Let Me Do It For You: Towards LLM Empowered Recommendation via Tool Learning

Yuyue Zhao
yyzha0@mail.ustc.edu.cn
University of Science and Technology of China
University of Amsterdam
Hefei, Anhui, China

Jiancan Wu*
wujcan@gmail.com
University of Science and Technology of China
Hefei, Anhui, China

Xiang Wang*†
xiangwang1223@gmail.com
University of Science and Technology of China
Hefei, Anhui, China

Wei Tang
weitang@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Dingxian Wang
dingxian.wang@student.uts.edu.au
University of Technology Sydney
Ultimo, New South Wales, Australia

Maarten de Rijke
m.derijke@uva.nl
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

Conventional recommender systems (RSs) face challenges in precisely capturing users' fine-grained preferences. Large language models (LLMs) have shown capabilities in commonsense reasoning and leveraging external tools that may help address these challenges. However, existing LLM-based RSs suffer from hallucinations, misalignment between the semantic space of items and the behavior space of users, or overly simplistic control strategies (e.g., whether to rank or directly present existing results). To bridge these gap, we introduce ToolRec, a framework for LLM-empowered recommendations via tool learning that uses LLMs as surrogate users, thereby guiding the recommendation process and invoking external tools to generate a recommendation list that aligns closely with users' nuanced preferences.

We formulate the recommendation process as a process aimed at exploring user interests in attribute granularity. The process factors in the nuances of the context and user preferences. The LLM then invokes external tools based on a user's attribute instructions and probes different segments of the item pool. We consider two types of attribute-oriented tools: rank tools and retrieval tools. Through the integration of LLMs, ToolRec enables conventional recommender systems to become external tools with a natural language interface. Extensive experiments verify the effectiveness of ToolRec, particularly in scenarios that are rich in semantic content.

* Jiancan Wu and Xiang Wang are corresponding authors.
† Xiang Wang is also affiliated with Institute of Dataspace, Hefei Comprehensive National Science Center.

SIGIR '24, July 14–18, 2024, Washington, DC, USA

ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657828

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender system, Large language models, Tool learning

ACM Reference Format:
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 (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657828

1 INTRODUCTION

Recommender systems (RSs) are typically designed to identify user preferences and subsequently suggest potential items of interest [9, 15, 22, 34, 43, 47, 48, 54]. This strategy has two important limitations. First, the capabilities of existing RSs to accurately capture a user's true preferences are limited when relying solely on historical interaction data. Second, conventional RSs are often "narrow experts," lacking commonsense knowledge about users and items, which leads to a restricted scope of recommendations [14].

Inspired by the commonsense reasoning and knowledge utilization capabilities of large language models (LLMs), there have been several attempts to integrate LLMs with RSs and mitigate their inherent limitations [26, 49, 62]:

- **LLMs as RSs**: Here, LLMs, whether initially trained or further fine-tuned using user-item interaction data, are adapted to serve as RSs [27, 41, 50] and directly generate candidate items in text. This approach easily suffers from the hallucination problem [2], especially given large item catalogue sizes and extensive item names [21].
- **LLMs enhance RSs:** Here, RSs are enhanced with world knowledge and reasoning abilities of LLMs [25, 27, 32, 41, 50, 59]. This category limits LLMs to offering semantic information within a conventional recommendation paradigm, sometimes leading to inconsistencies between the semantic and behavior spaces.

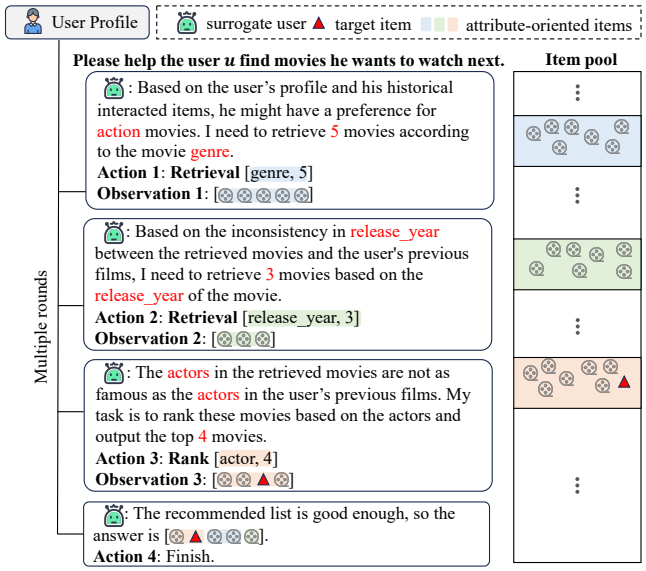


Figure 1: Illustrating how ToolRec works. The LLM-based *surrogate user* learns the real user's preferences and decides to employ attribute-oriented tools to explore areas of items. This process leads to a broad view of items, which, in turn, leads to the successful retrieval of the target item. It is important to note that the areas representing different attribute-oriented items that are retrieved according to a specific attribute may contain overlapping elements.

- **LLMs control RSs:** Here, LLMs are used to monitor and control the recommendation pipeline. Existing controllers either have simple control strategies [10, 16] or necessitate active user involvement [8]. Their decisions are rarely human-like, which hinders their effectiveness in applications.

