STAR: A Simple Training-free Approach for Recommendations using Large Language Models

Anonymous ACL submission

Abstract

Recent progress in large language models (LLMs) offers promising new approaches for recommendation system tasks. While the current state-of-the-art methods rely on fine-tuning LLMs to achieve optimal results, this process is costly and introduces significant engineering complexities. Conversely, methods that directly use LLMs without additional fine-tuning result in a large drop in recommendation quality, often due to the inability to capture collaborative information. In this paper, we propose a **S**imple **T**raining-free **A**pproach for **R**ecommendation (**STAR**), a framework that utilizes LLMs and can be applied to various recommendation tasks without the need for fine-tuning, while maintaining high quality recommendation performance. Our approach involves a retrieval stage that uses semantic embeddings from LLMs combined with collaborative user information to retrieve candidate items. We then apply an LLM for pairwise ranking to enhance next-item prediction. Experimental results on the Amazon Review dataset show competitive performance for next item prediction, even with our retrieval stage alone. Our full method achieves Hits@10 performance of +23.8% on *Beauty*, +37.5% on *Toys & Games*, and -1.8% on *Sports & Outdoors* relative to the best supervised models. This framework offers an effective alternative to traditional supervised models, highlighting the potential of LLMs in recommendation systems without extensive training or custom architectures.

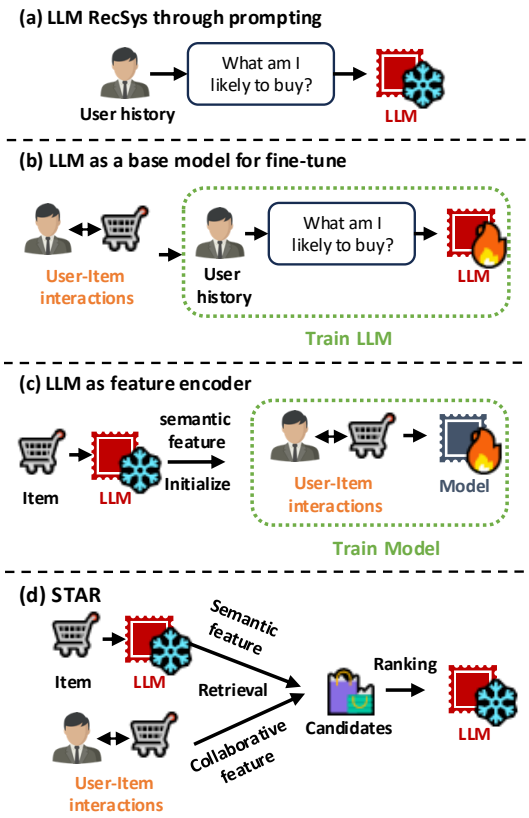


Figure 1: **A Motivating Example.** LLMs can be utilized in RecSys through (a) prompting, (b) fine-tuning on user-item interactions, and (c) using LLMs as feature encoders for training subsequent models. However, (a) cannot leverage collaborative knowledge, while (b) and (c) require extensive training and large-scale interaction data. Our framework STAR integrates collaborative knowledge into LLMs without additional training.

1 Introduction

Recent advances in large language models (LLMs) present new opportunities for addressing recommendation tasks (Brown et al., 2020; Team et al., 2023; Lin et al., 2023; Zhao et al., 2023; Li et al., 2023b; Chen et al., 2024; Tsai et al., 2024; Wu et al., 2024). Current approaches primarily leverage LLMs in three ways: (1) advanced *prompting* (Wang and Lim, 2023; Wang et al., 2023; Hou et al., 2024; Wang et al., 2024b; Xu et al., 2024a; Zhao et al., 2024; Liang et al., 2024); (2) as base models for direct *fine-tuning* (Geng et al., 2022; Zhang et al., 2023; Bao et al., 2023; Xu et al., 2024b; Tan et al., 2024; Kim et al., 2024); and (3) as feature encoders for *fine-tuning* subsequent generative models (Hou et al., 2023; Singh et al.,

2023; Rajput et al., 2024; Zheng et al., 2024) or sequential models (Sun et al., 2019; Yuan et al., 2023; Hu et al., 2024). While *prompting* LLMs for scoring and ranking utilizes their reasoning abilities, these models often largely underperform compared to fine-tuned approaches due to the absence of **collaborative** knowledge derived from user-item interactions. Conversely, using LLMs for *fine-tuning*, whether as base models or feature encoders, enhances model performance by leveraging their strong **semantic** understanding. However, this requires extensive training and large-scale interaction data.

The main objective of this work is to integrate the semantic capabilities of LLMs with collaborative knowledge from user-item interaction data, all without requiring additional training. To achieve this, we present a **S**imple **T**raining-free **A**pproach for **R**ecommendation (**STAR**) framework using LLMs. The STAR framework involves two stages: **Retrieval** and **Ranking**. The **Retrieval** stage scores new items using a combination of *semantic similarity* and *collaborative commonality* to the items in a user's history. Here, we utilize LLM-based embeddings to determine semantic similarity. Additionally, a *temporal* factor gives priority to user's recent interactions, and a *rating* factor aligns with user preferences to rank items within a specific set (§3.2). The **Ranking** stage leverages the reasoning capabilities of LLMs to adjust the rankings of the initially retrieved candidates. Specifically, we assess various LLM-based ranking approaches, including point-wise, pair-wise, and list-wise methods, while also determining the key information needed for the LLM to better understand user preferences and make accurate predictions (§3.3). Our experimental evaluation shows competitive performance across a diverse range of recommendation datasets, all without the need for supervised training or the development of custom-designed architectures.

We present extensive experimental results on the Amazon Review dataset (McAuley et al., 2015; He and McAuley, 2016). Our findings are as follow:

1. Our retrieval pipeline itself, comprised of both semantic relationship and collaborative information, achieves Hits@10 performance of +17.3% on *Beauty*, +26.2% on *Toys & Games*, and -5.5% on *Sports & Outdoors* relative to the best supervised models.

2. We show that pair-wise ranking further improves upon our retrieval performance, while point-wise and list-wise methods struggle to achieve similar improvements.

3. We illustrate that collaborative information is a critical component that adds additional benefits to the semantic information throughout our system, in both the retrieval and ranking stages.

