparks of artificial general recommender (AGR): early experiments with chatgpt. arXiv preprint arXiv:2305.04518 (2023)
41. Liu, J., Liu, C., Lv, R., Zhou, K., Zhang, Y.: Is ChatGPT a good recommender? a preliminary study (2023)
42. Liu, P., Zhang, L., Gulla, J.A.: Pre-train, prompt and recommendation: a comprehensive survey of language modelling paradigm adaptations in recommender systems. arXiv preprint arXiv:2302.03735 (2023)
43. Liu, Q., Chen, N., Sakai, T., Wu, X.M.: A first look at llm-powered generative news recommendation. arXiv preprint arXiv:2305.06566 (2023)
44. Lu, Y., Bartolo, M., Moore, A., Riedel, S., Stenetorp, P.: Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. In: ACL (2022)
45. Man, T., Shen, H., Jin, X., Cheng, X.: Cross-domain recommendation: An embedding and mapping approach. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 2464–2470. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/343
46. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: EMNLP, pp. 188–197 (2019)
47. Ouyang, L., et al.: Training language models to follow instructions with human feedback. NeurIPS **35**, 27730–27744 (2022)
48. Ren, X., et al.: Representation learning with large language models for recommendation. arXiv preprint arXiv:2310.15950 (2023)
49. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: UAI (2009)
50. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (2009)
51. Roitero, K., Carterette, B., Mehrotra, R., Lalmas, M.: Leveraging behavioral heterogeneity across markets for cross-market training of recommender systems. In: Seghrouchni, A.E.F., Sukthankar, G., Liu, T., van Steen, M. (eds.) WWW, pp. 694–702. ACM/IW3C2 (2020). https://doi.org/10.1145/3366424.3384362
52. Sanh, V., et al.: Multitask prompted training enables zero-shot task generalization. In: ICLR (2022)
53. Shin, K., Kwak, H., Kim, K., Kim, S.Y., Ramström, M.N.: Scaling law for recommendation models: Towards general-purpose user representations. CoRR abs/2111.11294 (2021). https://arxiv.org/abs/2111.11294
54. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Chang, Y., Zhai, C., Liu, Y., Maarek, Y. (eds.) Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, 5–9 February 2018, pp. 565–573. ACM (2018). https://doi.org/10.1145/3159652.3159656
55. Taori, R., et al.: Stanford alpaca: an instruction-following llama model (2023)
56. Touvron, H., et al.: Llama: open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)

57. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
58. Wang, J., Yuan, F., Cheng, M., Jose, J.M., Yu, C.: Beibei kong, zhijin wang, bo hu, and zang li. 2022. transrec: learning transferable recommendation from mixture-of-modality feedback. arXiv preprint arXiv:2206.06190 (2022)
59. Wang, L., Lim, E.P.: Zero-shot next-item recommendation using large pretrained language models. arXiv preprint arXiv:2304.03153 (2023)
60. Wang, W., Lin, X., Feng, F., He, X., Chua, T.S.: Generative recommendation: towards next-generation recommender paradigm. arXiv preprint arXiv:2304.03516 (2023)
61. Wang, X., Tang, X., Zhao, W.X., Wang, J., Wen, J.R.: Rethinking the evaluation for conversational recommendation in the era of large language models. arXiv preprint arXiv:2305.13112 (2023)
62. Wang, X., Zhou, K., Wen, J., Zhao, W.X.: Towards unified conversational recommender systems via knowledge-enhanced prompt learning. In: KDD (2022)
63. Wang, Y., et al.: Recmind: large language model powered agent for recommendation. arXiv preprint arXiv:2308.14296 (2023)
64. Wei, J., et al.: Finetuned language models are zero-shot learners. In: ICLR (2022)
65. Wei, W., et al.: Llmrec: large language models with graph augmentation for recommendation. In: WSDM (2024)