Tool learning for recommendations. Motivated by recent advancements in tool learning with foundation models [e.g., 35, 36, 55], we propose to use LLMs as a surrogate user to emulate her/his decision-making process along with the utilization of tools. At the core of our proposal is the task of learning to adaptively select appropriate recommender tools and curate a user-centric item list that is aligned with the user's preferences. Figure 1 illustrates a four-round example of how ToolRec works. The LLM starts the simulation by focusing on the movie genre and selects an initial set of 5 movies; satisfied with the genre of the returned movies, it then aims to complement the set based on the release_year, and retrieves 3 additional movies; subsequently, the LLM refines its focus towards the actors, leading to an adjustment of four movies in the list. Such iterative refinements continue until the simulator deems the movie list satisfactory enough to include the item of interest, thus finishing the recommendation process. Notice how the LLM directs the recommendation through multiple decision-making rounds with attribute signals, contrasting with conventional RSs. By adopting this approach, we aim to move beyond relying solely on users' historical interactions, resulting in a more tailored set of recommended items (cf. the red area on the right of Figure 1).

Using LLM tool learning to simulate users' decision-making processes presents distinct challenges. The first challenge concerns the recommendation ability of LLMs. Although LLMs are pretrained on extensive datasets [1, 28], improving their ability to produce quality recommendations remains a challenge, particularly in domain-specific scenarios. The second challenge is developing appropriate attribute-oriented tools. Attribute-oriented tools should not only be capable of exploring facets of the item pool (e.g., genre, release year of movies) but also need to be effective in handling different attribute choices during decision simulation. Lastly, leveraging LLMs to refine the set of candidate items in each round presents a significant challenge. This step could ensure that the final results benefit from an LLM's open-world knowledge and are not limited by a single "narrow expert."

A new proposal for tool learning for recommendations. To address the challenges listed above, we introduce ToolRec, for LLM-empowered recommendations via tool learning, which is aimed at aligning the emergent abilities of LLMs with the demands of recommender systems. ToolRec comprises three key components: (i) A **user decision simulation module**: We use LLMs initialized with user behavior history, acting as a *surrogate user* to evaluate user preferences against the current scenario. (ii) **Attribute-oriented tools**: We develop two distinct sets of attribute-based tools: rank tools and retrieval tools. The ranking tools are operationalized by LLMs with attribute-oriented ranking instructions, while the retrieval tools are operationalized by merging a frozen backbone with additional fine-tuned attribute encoders. These tools are activated when the *surrogate user* identifies unsatisfactory attributes, and then fetches corresponding candidate items. (iii) A **memory strategy**: This component checks the presence of intermediate results and stores them with associated tool marks, aiding the LLM in leveraging open-world knowledge to refine the final recommendation list. The latter two components are governed by the *surrogate user*'s decisions, and conclude the recommendation process once a satisfactory candidate item list has been found. ToolRec's iterative framework integrates LLMs into recommender systems while enhancing the quality of recommendations.

Contributions. Our main contributions are as follows:

- We propose ToolRec, a framework that deploys LLMs to enhance recommendations via tool learning. It employs LLMs to closely emulate user preferences, thereby improving the accuracy of recommendations generated during user decision simulation.
- To better meet the surrogate user's needs, we incorporate attribute-oriented tools and a memory strategy. Those components address the challenge of effective item retrieval based on identified attributes, ensuring the recommendations are well-aligned with user preferences.
- Experimental results on three real-world datasets demonstrate the effectiveness of ToolRec, especially in domains enriched by world knowledge.

2 RELATED WORK

We review tool learning with LLMs and the use of LLMs for recommendation.

2.1 LLMs with Tool Learning

There is a growing trend to employ LLMs to construct autonomous agents to achieve decision-making capabilities [4, 31, 35, 36, 61].

These LLM-based agents often fall short in domains that demand extensive expert knowledge and suffer from hallucination issues [42]. To alleviate these problems, these agents are enhanced with the ability to invoke external tools for action execution. Previous external tools can be grouped into three types: APIs [24, 30, 31, 35, 36], Databases & Knowledge Bases [11, 17, 23], and External Models [38, 52, 60].

The use of APIs as tools has become a popular approach. E.g., HuggingGPT [36] employs models on HuggingFace to accomplish complex user tasks. API-Bank [24] serves as an LLM-based API recommendation agent, and autonomously searches and generates suitable API calls across various programming languages and domains. ToolBench [30] is an LLM-based tool generation system that creates various tools based on natural language requirements. Connecting LLMs to external databases or knowledge bases enables access to domain-specific information, thus generating more realistic actions. E.g., ChatDB [17] uses SQL statements to query databases, enabling logical action execution by agents. Similarly, in the recommendation scenario, RecMind [44] and InteRecAgent [21] engage various expert systems such as MySQL and planners to retrieve detailed item information. Employing external models can expand the range of feasible actions. E.g., MemoryBank [60] employs two language models to enhance text retrieval capabilities: one for encoding input text and the other for matching query statements. MM-REACT [52] integrates various vision models to improve its performance in visual understanding tasks.

Our attribute-oriented rank tools can be summarized into APIs, while retrieval tools can be summarized into external models. This approach helps our surrogate user to explore different interest areas and return a better user preference-aligned item set.

2.2 LLMs for Recommendation

Methods using LLMs in RSs come in three groups, based on the role of the LLMs: LLMs as RSs [3, 12, 13, 19, 51, 53, 58], LLMs as assistants [27, 41, 50], and LLMs as pipeline controllers [8, 10, 21, 44].