These findings show that it is possible to build a recommendation system utilizing LLMs without additional fine-tuning that can significantly close the quality gap of fully fine-tuned systems, and in many cases even achieve higher quality.

2 Related Works

Recent studies have explored the role of LLMs in recommendation systems through three primary approaches: (1) using prompting for scoring and ranking, (2) fine-tuning as a base model, and (3) serving as a feature encoder.

LLM prompting for scoring and ranking. LLMs can generate recommendations by understanding user preferences or past interactions expressed in natural language.

This is typically achieved through generative selection prompting, where the model ranks and selects top items from a given candidate set (Wang and Lim, 2023; Wang et al., 2023; Hou et al., 2024; Wang et al., 2024b; Xu et al., 2024a; Zhao et al., 2024; Liang et al., 2024). Another line of work applies ranking prompting (Dai et al., 2023), inspired by LLM-based ranking in information retrieval (Zhu et al., 2023; Wang et al., 2024a), using point-wise (Liang et al., 2022; Zhuang et al., 2023), pair-wise (Qin et al., 2024), or list-wise (Sun et al., 2023; Qin et al., 2024) approaches. However, LLM prompting alone, without a retrieval stage to pre-select candidate items for scoring and ranking, is less effective than fine-tuned models due to lack of collaborative knowledge derived from user-item interaction data. As a result, many approaches use a fine-tuned model for candidate retrieval and an LLM for ranking. However, in this setup, overall performance is primarily determined by the retrieval quality rather than the LLM itself.

LLM as base model for fine-tuning. To integrate collaborative knowledge with the semantic understanding of LLMs, recent studies have explored fine-tuning using user-item interaction data.

While this improves recommendation performance, it requires extensive training and large-scale interaction datasets (Geng et al., 2022; Zhang et al., 2023; Bao et al., 2023; Xu et al., 2024b; Tan et al., 2024; Kim et al., 2024).

LLM as feature encoder. LLMs can also be used as text encoders to capture richer semantic information from item metadata and user profiles (Reimers and Gurevych, 2019; Cer et al., 2018; Ni et al., 2021; Lee et al., 2024). To further optimize these representations, researchers have explored several approaches: (1) mapping continuous LLM embeddings into discrete tokens using vector quantization and training a subsequent generative model (Hou et al., 2023; Singh et al., 2023; Rajput et al., 2024; Zheng et al., 2024); (2) training sequential models by initializing the embedding layer with LLM embeddings (Sun et al., 2019; Yuan et al., 2023; Hu et al., 2024); and (3) training models to directly compute the relevance between item and user embeddings (*i.e.,* embeddings of user selected items) (Ding et al., 2022; Hou et al., 2022; Gong et al., 2023; Li et al., 2023a; Liu et al., 2024; Li et al., 2024; Ren et al., 2024; Sheng et al., 2024).

In this work, we utilize LLMs as both feature encoders and ranking functions by integrating collaborative knowledge from user-item interaction data. Our findings show that LLM embeddings can serve as effective item representations, achieving strong results in sequential recommendation tasks without extensive optimization. This aligns with (Harte et al., 2023) but differs in our use of novel scoring rules that incorporate both collaborative and temporal information.

3 STAR: Simple Training-Free RecSys

This section initially outlines the problem formulation (§3.1). Subsequently, we detail the proposed retrieval (§3.2) and ranking pipelines (§3.3).

3.1 Sequential Recommendation

The sequential recommendation task predicts the next item a user will interact with based on their interaction history. For a user $u \in U$, where U is the set of all users, the interaction history is represented as a sequence of items $S_u = \{s_1, s_2, \ldots, s_n\}$, with each item $s_i \in I$ belonging to the set of all items I. Each user history item s_i is associated with a rating $r_i \in \{1, 2, 3, 4, 5\}$ given by the user u. The goal is to predict the next item $s_{n+1} \in I$ that the user is most likely to interact with.

3.2 Retrieval Pipeline

The retrieval pipeline aims to assign a score to an unseen item $x \in I$ given the sequence S_u. To achieve this, we build two scoring components: one that focuses on the **semantic** relationship between items and another that focuses on the **collaborative** relationship.

Semantic relationship. Understanding how similar a candidate item is to the items in a user's interaction history $s_i \in S_u$ is key to accurately gauging how well candidate items align with user preferences. Here we leverage LLM embedding models, where we pass in text prompts representing items and collect embedding vectors of dimension d_e. We construct a prompt based on the item information and metadata, which can include fields like *title*, *description*, *category*, *brand*, *sales ranking*, *price*, etc. (See Appendix A.3 for the full prompt details). We collect embeddings for each item $i \in I$, resulting in $E \in \mathbb{R}^{n \times d_e}$, where n is number of total items in I.

The semantic relationship between two items (i_a, i_b) is then calculated using the cosine similarity between their embeddings $E_{i_a}, E_{i_b} \in E$. This measure provides a numerical representation of how closely related the items are in semantic space. For our experiments, we precompute the entire semantic relationship matrix $R_{\text{S}} \in \mathbb{R}^{n \times n}$. For many domains, this is a practical solution. However, if $|I|$ is very large, Approximate Nearest Neighbor methods (Guo et al., 2020; Sun et al., 2024) are efficient approaches to maintain quality and reduce computation.

Collaborative relationship. Semantic similarity between a candidate item and items in a user's interaction history is a helpful cue for assessing the similarity of items based on the item information. However, this alone does not fully capture the engagement interactions of items by multiple users. To better understand the collaborative relationship, we consider how frequently different combinations of items are interacted with by users. These shared interaction patterns can provide strong indicators of how likely the candidate item is to resonate with a broader audience with similar preferences. For each item $i \in I$, we derive an interaction array that represents user interactions, forming a set of sparse user-item interaction arrays $C \in \mathbb{R}^{n \times m}$, where m is number of users in U. The collaborative relationship between two items (i_a, i_b) is then computed

Figure 2: **STAR Framework overview.** We use the semantic relationship scores in R_S and the collaborative relationship scores in R_C to score the items in the user history compared to new items to recommend. The final score for one new item is a weighted average from the semantic relationship and collaborative relationship scores, with additional weights from the user's ratings r and a temporal decay $\lambda < 1$ which prioritize recent interactions. The top scoring retrieved items are sent to the LLM Ranking, where we can use point-wise, pair-wise, or list-wise ranking approaches to further improve upon the scoring of recommended items.

by using the cosine similarity between their sparse arrays $C_{i_a}, C_{i_b} \in C$, capturing the normalized co-occurrence of the items. To streamline the process, we pre-compute and store these values in a collaborative relationship matrix $R_C = \frac{C \cdot C^\top}{\|C\|\|C^\top\|} \in \mathbb{R}^{n \times n}$, which is typically very sparse.