66. Wu, L., et al.: A survey on large language models for recommendation. arXiv preprint arXiv:2305.19860 (2023)
67. Xiao, S., et al.: Training large-scale news recommenders with pretrained language models in the loop. In: Zhang, A., Rangwala, H. (eds.) KDD 2022: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022, pp. 4215–4225. ACM (2022). https://doi.org/10.1145/3534678.3539120
68. Yuan, F., He, X., Karatzoglou, A., Zhang, L.: Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In: Huang, J.X., et al. (eds.) SIGIR (2020)
69. Yuan, F., Zhang, G., Karatzoglou, A., Jose, J.M., Kong, B., Li, Y.: One person, one model, one world: learning continual user representation without forgetting. In: Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., Sakai, T. (eds.) SIGIR (2021)
70. Zang, T., Zhu, Y., Liu, H., Zhang, R., Yu, J.: A survey on cross-domain recommendation: taxonomies, methods, and future directions. ACM Trans. Inf. Syst. **41**(2), 42:1–42:39 (2023). https://doi.org/10.1145/3548455
71. Zhang, J., Bao, K., Zhang, Y., Wang, W., Feng, F., He, X.: Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. arXiv preprint arXiv:2305.07609 (2023)
72. Zhang, J., et al.: Agentcf: collaborative learning with autonomous language agents for recommender systems. arXiv preprint arXiv:2310.09233 (2023)
73. Zhang, J., Xie, R., Hou, Y., Zhao, W.X., Lin, L., Wen, J.R.: Recommendation as instruction following: a large language model empowered recommendation approach. arXiv preprint arXiv:2305.07001 (2023)
74. Zhang, Z., Wang, B.: Prompt learning for news recommendation. arXiv preprint arXiv:2304.05263 (2023)
75. Zhao, C., Li, C., Xiao, R., Deng, H., Sun, A.: CATN: cross-domain recommendation for cold-start users via aspect transfer network. In: Huang, J.X., et al. (eds.) SIGIR, pp. 229–238. ACM (2020). https://doi.org/10.1145/3397271.3401169

76. Zhao, W.X., Lin, Z., Feng, Z., Wang, P., Wen, J.R.: A revisiting study of appropriate offline evaluation for top-n recommendation algorithms. ACM Trans. Inf. Syst. **41**(2), 1–41 (2022)
77. Zhao, W.X., et al.: Recbole: towards a unified, comprehensive and efficient framework for recommendation algorithms. In: CIKM (2021)
78. Zhao, W.X., et al.: A survey of large language models. arXiv preprint arXiv:2303.18223 (2023)
79. Zhao, Z., Wallace, E., Feng, S., Klein, D., Singh, S.: Calibrate before use: improving few-shot performance of language models. In: ICML (2021)
80. Zheng, B., Hou, Y., Lu, H., Chen, Y., Zhao, W.X., Wen, J.R.: Adapting large language models by integrating collaborative semantics for recommendation. arXiv preprint arXiv:2311.09049 (2023)
81. Zhou, K., et al.: S3-rec: self-supervised learning for sequential recommendation with mutual information maximization. In: CIKM (2020)
82. Zhu, F., Chen, C., Wang, Y., Liu, G., Zheng, X.: DTCDR: a framework for dual-target cross-domain recommendation. In: Zhu, W., et al. (eds.) CIKM, pp. 1533–1542. ACM (2019). https://doi.org/10.1145/3357384.3357992
83. Zhu, F., Wang, Y., Chen, C., Liu, G., Zheng, X.: A graphical and attentional framework for dual-target cross-domain recommendation. In: Bessiere, C. (ed.) IJCAI, pp. 3001–3008. ijcai.org (2020). https://doi.org/10.24963/ijcai.2020/415
84. Zhu, F., Wang, Y., Chen, C., Zhou, J., Li, L., Liu, G.: Cross-domain recommendation: challenges, progress, and prospects. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event/Montreal, Canada, 19–27 August 2021, pp. 4721–4728. ijcai.org (2021). https://doi.org/10.24963/ijcai.2021/639
85. Zhu, Y., et al.: Personalized transfer of user preferences for cross-domain recommendation. In: Candan, K.S., Liu, H., Akoglu, L., Dong, X.L., Tang, J. (eds.) WSDM, pp. 1507–1515. ACM (2022). https://doi.org/10.1145/3488560.3498392