LLMs as RSs involve using LLMs to generate candidate items. For instance, P5 [12] is fine-tuned on T5 [33] with a collection of personalized prompts to achieve zero-shot generalization. Following this, UP5 [19] and VIP5 [13] extend the paradigm to fairness and multimodal tasks, respectively. InstructRec [58] views recommendation as an instruction following task for LLMs, tuning T5 with user-specific instructions. TALLRec [3] trains LLaMA [39] with LoRA [18] to follow instructions and respond to a binary query provided within the contextual information. LLaRA [25] utilizes hybrid prompting to bridge the modality gap between traditional recommendation systems and LLMs.

When LLMs serve as Assistants, LLMs are leveraged for their factual knowledge or reasoning capabilities to generate or encode auxiliary textual features, assisting conventional RSs with semantic information. For example, Various studies use auxiliary textual features from BERT [7] for document ranking [63], while others aim to leverage these features for news recommendation [32, 46, 59]. KAR [50] extracts reasoning and factual knowledge from LLMs to serve as augmented features, enhancing recommendation models in a model-agnostic manner.

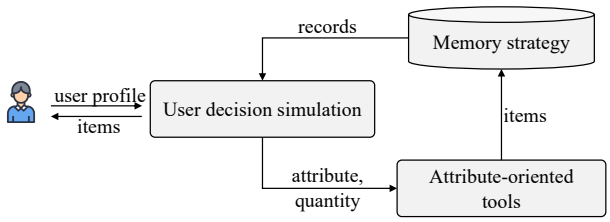


Figure 2: An overview of the proposed LLM-based recommendation method via tool learning.

In the setting where LLMs act as Pipeline Controllers, models such as Chat-REC [10] use ChatGPT to understand user preferences, decide whether to use the backend recommender system, and refine the suggested items before showing them to the user. RecLLM [8] and InteRecAgent [21] propose frameworks for integrated conversational recommender systems with LLMs managing dialogues, understanding user preferences. InteRecAgent further determines the usage of various tools (i.e., information query tools, retrieval tools, and rank tools) to support the candidate items. RecMind [44], inspired by P5's various-task experiment setup, manages tool usage through a Self-Inspiring mechanism. Agent4Rec [56] simulates and infers user-personalized preferences and behavior patterns using a LLM-based movie recommendation simulator.

While these pipeline controller efforts address challenges similar to ours, there are several aspects that make our approach different from theirs. First, we focus on the sequential recommendation task and top-N recommendation setting. Second, we introduce attribute-oriented tools designed specifically for the recommendation exploration journey. Lastly, we propose a proactive analysis of preference mismatches by using an LLM as a surrogate user.

3 METHODOLOGY

We propose ToolRec, as illustrated in Figure 2. We first formalize the recommendation process as a process exploring users' interests. Then, we introduce a general framework that adapts LLMs as surrogate users to enhance the recommendation mechanism through attribute-oriented tools.

3.1 Problem Formulation

In the context of sequential recommendation, as illustrated in Figure 1, given user u with a historical interaction sequence $\mathcal{H} = \{i^1, i^2, \ldots, i^{n-1}\}$, the goal is to predict the next item of interest $i^n \in \mathcal{I}$, where $\mathcal{I}$ is the complete item pool. Conventional RSs retrieve a subset $\mathcal{I}_c \subseteq \mathcal{I}$ based on the predicted preferences of user u. However, this easily results in the desired item i^n being absent from $\mathcal{I}_c$.

In this work, we position LLMs as the central controller for recommendation, simulating the user exploration w.r.t. item attributes in a multi-round manner. The interaction history $\mathcal{H}$ of user u is fed into the LLM, so as to initialize the profile of surrogate user $\hat{u}$. In the first round, $\hat{u}$ identifies a key attribute a_1 and uses external tools to fetch the related item set $\mathcal{I}_{\hat{u}}^{a_1}$. For clarity and brevity, we omit the subscript $\hat{u}$ when the user is clear from the context. In the second round, $\hat{u}$ contrasts the preferences between $\mathcal{I}^{a_1}$ and $\mathcal{H}$, selects another attribute a_2, and retrieves $\mathcal{I}^{a_2}$. Such iterative refinements continue until $\hat{u}$ is satisfied with the retrieved items,

and outputs the final set $\mathcal{I}_{\hat{u}}$ ranked by preference. Each derived item set reflects the interest emerging from the respective round.

3.2 User Decision Simulation

To validate LLMs' capability in simulating user preferences and utilizing external tools, we draw inspiration from prior tool-learning studies [30, 31, 35, 55] and propose a user decision simulation process. Here is an example of the simulation prompt in the movie recommendation scenario:

User Decision Simulation Prompt Example

"Assuming you are an online movie recommender, your task is to help users find movies they would like to watch next based on their interests. To effectively recommend movies, you should follow three steps: Thought, Action, Observation.
During the Thought step, your objective is to consider how to make the final movie list match the user's preferences, and decide the best course of action $< \mathcal{D} >$. If the movie list is good enough or no better action to take, you will finish early with the candidate movie list. For the Action step, you have three options: Retrieval, Rank, and Finish. $< \mathcal{T} >$"

where $<\mathcal{D}>$ represents the collection of demonstrations that show LLMs what a good movie list looks like, and $<\mathcal{T}>$ denotes the detailed description of the tools. The foundational elements of the user decision simulation methodology are twofold:

Chain-of-thought prompting (CoT). We use CoT [45] to synergize reasoning and action. Here, "reasoning" refers to the "thinking procedure" for how to recommend suitable items to users, and decide the details of the subsequent action. Meanwhile, "action" entails executing the directives derived from reasoning, making "observations," and yielding the corresponding outcome. At each step t, the LLM-driven surrogate user $\hat{u}$ receives an observation o^{t-1} from the last step and derives a thought g_t to take an action $\mathcal{A}_t$ following some policy $\pi(g_t, \mathcal{A}_t|c_t)$, where π could be implemented by any LLMs, specifically ChatGPT, in our ToolRec, and $c_t = (o_0, g_1, \mathcal{A}_1, o_1, \ldots, g_{t-1}, \mathcal{A}_{t-1}, o_{t-1})$ is the *context* to the $\hat{u}$. The primary objective is to deduce the policy and mapping $c_t \rightarrow (g_t, \mathcal{A}_t)$. As shown in the left of Figure 1, within the context c_t, $\hat{u}$ observes a discrepancy in the movie actors between the retrieved movies and the user's previous movies. Consequently, it decides to rank the movie list based on their actors (i.e., $\mathcal{A}_t$) and obtain o_t. If $\hat{u}$ is satisfied within the context (i.e., c_N), by comparing the candidate items to the user's history, then $\hat{u}$ will conclude the process and present the final set of candidate items, denoted as $\mathcal{I}_{\hat{u}}$.
Tool learning. Tool learning technology refers to combine the strength of LLMs and specialized tools, as discussed in previous studies [30]. Here, our tools are activated by the generated action $\mathcal{A}_t$ [31, 35, 55]. For each context c_t, the initial observation $o_0 = (\mathcal{H}, \mathcal{T}, \mathcal{D})$ is composed of the user u's historical interactions $\mathcal{H}$, tool description $\mathcal{T}$, and demonstration $\mathcal{D}$. The tools can be categorized into two types based on $\mathcal{T}_{type}$, which can be either "retrieve tools" or "rank tools." The description $\mathcal{T}$ elucidates the impact of using the tools, and $\mathcal{D}$ offers practical demonstrations of their application. Together, these components promote the effective utilization of external tools. Tool actions are formulated as $\mathcal{T}_{type}[a_t, \$K]$, where $\mathcal{T}_{type} \in \{Retrieval, Rank\}$, a_t indicates the chosen attribute based on $\hat{u}$'s decision at time t, and $\$K$ specifies the number of items to be returned. For instance, as shown in Figure 1, at the first step, $\hat{u}$ opts to use the retrieval tools conditioned on the attribute "genre" and retrieves 5 items. Meanwhile, at step 3, $\hat{u}$ decides to employ rank tools conditioned on the attribute "actor", returning the top 4 items. This phase is constructed to emulate a human-like approach of leveraging tools to broaden their choices through linguistic reasoning.

The CoT prompting phase controls the iterative decision process, determining when to employ external tools or finalize the recommendations. Concurrently, feedback from these tools enhances item exploration and further refines the recommendation process.

3.3 Attribute-oriented Tools

By empowering an LLM to use tools, we can explore different parts of the item pool, uncovering target items that remain latent. To achieve this, we have designed two types of tools.

3.3.1 Rank tools. For attribute-oriented rank tools, we incorporate a ranking instruction template and employ LLMs to order the candidate items. Given that LLMs have demonstrated proficiency in both zero-shot and few-shot scenarios [16], their capabilities are essential for returning item sets that align more closely with the user's latent intent. Our instruction for ranking is framed as: "{*User Historical Record*} {*Prior Retrieved Item Set*}. *Please rank the above recommended movies by measuring the possibilities that the user would like to watch next most according to the movie* [*Attribute Pattern* a_t] *attribute and the given movie history records, and output top* [*Output Size Pattern* $\$K$] *movies except user's historical movies.*"

3.3.2 Retrieval tools. For each user, our attribute-oriented retrieval tools accept an attribute pattern a_t and a specified item set size $\$K$, subsequently returning the matching candidate item set. An intuitive way is creating dedicated models tailored to each attribute pattern. However, it is markedly inefficient to fully train and store separate models for every possible attribute permutation. To address this, we introduce a two-stage method for managing attribute-specific variations:

Pre-training. The pre-training stage is for the original model with no attribute-specific consideration. Without loss of generality, we incorporate the attribute-specific switches into the popular sequential recommendation model, SASRec [22]. For a historical behavior sequence $\mathcal{H}$, the l-th layer behavior representation matrix is denoted as $\mathbf{H}^l = \{\mathbf{h}_1^l, \mathbf{h}_2^l, \ldots, \mathbf{h}_{|\mathcal{H}|}^l\}$, where $\mathbf{h}_i^l$ is the l-th layer's representation of the i-th behavior within the sequence $\mathcal{H}$. Then in the final layer L, the pre-training behavior representation $\mathbf{H}$ is derived:

$$\mathbf{H} = f_{seq}(\mathcal{H}|\beta) = \mathbf{h}_{|\mathcal{H}|}^L, \ \mathbf{H}^{l+1} = \text{Transformer}_h^l(\mathbf{H}^l), \qquad (1)$$

where f_{seq} represents the sequential model, β is the pre-training parameter, Transformer denotes the Transformer architecture encoder, and $\mathbf{H}$ is considered as the user representation at the pre-training phase for sequential recommendation.
Attribute-specific encoder in tuning. As illustrated in Figure 3, after learning the pretrained model, we freeze the pre-training parameter β. Our goal is to fine-tune the attribute-specific encoder. To achieve this, we construct an additional attribute encoder f_{attr}. This encoder takes the user's historical item attribute sequence, a_u,