Scoring rules. The score for an unseen item $x \in I$ is calculated by averaging both the semantic and collaborative relationships between items in $S_u = \{s_1, s_2, \ldots, s_n\}$ as follows:

$$\texttt{score}(x) = \frac{1}{n} \sum_{j=1}^{n} r_j \lambda^{t_j} \left[a R_S^{xj} + (1-a) R_C^{xj} \right] \tag{1}$$

where R_S^{xj} and R_C^{xj} represent the semantic and collaborative relationships between the unseen item x and item $s_j \in S_u$, respectively. In this equation, r_j is the rating given by user u to item s_j, and λ^{t_j} is an exponential decay function applied to the temporal order t_j of s_j in the sequence S_u. Here, t_j is set to 1 for the most recent item in S_u and increments by 1 up to n for the oldest item. The framework, illustrated in Figure 2, outputs the top k items in descending order based on their scores.

3.3 Ranking Pipeline

After retrieving the top k items, denoted as I_k, from the initial retrieval process, a LLM is employed to further rank these items to enhance the overall next-item recommendation quality. The items in I_k are already ordered based on scores from the retrieval framework, which reflect *semantic*, *collaborative*, and *temporal* information. We intentionally incorporate this initial order into the ranking process to enhance both efficiency and effectiveness. This framework then leverages the capabilities of the LLM to better capture user preference, complex relationships and contextual relevance among the items.

3.3.1 Rank schema

We present three main strategies for ranking: (1) **Point-wise** evaluates each item $x \in I_k$ independently, based on the user sequence S_u, to determine how likely it is that user u will interact with item x. If two items receive the same score, their rank follows the initial order from I_k; (2) **Pair-wise** evaluates the preference between two items $x_i, x_j \in I_k$ based on the user sequence S_u. We adopt a sliding window approach, starting from the items with the lowest retrieval score at the bottom of the list (Qin et al., 2024). The LLM compares and swaps adjacent pairs, while iteratively stepping the comparison window one element at a time. (3) **List-wise** evaluates the preference among multiple items $x_i, \ldots, x_{i+w} \in I_k$ based on the user sequence S_u. This method also uses a sliding window approach, with a window size w and a stride d to move the window across the list, refining the

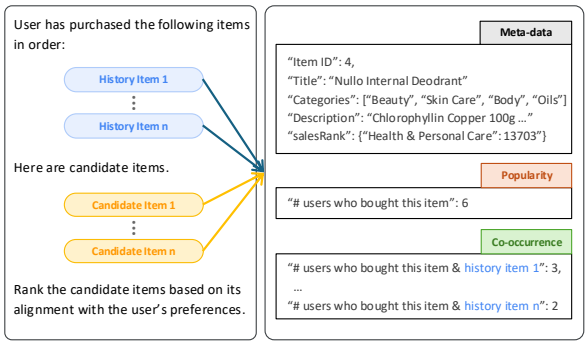


Figure 3: **Prompt overview for the ranking pipeline.** The prompt includes history items, candidate items, and instructions for the ranking strategy. Each item is represented by metadata, along with additional details such as popularity and co-occurrence, formatted in JSON. Full prompt is available in Appendix A.4.

ranking as it passes (Sun et al., 2023). In this setup, **pair-wise** is a special case of **list-wise** with $w = 2$ and $d = 1$.

3.3.2 Item information

We represent the metadata (e.g., Item ID, title, category, etc.) for each item in the user sequence $s_j \in S_u$ and each candidate item to be ranked $x \in I_k$ as JSON format in the input prompt. Additionally, we incorporate two more types of information that can help the reasoning capabilities of the LLM: (1) **Popularity** is calculated as the number of users who have interacted with the item x, simply by counting the occurrences in the training data. This popularity value is then included in the prompt for both the items in the user sequence $s_j \in S_u$ and the candidate item to be ranked $x \in I_k$ as "*Number of users who interacted with this item: ###*"; (2) **Co-occurrence** is calculated as the number of users who have interacted with both item x and item $s_j \in S_u$. The resulting value is then included for candidate items $x \in I_k$ as "*Number of users who interacted with both this item and item s_j: ###*".

4 Experimental Setup

Datasets. We evaluate the performance using public 2014 Amazon review datasets (McAuley et al., 2015; He and McAuley, 2016). Specifically, we select the *Beauty*, *Toys and Games*, and *Sports and Outdoors* categories, as these have been used in previous studies (Geng et al., 2022; Tan et al., 2024) and provide data points for comparison (see Table 1). We follow the same data processing steps

Dataset	# Users	# Items	# Interactions	Density
Beauty	22,363	12,101	198,502	0.0734%
Toys and Games	19,412	11,924	167,597	0.0724%
Sports and Outdoors	35,598	18,357	296,337	0.0453%

Table 1: **Dataset statistics**. Density is the percentage of actual user-item interactions out of all interactions.

as in prior work, filtering out users and items with fewer than five interactions, maintaining consistent baseline settings.

Dataset Construction and Evaluation Metrics. We follow conventional supervised models, where the last item, s_n, is reserved for testing and the second to last item, s_{n-1}, is used for validation. The remaining items are used for training. For the final predictions of s_n, all training and validation items are used as input. Although our method does not train model parameters, we only use the training data to calculate the collaborative user interaction values used for R_C in retrieval and for **popularity** and **co-occurence** in ranking. We report Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HR) at ranks 5 and 10.

Compared Methods. We compare our model with following supervised trained models: (1) **KNN** is a user-based collaborative filtering method that finds the top 10 most similar users to a given user and averages their ratings to score a specific item; (2) **Caser** uses convolution neural networks to model user interests (Tang and Wang, 2018); (3) **HGN** uses hierarchical gating networks to capture both long and short-term user behaviors (Ma et al., 2019); (4) **GRU4Rec** employs GRU to model user action sequences (Hidasi et al., 2015); (5) **FDSA** uses a self-attentive model to learn feature transition patterns (Zhang et al., 2019); (6) **SASRec** uses a self-attention mechanism to capture item correlations within a user's action sequence (Kang and McAuley, 2018); (7) **BERT4Rec** applies a masked language modeling (MLM) objective for bi-directional sequential recommendation (Sun et al., 2019); (8) **S^3-Rec** extends beyond the MLM objective by pre-training with four self-supervised objectives to learn better item representations (Zhou et al., 2020). (9) **P5** fine-tunes a pre-trained LM for use in multi-task recommendation systems by generating tokens based on randomly assigned item IDs (Geng et al., 2022); (10) **TIGER** also fine-tunes LMs to predict item IDs directly, but these IDs are semantic, meaning they are learned based on the content of the items (Rajput et al., 2024);

and (11) **IDGenRec** goes further by extending semantic IDs to textual IDs, enriching the IDs with more detailed information (Tan et al., 2024).

Implementation Details. Unless otherwise specified, we use Gecko `text-embedding-004` (Lee et al., 2024)[1] to collect LLM embeddings for retrieval, and we use `gemini-1.5-flash`[2] for LLM-based ranking. All API calls were completed as of September 1, 2024.

5 Experimental Results

In this section, we present a performance analysis of our retrieval (§5.1) and ranking pipeline (§5.2).

5.1 Retrieval Pipeline

The second group of Table 2 presents the performance of our retrieval framework. `STAR-Retrieval` alone achieves the best or second-best results compared to all baselines and fine-tuned methods. There is a significant improvement across all metrics for *Toys and Games*, ranging from +26.50% to +35.3%. In *Beauty*, all metrics besides NDCG@5 (-1.2%) are improved from a range of +6.1% to +20.0%. In *Sports and Outdoors*, the results are second best to IDGenRec, trailing from a range of -19.6% to -5.57%. Furthermore, we explore the following questions in greater detail, with additional details provided in Appendix A.1:

Impact of semantic and collaborative information. We assess the impact of semantic (R_S) and collaborative (R_C) information by varying their weighting factor a. As shown in the top panel of Figure 4, the optimal performance occurs between $a = 0.5$ and 0.6, with $a = 0.5$ chosen for simplicity. The far left and right sides of that figure show that results significantly degrade when using only one component ($a = 0.0$ or $a = 1.0$), confirming that combining both enhances retrieval effectiveness.

Impact of user history length l and recency factor λ. We analyze the effect of user history length (l) and recency factor (λ) on retrieval performance. As shown in the right panel of Figure 4, performance improves with more history items up to $l = 3$ but declines beyond that. The middle panel indicates that a recency factor of $\lambda = 0.7$ effectively prioritizes recent interactions, outperforming

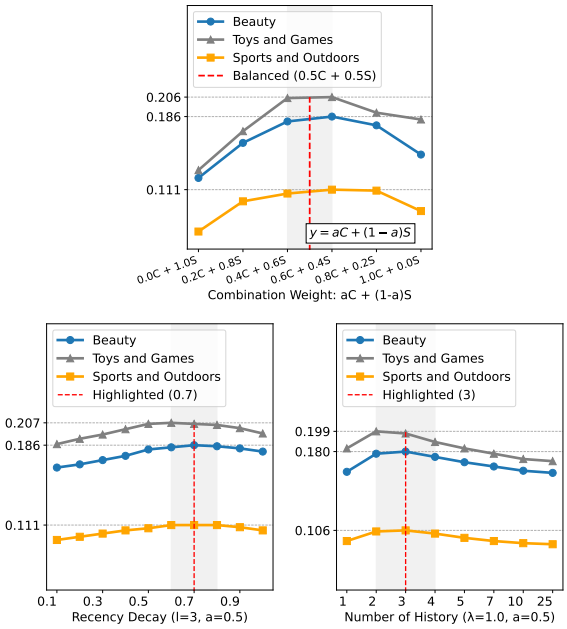


Figure 4: **Retrieval performance (Hits@50)** with different weighting factor a between R_S and R_C (**top**), recency factor λ (**bottom-left**), and number of history l (**bottom-right**). The shaded regions show the best range. $a = 0.5$, $\lambda = 0.7$, and $l = 3$ show the best.

no recency adjustment ($\lambda = 1$). However, an excessively small λ overly discounts past items, resembling the effect of using only one history item.

Impact of user rating. In Equation 1, we propose using the user ratings to help score items. To assess their impact, we compare results using actual ratings (r) versus a uniform rating ($r = 1$ for all items). Surprisingly, ignoring the rating information consistently produced better results (See Table 3). This likely due to a task mismatch—our focus is on predicting interactions, not user ratings. Consequently, in our main evaluation and the rest of the analysis, we disregard ratings (by setting $r = 1$ for all items), which is also practical since most real-world interactions lack explicit ratings.

Scoring rule analysis. Prior studies (Harte et al., 2023; Liang et al., 2024) use LLM embeddings of items in a user sequence (S_u) to retrieve candidates. These methods generate a "user embedding" by averaging item embeddings, then identify the most similar candidate based on embedding similarity. To evaluate our scoring rule, we compare it against this pooling approach. As shown in Table 4, our method outperforms pooling even without collaborative information ($S = 1.0$, $C = 0.0$). Incorporating collaborative information ($S = 0.5$, $C = 0.5$) further improves performance, highlighting the advantage of our approach.