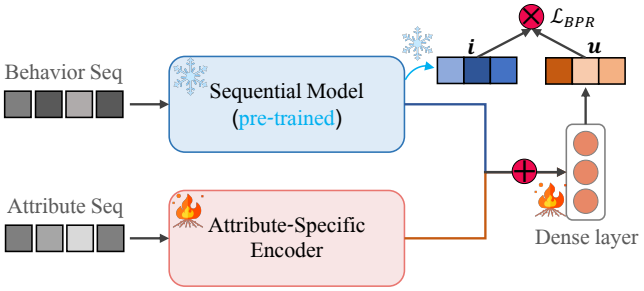


Figure 3: Fine-tuning stage of attribute-oriented retrieval tools. The parameters in the blue 'ice' section remain fixed, while those in the red 'flame' section are exclusively fine-tuned.

as input. Importantly, $\mathcal{H}$ and a_u are aligned based on items in the historical sequence. Subsequently, we define the l-th layer attribute representation matrix as $\boldsymbol{a}_u^l = \{\boldsymbol{a}_1^l, \boldsymbol{a}_2^l, \cdots, \boldsymbol{a}_{|a_u|}^l\}$. Similar to the pre-training approach, the attribute sequence representation $\boldsymbol{a}_u$ is learned as:

$$\boldsymbol{a}_u = f_{attr}(a_u|\gamma) = \boldsymbol{a}_{|a_u|}^L, \boldsymbol{a}_u^{l+1} = \text{Transformer}_a^l(\boldsymbol{a}_u^l). \quad (2)$$

We then incorporate a dense layer to encode the combination of attribute representation $\boldsymbol{a}_u$ and the frozen behavior representation $\mathbf{H}$, resulting in a new user representation:

$$\mathbf{u} = \text{Dense}(\boldsymbol{a}_u \oplus \mathbf{H}, \theta), \quad (3)$$

where γ and θ are the trainable parameters in the tuning phase.

The sequential recommender is trained by minimizing the BPR loss function:

$$\mathcal{L}_{BPR} = -\sum_{(\mathcal{H},v)\in O^+,(\mathcal{H},w)\in O^-,} \log\sigma(\phi(\mathbf{u},v) - \phi(\mathbf{u},w)), \quad (4)$$

where O^+ and O^- denote the positive samples and negative samples, $\phi(\cdot)$ represents the inner-product layer, and $\sigma(\cdot)$ refers to the Sigmoid activation function. During the pre-training phase, $\mathbf{u}$ is represented by $\mathbf{H}$. The finetuned user embedding $\mathbf{u}$ is designed to be sensitive to the specific attribute, while maintaining the personalized sequential recommendation learned in the pre-training phase.

3.4 Memory Strategy

The vast number of items and the complex item names&IDs pose a challenge for LLMs when generating control commands or tool usages. Additionally, items retrieved from various tools should be systematically ordered to aid the LLM-based surrogate user $\hat{u}$ in making decisions. Therefore, we introduce a memory strategy, ensuring the correctness of generated items and cataloging candidate items with their respective tool annotations.

The memory strategy is initialized with the item pool directory. Whenever external tools return candidate items, particularly from attribute-oriented rank tools, the strategy verifies the presence of these items in the initial directory. If there are any discrepancies, the tools are prompted to re-run with additional incorrect details attached behind. Once validated, the candidate items are recorded alongside their associated tool marks, in order to serve the subsequent tool calls. As an illustration, a typical prompt might be

Table 1: The statistics of the datasets used.

Datasets	#Users	#Items	#Interactions	Sparsity (%)
ML-1M	6,041	3,884	1,000,209	95.74
Amazon-Book	158,349	97,566	3,876,663	99.97
Yelp2018	77,278	45,582	2,102,836	99.94

"*Here's the top* [*Output Size Pattern* K] *movie ID, movie name, and the recommendation confidence score from the recommender system with* [*Attribute Pattern* $a_{\hat{u}}$] *type.* {*Candidate Item Set*}."

4 EXPERIMENTS

In this section, we report on extensive experiments aimed at evaluating the performance of our proposed ToolRec. Our experiments focus on answering the following research questions: **(RQ1)** How does ToolRec compare to conventional RSs and LLM-based RSs in the sequential recommendation setting? **(RQ2)** How do different components (*i.e., user decision simulation, termination round, attribute-oriented retrieval tools*) influence our ToolRec? And **(RQ3)** Are LLMs capable of using their inherent knowledge to cater to the recommendation task's needs?

4.1 Experimental Settings

4.1.1 Datasets. To evaluate the effectiveness of our methods, we conduct experiments on three real-world datasets: Ml-1M, Amazon-Book, and Yelp2018.

- **ML-1M**. This dataset is derived from the MovieLens-1M[1] benchmark, which contains user ratings for movies with timestamps. We take the movies' *genre* and *release year* as the attribute information.
- **Amazon-Book**. This dataset[2] is extracted from *Amazon.com* platform. We adopt the Book category to evaluate our method. We take the books' *Price* and *Sales rank* as the attribute information. A 10-core setting is applied to maintain dataset quality.
- **Yelp2018**. This dataset is collected from the 2018 edition of the Yelp Challenge.[3] We utilize local businesses' *Categories*, *City* and *Stars* as attributes. We employ the 10-core setting to ensure a minimum of ten interactions for each user and item.