[1] https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/text-embeddings-api

[2] https://deepmind.google/technologies/gemini/flash/

Category	Method / Model	Train	Beauty				Toys and Games				Sports and Outdoors			
			H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10
Baseline	KNN	✓	0.004	0.003	0.007	0.004	0.004	0.003	0.007	0.004	0.001	0.001	0.002	0.001
	Caser (Tang and Wang, 2018)	✓	0.021	0.013	0.035	0.018	0.017	0.011	0.027	0.014	0.012	0.007	0.019	0.010
	HGN (Ma et al., 2019)	✓	0.033	0.021	0.051	0.027	0.032	0.022	0.050	0.028	0.019	0.012	0.031	0.016
	GRU4Rec (Hidasi et al., 2015)	✓	0.016	0.010	0.028	0.014	0.010	0.006	0.018	0.008	0.013	0.009	0.020	0.011
	BERT4Rec (Sun et al., 2019)	✓	0.020	0.012	0.035	0.017	0.012	0.007	0.020	0.010	0.012	0.008	0.019	0.010
	FDSA (Zhang et al., 2019)	✓	0.027	0.016	0.041	0.021	0.023	0.014	0.038	0.019	0.018	0.012	0.029	0.016
	SASRec (Kang and McAuley, 2018)	✓	0.039	0.025	0.061	0.032	0.046	0.031	0.068	0.037	0.023	0.015	0.035	0.019
	$\mathbf{S^3}$-Rec (Zhou et al., 2020)	✓	0.039	0.024	0.065	0.033	0.044	0.029	0.070	0.038	0.025	0.016	0.039	0.020
	P5 (Zhou et al., 2020)	✓	0.016	0.011	0.025	0.014	0.007	0.005	0.012	0.007	0.006	0.004	0.010	0.005
	TIGER (Geng et al., 2022)	✓	0.045	0.032	0.065	0.038	0.052	0.037	0.071	0.043	0.026	0.018	0.040	0.023
	IDGenRec (Tan et al., 2024)	✓	0.062	0.049	0.081	0.054	0.066	0.048	0.087	0.055	**0.043**	**0.033**	**0.057**	**0.037**
STAR-Retrieval	-	✗	0.068	0.048	0.098	0.057	0.086	0.061	0.118	0.071	0.038	0.026	0.054	0.031
STAR-Ranking	point-wise	✗	0.068	0.047	0.096	0.056	0.086	0.061	0.117	0.071	0.037	0.026	0.054	0.031
	pair-wise	✗	**0.072**	**0.051**	**0.101**	**0.060**	**0.090**	**0.064**	**0.120**	**0.073**	0.040	0.028	0.056	0.034
	list-wise	✗	0.065	0.047	0.090	0.055	0.083	0.060	0.111	0.069	0.036	0.026	0.052	0.031

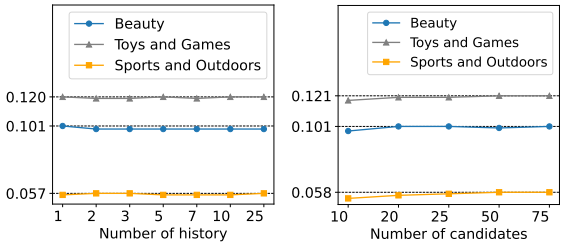
Table 2: **Performance (Hits@K, NDCG@K)** comparison among supervised models, and `STAR retrieval` & `ranking` pipeline. The first group in the table represents supervised models; The second group shows the retrieval pipeline with parameters set to an exponential decay rate of $\lambda = 0.7$, history length of $l = 3$, and a weight factor of $a = 0.5$; The third group consists of ranking pipeline which use `gemini-1.5-flash`. The best model for each dataset is shown in **bold**, and the second best is underlined.

Rating	Beauty		Toys & Games		Sports & Outdoors	
	H@10	N@10	H@10	N@10	H@10	N@10
w/ rating	0.095	0.056	0.115	0.069	0.052	0.030
w/o rating	**0.098**	**0.057**	**0.118**	**0.071**	**0.054**	**0.031**

Table 3: **Retrieval (Hits@10, NDCG@10)** comparing w/ and w/o incorporating user rating.

Scoring method	Beauty		Toys & Games		Sports & Outdoors	
	H@10	N@10	H@10	N@10	H@10	N@10
Average Pooling	0.060	0.033	0.080	0.043	0.033	0.017
STAR (S=1.0, C=0.0)	0.072	0.042	0.095	0.055	0.039	0.022
STAR (S=0.5, C=0.5)	**0.098**	**0.057**	**0.118**	**0.071**	**0.054**	**0.031**

Table 4: **Retrieval performance (Hits@10, NDCG@10)** comparison between `STAR-retrieval` pipeline and average embedding pooling. S representing Semantic Information and C representing Collaborative Information weightings.



(a) Number of history l (b) Number of candidates k

Figure 5: **Pair-wise ranking performance (Hits@10)** trend by different number of history l and number of candidates k

5.2 Ranking Pipeline

The third group of Table 2 highlights the performance of the ranking framework, which improves upon the retrieval stage results. Pair-wise ranking improves all metrics over `STAR-Retrieval` performance by +1.7% to +7.9%. This further improves the results over other baselines for *Beauty* and *Toys and Games*, while closing the gap on IDGenRec in *Sports and Outdoors*. Point-wise and list-wise methods struggle to achieve similar improvements. We examine the following questions in more detail, with additional information available in Appendix A.2.

Effectiveness of ranking and impact of window size and stride. Previous approaches use a *selection prompt*, instructing the LLM to choose the top-k items from a candidate set (Wang and Lim, 2023; Hou et al., 2024; Wang et al., 2024b). In contrast, our method uses a *ranking* prompt, which explicitly instructs the LLM to rank all items within the available context window. We assess the effectiveness of the ranking approach in comparison to the selection approach and a point-wise prompt. As shown in Table 5, results show that ranking with a small window size (*e.g.,* pair-wise or list-wise with a window of 4) consistently outperforms selection and pair-wise approaches. These findings align with prior work in document retrieval, where list-wise ranking with large windows or reasoning prompts is challenging for LLMs due to the need for task-specific knowledge. In contrast, pair-wise ranking is relatively easier and can yield better performance, even for smaller LMs (Qin et al., 2024).

Impact of candidate count (k) and history length (l). As shown in Figure 5, varying the number of historical items (l) has little effect on performance. In contrast, increasing the number of candidate items (k) slightly improves accuracy by increasing the chances of including the correct item. However, the performance gains are minimal, and

Prompt Style	Window Size	Stride	Beauty				Toys and Games				Sports and Outdoors			
			H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10	H@5	N@5	H@10	N@10
None (STAR-Retrieval)	-	-	0.0684	0.0480	0.0977	0.0574	0.0857	0.0606	0.1176	0.0709	0.0379	0.0262	0.0542	0.0314
Selection	-	-	0.0691	0.0484	0.0958	0.0570	0.0841	0.0613	0.1109	0.0699	0.0376	0.0269	0.0520	0.0316
Point-wise	1	1	0.0685	0.0472	0.0956	0.0558	0.0855	0.0611	0.1170	0.0713	0.0370	0.0257	0.0539	0.0312
Pair-wise	2	1	0.0716	**0.0506**	**0.1008**	**0.0600**	**0.0899**	**0.0639**	0.1196	**0.0734**	0.0401	**0.0283**	**0.0564**	**0.0335**
List-wise	4	2	**0.0724**	0.0502	0.1002	0.0592	0.0894	0.0634	0.1195	0.0732	**0.0406**	0.0282	0.0559	0.0331
	8	4	0.0688	0.0484	0.0988	0.0581	0.0874	0.0625	**0.1202**	0.0731	0.0388	0.0276	0.0556	0.0330
	10	5	0.0676	0.0480	0.0981	0.0578	0.0853	0.0616	0.1201	0.0728	0.0379	0.0270	0.0558	0.0327
	20	-	0.0653	0.0471	0.0903	0.0551	0.0829	0.0603	0.1113	0.0694	0.0364	0.0262	0.0518	0.0311

Table 5: **Ranking performance (Hits@K, NDCG@K)** by window size and stride. Here we use 20 candidates from the retrieval stage. The best prompt for each dataset is shown in **bold**, and the second best is underlined.

Item prompt	Beauty		Toys & Games		Sports & Outdoors	
	H@10	N@10	H@10	N@10	H@10	N@10
Metadata	0.1000	0.0567	0.1193	0.0690	0.0544	0.0315
+ popularity	0.0998	0.0564	0.1174	0.0701	0.0549	0.0316
+ co-occurrence	**0.1008**	**0.0600**	0.1196	0.0734	**0.0564**	**0.0335**
+ popularity, co-occurrence	0.0999	0.0599	**0.1203**	**0.0736**	0.0550	0.0322

Table 6: **Pair-wise ranking (Hits@10 & NDCG@10)** by varying information in the item prompt.

Shuffle Candidates	LLM Ranking	Beauty		Toys & Games		Sports & Outdoors	
		H@10	N@10	H@10	N@10	H@10	N@10
✗	✗	0.0977	0.0574	0.1176	0.0709	0.0542	0.0314
✓	✗	0.0687	0.0312	0.0779	0.0349	0.0371	0.0169
✗	✓	0.1008	0.0600	0.1196	0.0734	0.0564	0.0335
✓	✓	0.0793	0.0485	0.0949	0.0596	0.0452	0.0275

Table 7: **Ranking (Hits@10 & NDCG@10)** comparison with random shuffling of the retrieved items.

given the high computational cost of processing more candidates, the trade-off may not be worthwhile.

Impact of additional item information in prompts. We analyze the effect of adding extra item information—specifically, popularity and co-occurrence data—alongside item metadata in prompts. As shown in Table 6, incorporating co-occurrence data improves NDCG@10 by +0.2% to +3.4% compared to using metadata alone. In contrast, adding popularity information does not enhance performance and sometimes even reduces it. This suggests that popularity bias does not help LLMs make better ranking decisions, consistent with prior research showing its ineffectiveness in recommendation tasks (Abdollahpouri and Mansoury, 2020; Abdollahpouri et al., 2019; Zhu et al., 2021).

Impact of candidate order. We investigate whether the order of the retrieval candidate items affects the ranking performance of our recommendation system. To assess this, we conducted an experiment comparing the pairwise ranking outcomes between two sets of top 20 candidates: (1) Random order: Candidates were randomly shuffled; (2) Retrieval Order: Candidates were ordered based on their scores from the retrieval pipeline. As shown in Table 7, ordering the candidates according to their retrieval pipeline scores significantly improves ranking performance compared to a random arrangement. Comparing rows 2 and 4 show that our pair-wise ranking can improve results of a randomly shuffled list. However, the results are even better when the candidate list is ranked by the retrieval score (row 3). More analysis needs to be done to determine if $O(n \log n)$ or $O(n^2)$ comparisons could better rank a randomly shuffled candidate list compared to a sliding window approach, although this would come at an even higher computation cost.

6 Conclusion

In this paper, we introduced a **S**imple **T**raining-free **A**pproach for **R**ecommendation (**STAR**) that uses the power of large language models (LLMs) to create a generalist framework applicable across multiple recommendation domains. Our method comprises two key stages: a retrieval phase and a ranking phase. In the retrieval stage, we combine semantic embeddings from LLMs with collaborative user information to effectively select candidate items. In the ranking stage, we apply LLMs to enhance next-item prediction and refine the recommendations. Experimental results on a large-scale Amazon review dataset demonstrate that our retrieval method alone outperforms most supervised models. By employing LLMs in the ranking stage, we achieve further improvements. Importantly, our study highlights that incorporating collaborative information is critical in both stages to maximize performance. Our findings reveal that LLMs can effectively function as generalists in recommendation tasks without requiring any domain-specific fine-tuning. This opens up exciting possibilities for developing versatile and efficient recommendation systems that are readily adaptable across diverse domains.

7 Limitations

The **STAR** framework presents an effective alternative to traditional supervised models, showcasing the potential of LLMs in recommendation systems without the need for extensive training or custom architectures. However, several limitations remain, which also indicate directions for future improvement:

Importance of item modality and enriched item meta-data. The **STAR** framework's ability to capture semantic relationships between items relies significantly on the presence of rich item text meta-data. Without such meta-data and with only user-item interaction data available, the framework's semantic relationship component will be less effective. To maximize the use of semantic relationships between items, future work should explore incorporating additional modalities, such as visual or audio data, to generate more comprehensive semantic representations of items, fully utilizing all the available information.

Improving Retrieval Simplicity and Scalability. Although our work demonstrates the effectiveness of a general training-free framework, the current method requires different choices for parameters. In future work, we will explore ways to either reduce the number of parameters choices or select values more easily. In our current implementation, we compute the full set of item-item comparisons for both the semantic and collaborative information. This computation is infeasible if the item set is too large. In future work, we will run experiments to measure how effective approximate nearest neighbor methods are at reducing computation and maintaining retrieval quality.

Beyond LLM ranking. The importance of our work highlights that high quality results can be achieved without additional fine-tuning. However, in the current method, our **STAR** ranking pipeline utilizes costly LLM calls that would result in high latency. This may be a suitable solution to use in offline scenarios, but would be prohibitive to serve large-scale and real-time user traffic. Future work needs to explore how we can improve efficiency, such as using a mix of pair-wise and list-wise ranking. Our work shows a promising first step to creating high quality, training-free, and general recommendation systems.