For each dataset, we organize users' interactions chronologically based on timestamps, allowing us to create the corresponding historical interaction sequences. Items are described using their product IDs&Names. We summarize the statistics of our datasets in Table 1.

4.1.2 Evaluation protocols. We apply the leave-one-out strategy [22, 37] and employ timestamps to set the sequence order, dividing the interaction data into training, validation, and test sets. The attribute-oriented retrieval tools are trained on the training and validation sets. To measure the recommendation performance, we adopt two widely used metrics *NDCG@N* and *Recall@N* to evaluate the results within the top-N positions, with $N = 10$ in our experiments. Due to budget constraints, following prior work [10, 16], we randomly sample 200 users and their historical behaviors from the test set for each dataset. A similar, and similarly-sized, setting has been adopted in other recent LLM-related recommendation researches [56, 57]. To enhance the robustness and credibility of our results, we repeated

[1] https://grouplens.org/datasets/movielens/ [2] https://nijianmo.github.io/amazon/
[3] https://www.yelp.com/dataset

the experiments three times, each with a different sample of 200 users. The average results and standard deviations from these trials are presented in Table 2.

4.1.3 Baselines. We compare ToolRec[4] against two traditional approaches (the first two below), one that uses LLMs as RSs (the following one), two that enhance RSs with LLMs (the next two), and two that use LLMs to control RSs (the remaining two).

- **SASRec** [22]. A self-attention-based sequential recommender, which employs the encoder of the Transformer architecture to generate representations of users' behavior sequences.
- **BERT4Rec** [37]. A bidirectional self-attention-based sequential recommender. It uses the Transformer encoder to predict randomly masked items in a sequence by conditioning on both their left and right context, thereby capturing user historical behaviors.
- **P5** [12]. An encoder–decoder Transformer-based approach that unifies different recommendation related tasks into a single generative LLM. For our sequential recommendation downstream task, we adopt the personalized prompts from OpenP5 [51] and apply the same random indexing method to items as used in our ToolRec. This model is fully fine-tuned using these personalized prompts on the pre-trained T5-small [33].
- **SASRec**$_{BERT}$ [7]. An attention-based method that modifies the single interaction sequence encoder by adding a feature encoder (structured as shown in Figure 3). This model is fully fine-tuned using semantic representations pre-trained with BERT.
- **BERT4Rec**$_{BERT}$. A variation of the BERT4Rec sequential recommender enhanced with BERT's pre-trained representations.
- **Chat-REC** [10]. The first work was on using an LLM as a controller. For our sequential recommendation task, we adopt the recommendation prompt based on the original paper, choose SASRec to supply the candidate items, and modify the output format for parsing.
- **LLMRank** [16]. A LLM-based ranking model. In our full-ranking experimental setup, we retrieve thirty candidate items (according to the original paper) from SASRec and adjust the output format for parsing.
- **ToolRec**. The method that we propose. We primarily evaluate two versions of ToolRec: ToolRec that is implemented using external attribute-oriented retrieval tools with frozen *SASRec* parameters; and ToolRec$_B$ that is developed using BERT4Rec as the backbone for external attribute-oriented retrieval tools.

4.1.4 Implementation details. We use the gpt-3.5-turbo-16k (ChatGPT for short) model as our primary LLM within ToolRec. This model is responsible for parsing user preferences and assisting in tool learning. We retain the default hyperparameters of ChatGPT without modifications. To enable ToolRec to emulate user decisions, we incorporate decision demonstrations into the prompt for in-context-learning. For attribute-oriented tools, retrieval tools are enhanced with corresponding additional item attributes specific to each dataset. Attributes are represented using word embeddings from GloVe [29]. These tools are developed on two widely recognized backbones: SASRec and BERT4Rec. Additionally, our rank tools are constructed with instructions on ChatGPT. We limit our decision processes to a maximum of eight rounds.

[4] Our codes are available at https://github.com/Go0day/ToolRec-Code.

4.2 Performance Comparison (RQ1)

We compare our proposal with conventional sequential recommenders and LLM-enhanced sequential recommenders, as detailed in Section 4.1.3. The results are reported in Table 2, from which we observe:

- In general, ToolRec outperforms all baselines on the ML-1M and Amazon-Book datasets. The performance improvement can be attributed to the efficacy of our designed ToolRec framework. By delving into various facets of the item pool and utilizing LLMs to guide the exploration process, it can more effectively align with the user's intent.
- ToolRec and ToolRec$_B$ consistently demonstrate improved performance compared to their respective underlying models (i.e., SASRec and BERT4Rec). This confirms the superiority of the ToolRec framework and underscores its adaptability. The results suggest that our approach, which harnesses LLMs for recommendation through tool learning, has the potential to be integrated as a supplementary module in various recommendation systems.
- ToolRec exhibits subpar performance on the Yelp2018 dataset. We attribute this to the LLM's limited knowledge of local businesses. Since LLMs are primarily trained on widely available web data, they might possess a more robust understanding of topics like movies and books than local (niche) businesses. In more specialized domains, ToolRec, which involves multiple interactions between the LLM and external tools, has the potential to make more incorrect decisions and may exhibit heightened deficiencies when compared to other LLM-based approaches such as LLMRank and SASRec$_{\text{BERT}}$.
- In the approach where LLMs serve as RSs, P5 demonstrates strong performance on the ML-1M. This is exciting, considering we use datasets on the scale of millions (Table 1), and accurately generating items for users is challenging. However, P5 shows weaker performance on the Amazon-Book and Yelp2018 datasets, potentially due to the fact that the sparsity of those datasets is higher than ML-1M, thus leading to more severe hallucination issues. The development of more carefully designed item indexing methods [20] may help alleviate this problem, but this falls beyond the scope of our current research.
- For LLM-enhanced RS approaches, such as SASRec$_{BERT}$ and BERT4Rec$_{BERT}$, it is not surprising to see improvements over their backbone models in most cases. However, on the ML-1M dataset, SASRec$_{BERT}$ performs much worse than SASRec. Meanwhile, on the Amazon-Book dataset, BERT4Rec$_{BERT}$'s performance is comparable to its baseline version. While language models can generally augment recommendation tasks, addressing the disparity between semantic and behavioral spaces remains challenging.
- For LLM-controlled RS approaches, such as Chat-REC and LLMRank, they couldn't achieve consistent improvements over SASRec in the same way as ToolRec. This suggests that merely using basic control strategies or employing LLM as a ranker might not be the most effective way forward for recommendation tasks. A more comprehensive strategy, as demonstrated by ToolRec, appears better suited to harnessing LLMs to enhance recommendations.

Table 2: The test performance comparison on three real-world datasets. The bold font denotes the winner in that column. The row "Improvement" indicates the relative performance gain of our ToolRec and the suboptimal method.

	ML-1M		Amazon-Book		Yelp2018	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
SASRec	0.203±0.047	0.1017±0.016	0.047±0.015	0.0205±0.006	0.030±0.005	0.0165±0.006
BERT4Rec	0.158±0.024	0.0729±0.008	0.042±0.015	0.0212±0.009	0.033±0.021	**0.0218**±0.016
P5	0.208±0.021	0.0962±0.009	0.006±0.003	0.0026±0.002	0.012±0.005	0.005±0.001
SASRec_{BERT}	0.192±0.015	0.0967±0.006	0.042±0.003	0.0194±0.002	0.032±0.016	0.0131±0.007
BERT4Rec_{BERT}	0.202±0.013	0.0961±0.009	0.045±0.023	0.0233±0.012	**0.040**±0.028	0.0208±0.015
Chat-REC	0.185±0.044	0.1012±0.016	0.033±0.015	0.0171±0.007	0.022±0.003	0.0121±0.001
LLMRank	0.183±0.049	0.0991±0.020	0.047±0.013	0.0246±0.004	0.030±0.005	0.0140±0.004
ToolRec	**0.215**±0.044	**0.1171**±0.018	**0.053**±0.013	**0.0259**±0.005	0.028±0.003	0.0159±0.001
ToolRec_B	0.185±0.018	0.0895±0.002	0.043±0.013	0.0223±0.008	0.025±0.005	0.0136±0.009
Improvement	3.36%	15.10%	14.28%	5.14%	−29.16%	−27.32%

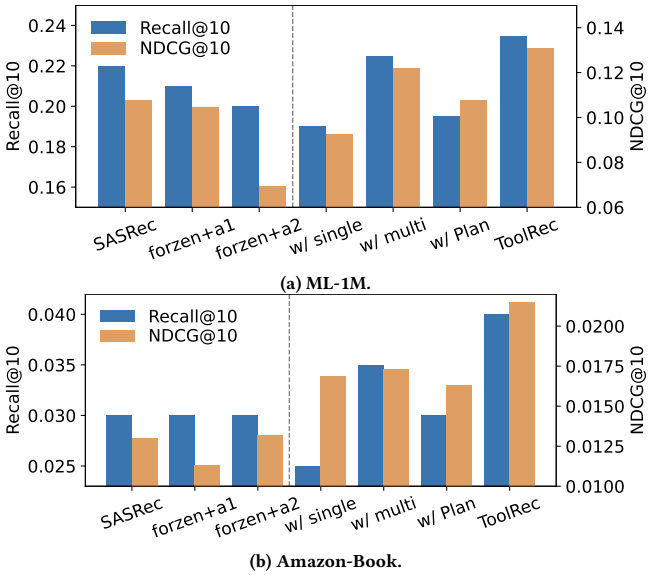
4.3 Decomposing ToolRec (RQ2)

Next, we delve deeper into ToolRec. We examine the user decision simulation, revealing how multi-round interactions enhance recommendation quality. We evaluate the efficiency of our attribute-oriented retrieval tools, highlighting the balance between rich information and computational practicality.

4.3.1 Effectiveness of user decision simulation. To verify the contribution of the user decision simulation component (cf. Section 3.2), we conducted an ablation study considering three variants of ToolRec: (i) Disabling the CoT and tool learning components of ToolRec, we forced the LLM to rank the candidate items from SASRec and output the result. This variant is denoted as "w/ single"; (ii) Unlike "w/ single", we had the LLM rank candidate items using both SASRec and the attribute-oriented retrieval tools, termed "w/ multi"; (iii) We disable the CoT component, and instead, the LLM was instructed to generate all the steps of tool-calling at once and then strictly follow the execution plan. This setup is termed "w/ Plan". To ensure fairness in our comparisons, both the "w/ single" and "w/ multi" variants use the same number of candidate items: thirty items in total.