References

Himan Abdollahpouri and Masoud Mansoury. 2020. Multi-sided exposure bias in recommendation. *arXiv preprint arXiv:2006.15772*.

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2019. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286*.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. 2024. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web*, 27(4):42.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1126–1132.

Hao Ding, Anoop Deoras, Yuyang (Bernie) Wang, and Hao Wang. 2022. Zero shot recommender systems. In *ICLR 2022 Workshop on Deep Generative Models for Highly Structured Data*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Yuqi Gong, Xichen Ding, Yehui Su, Kaiming Shen, Zhongyi Liu, and Guannan Zhang. 2023. An unified search and recommendation foundation model for cold-start scenario. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4595–4601.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR.

Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1096–1102.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 585–593.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, pages 364–381. Springer.

Jun Hu, Wenwen Xia, Xiaolu Zhang, Chilin Fu, Weichang Wu, Zhaoxin Huan, Ang Li, Zuoli Tang, and Jun Zhou. 2024. Enhancing sequential recommendation via llm-based semantic embedding learning. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 103–111.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. *arXiv preprint arXiv:2404.11343*.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023a. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1258–1267.

Juanhui Li, Haoyu Han, Zhikai Chen, Harry Shomer, Wei Jin, Amin Javari, and Jiliang Tang. 2024. Enhancing id and text fusion via alternative training in session-based recommendation. *arXiv preprint arXiv:2402.08921*.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2023b. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Yueqing Liang, Liangwei Yang, Chen Wang, Xiongxiao Xu, Philip S Yu, and Kai Shu. 2024. Taxonomy-guided zero-shot recommendations with llms. *arXiv preprint arXiv:2406.14043*.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, et al. 2023. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817*.

Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 452–461.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 825–833.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting.

In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, Mexico City, Mexico. Association for Computational Linguistics.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2024. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, pages 3464–3475.

Leheng Sheng, An Zhang, Yi Zhang, Yuxin Chen, Xiang Wang, and Tat-Seng Chua. 2024. Language models encode collaborative signals in recommendation. *arXiv preprint arXiv:2407.05441*.

Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, et al. 2023. Better generalization with semantic ids: A case study in ranking for recommendations. *arXiv preprint arXiv:2306.08121*.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450.

Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2024. Soar: improved indexing for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 36.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Idgenrec: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Alicia Tsai, Adam Kraft, Long Jin, Chenwei Cai, Anahita Hosseini, Taibai Xu, Zemin Zhang, Lichan Hong, Ed Chi, and Xinyang Yi. 2024. Leveraging LLM reasoning enhances personalized recommender systems. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13176–13188, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Large search model: Redefining search stack in the era of llms. In *ACM SIGIR Forum*, volume 57, pages 1–16. ACM New York, NY, USA.

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024b. RecMind: Large language model powered agent for recommendation. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364, Mexico City, Mexico. Association for Computational Linguistics.

Yu Wang, Zhiwei Liu, Jianguo Zhang, Weiran Yao, Shelby Heinecke, and Philip S Yu. 2023. Drdt: Dynamic reflection with divergent thinking for llm-based sequential recommendation. *arXiv preprint arXiv:2312.11336*.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Lanling Xu, Junjie Zhang, Bingqian Li, Jinpeng Wang, Mingchen Cai, Wayne Xin Zhao, and Ji-Rong Wen. 2024a. Prompting large language models for recommender systems: A comprehensive framework and empirical analysis. *arXiv preprint arXiv:2401.04997*.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024b. Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 386–394.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR*

Conference on Research and Development in Information Retrieval, pages 2639–2649.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001*.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, Xiaofang Zhou, et al. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326.

Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten de Rijke. 2024. Let me do it for you: Towards llm empowered recommendation via tool learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1806.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. 2021. Popularity bias in dynamic recommendation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2439–2449.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Berdersky. 2023. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. *arXiv preprint arXiv:2310.14122*.

A Appendix

A.1 Retrieval Performance Analysis

LLM embedding performance comparison on retrieval. We test different LLM embedding APIs with different embedding size to understand the impact of LLM embeddings capturing item semantic similarity . Figure 6 illustrates the performance differences between the Gecko `text-embedding-004` (Lee et al., 2024), OpenAI `text-embedding-3-small`, and `text-embedding-3-large` models[3]. Overall, higher-dimensional embeddings and larger models performed better, indicating that enhanced semantic representation capabilities lead to improved semantic relationship capture.

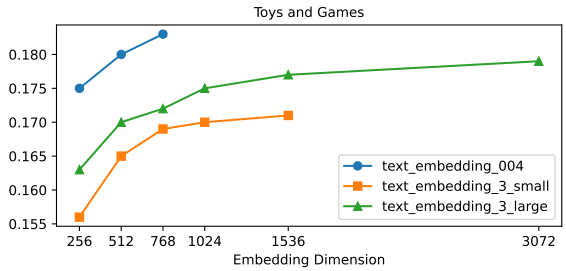


Figure 6: **Retrieval performance (Hits@50)** comparison by embedding APIs for Toys and Games. The models text-embedding-004, text-embedding-3-small, and text-embedding-3-large each have a maximum dimension of 768, 1536, and 3072, respectively.

A.2 Ranking Performance Analysis

LLM performance comparison on ranking. To evaluate how model capabilities impact ranking performance, we compare four models: `gemini-1.5-flash`, `gemini-1.5-pro`, `gpt-4o-mini`, and `gpt-4o`. In Table 8, we observe the larger models (`gemini-1.5-pro` and `gpt-4o`) tend to outperform their smaller counterparts, although the performance differences are minimal. This finding aligns with results from other studies (Qin et al., 2024), which suggest that increased model capability has limited influence on pairwise ranking tasks. Despite being more computationally expensive, pairwise ranking methods tend to be more robust than alternative approaches.

Model	Beauty		Toys & Games		Sports & Outdoors	
	H@10	N@10	H@10	N@10	H@10	N@10
`gemini-1.5-flash`	**0.101**	0.060	0.120	0.073	**0.056**	0.034
`gemini-1.5-pro`	0.100	0.060	0.120	**0.075**	0.056	**0.034**
`gpt-4o-mini`	0.100	0.058	0.120	0.072	0.056	0.033
`gpt-4o`	0.100	**0.060**	**0.121**	0.074	0.056	0.033

Table 8: **Pair-wise ranking performance (Hits@10 & NDCG@10)** comparison by different models.

[3]https://platform.openai.com/docs/guides/embeddings/

A.3 Item Encoding Prompt for Retrieval Example

Below is an example of an item prompt for encoding with an LLM embedding API. This example includes fields for *title*, *description*, *category*, *brand*, *sales ranking*, and *price*. We omit metadata fields like *Item ID* and *URL*, as those fields contain strings that can contain spurious lexical similarity (e.g., IDs: "000012", "000013' or URLs: "https://abc.com/uxrl", "https://abc.com/uxrb") and can reduce the uniformity of the embedding space and make it difficult to distinguish between semantically different items

```
description:
    LENGTH: 70cm / 27.55 inches
    Color: Mix Color
    EST. SHIPPING WT.: 310g
    Material: Synthetic High Temp Fiber
    Cap Construction: Capless
    Cap Size: Average
    1. The size is adjustable and no pins or tape should be required. It should fit most people.
       Adjust the hooks inside the cap to suit your head.
    2. Please be aware that colors might look slightly different in person due to
       camera quality and monitor settings. Stock photos are taken in natural light with no flash.
    3. Please ask all questions prior to purchasing. I will replace defective items.
       Indicate the problem before returning.
       A 30-day return/exchange policy is provided as a satisfaction guarantee.

title: 63cm Long Zipper Beige+Pink Wavy Cosplay Hair Wig Rw157
salesRank: {'Beauty': 2236}
categories: Beauty > Hair Care > Styling Products > Hair Extensions & Wigs > Wigs
price: 11.83
brand: Generic
```

A.4 Ranking Prompt Example

Below is an example of a single pass in a list-wise ranking pipeline with a window size of 4 and a stride of 2 ($w = 4$ and $d = 2$) assuming there are 3 history items ($l = 3$).

System: You are an intelligent assistant that can rank items based on the user's preference.

User: User 1656 has purchased the following items in this order:

```
{
    "Item ID": 1069,
    "title": "SHANY Professional 13-Piece Cosmetic Brush Set with Pouch,
    Set of 12 Brushes and 1 Pouch, Red",
    "salesRank_Beauty": 248,
    "categories": [
        ["Beauty", "Tools & Accessories", "Makeup Brushes & Tools", "Brushes & Applicators"]
    ],
    "price": 12.95,
    "brand": "SHANY Cosmetics"
},
{
    "Item ID": 2424,
    "title": "SHANY Eyeshadow Palette, Bold and Bright Collection, Vivid, 120 Color",
    "salesRank_Beauty": 1612,
    "categories": [
        ["Beauty", "Makeup", "Eyes", "Eye Shadow"]
    ],
    "price": 16.99,
    "brand": "SHANY Cosmetics"
},
{
    "Item ID": 2856,
    "title": "SHANY Studio Quality Natural Cosmetic Brush Set with Leather Pouch, 24 Count",
    "salesRank_Beauty": 937,
    "categories": [
        ["Beauty", "Tools & Accessories", "Bags & Cases", "Cosmetic Bags"]
    ],
    "price": 26.99,
    "brand": "SHANY Cosmetics"
}
```

I will provide you with 4 items, each indicated by number identifier []. Analyze the user's purchase history to identify preferences and purchase patterns. Then, rank the candidate items based on their alignment with the user's preferences and other contextual factors.

Assistant: Okay, please provide the items.

User: [1]

```
{
    "title": "SHANY Cosmetics Intense Eyes Palette 72 Color Eyeshadow Palette, 17 Ounce",
    "salesRank_Beauty": 181358,
    "categories": [
        ["Beauty", "Makeup", "Makeup Sets"]
    ],
    "price": 26.4,
    "brand": "SHANY Cosmetics",
    "Number of users who bought both this item and Item ID 1069": 18,
    "Number of users who bought both this item and Item ID 2424": 0,
    "Number of users who bought both this item and Item ID 2856": 16
}
```

Assistant: Received item [1].

User: [2]

```
{
    "title": "SHANY Cosmetics Carry All Train Case with Makeup and Reusable Aluminum Case, Cameo",
    "salesRank_Beauty": 2439,
    "categories": [
        ["Beauty", "Makeup", "Makeup Sets"]
    ],
    "price": 39.99,
    "brand": "SHANY Cosmetics",
    "Number of users who bought both this item and Item ID 1069": 27,
    "Number of users who bought both this item and Item ID 2424": 1,
    "Number of users who bought both this item and Item ID 2856": 29
}
```


Assistant: Received item [2].

User: [3]


```
{
    "title": "SHANY COSMETICS The Masterpiece 7 Layers All-in-One Makeup Set",
    "salesRank_Beauty": 2699,
    "categories": [
        ["Beauty", "Makeup", "Makeup Sets"]
    ],
    "price": 41.89,
    "brand": "SHANY Cosmetics",
    "Number of users who bought both this item and Item ID 1069": 23,
    "Number of users who bought both this item and Item ID 2424": 2,
    "Number of users who bought both this item and Item ID 2856": 25
}
```


Assistant: Received item [3].

User: [4]


```
{
    "title": "SHANY Silver Aluminum Makeup Case, 4 Pounds",
    "salesRank_Beauty": 16605,
    "categories": [
        ["Beauty", "Tools & Accessories", "Bags & Cases", "Train Cases"]
    ],
    "price": 59.95,
    "brand": "SHANY Cosmetics",
    "Number of users who bought both this item and Item ID 1069": 32,
    "Number of users who bought both this item and Item ID 2424": 1,
    "Number of users who bought both this item and Item ID 2856": 40
}
```


Assistant: Received item [4].

User: Analyze the user's purchase history to identify user preferences and purchase patterns. Then, rank the 4 items above based on their alignment with the user's preferences and other contextual factors. All the items should be included and listed using identifiers, in descending order of the user's preference. The most preferred recommendation item should be listed first. The output format should be [] > [], where each [] is an identifier, e.g., [1] > [2]. Only respond with the ranking results, do not say any word or explain. Output in the following JSON format:

```
{
    "rank": "[] > [] .. > []"
}
```