We demonstrate the experimental results in Figure 4 and have the following findings:

- Removing the CoT and tool learning component degrades the model's performance. The "w/ single" variant consistently underperforms both "w/ multi" and "w/ Plan", and its performance is even subpar compared to SASRec. This decline in performance can be attributed to "w/ single" relying exclusively on the LLM's zero-shot ranking without leveraging additional information to refine the results.
- It is important to note that the performance of attribute-oriented retrieval tools, namely "frozen + $a1$" and "frozen + $a2$", is inferior to SASRec in dataset ML-1M. However, results refined by 'w/ multi' not only surpass 'w/ single' but also outperform SASRec. This improvement suggests that the strength of our approach is not solely due to the broader item size, but also derives from the assistance of the additional attribute information.
- When comparing "w/ Plan" with "w/ multi", the latter consistently achieves superior results. One potential reason might be that the "generate-then-execute" approach in "w/ Plan" lacks explicit performance guidance, causing it to miss the target. This further underscores the importance of the user decision simulation in our approach.



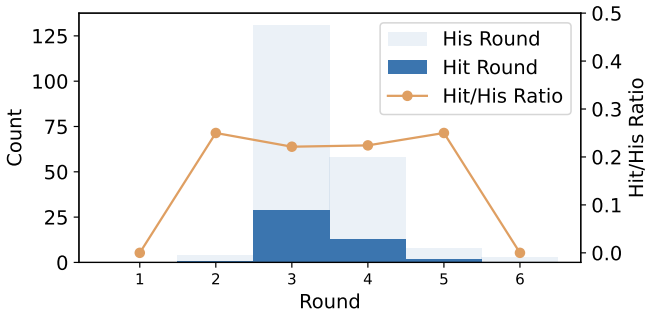
(a) ML-1M.

(b) Amazon-Book.

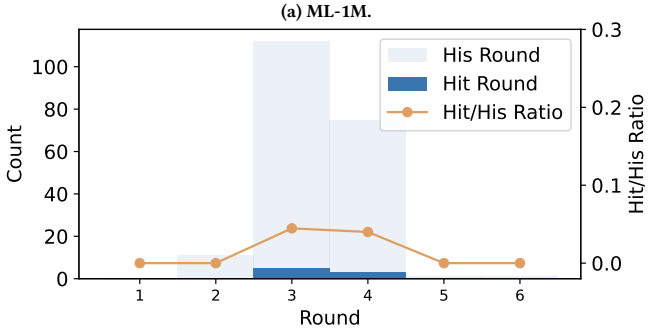
Figure 4: Performance of ToolRec and its variants. The right side of the dividing line indicating the methods involving LLMs.

4.3.2 Analysis of round termination in ToolRec. Figure 5 illustrates the distribution of the number of termination rounds, N, in ToolRec.

Based on the data, we make the following observations: (i) The majority of processes conclude within three or four rounds. This suggests that after a few iterations, our LLM-based surrogate user, denoted as $\hat{u}$, develops a good understanding of whether the user's preferences have been adequately addressed. (ii) While the majority of successful hits also occur within three or four rounds, an interesting trend emerges in the ML-1M dataset: both shorter and longer processes tend to be more successful in reaching the target item. One interpretation is that shorter rounds signify tasks that are more straightforward for the surrogate user $\hat{u}$. But for users with

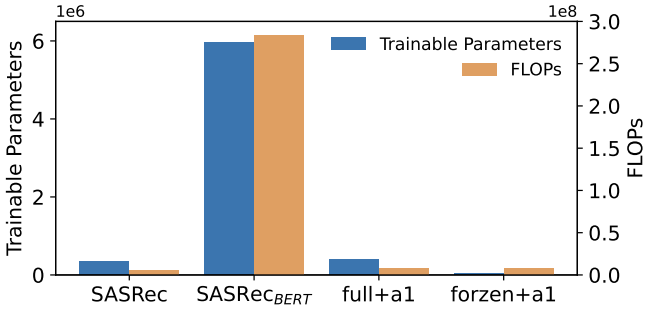


(a) ML-1M.

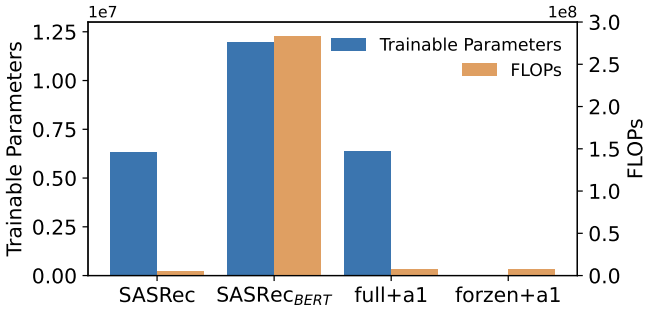


(b) Amazon-Book.

Figure 5: Distribution of termination rounds for ToolRec. "His Round" indicates the distribution of termination rounds for all users, while "Hit Round" highlights the termination round where the recommended list accurately contains the user's target item.



(a) ML-1M.



(b) Amazon-Book.

Figure 6: Comparison of trainable parameters and FLOPs for various retrieval model configurations.

diverse interests or nuanced tastes, $\hat{u}$ might need additional rounds to gather more information and determine if the recommendation process has been satisfactorily completed.

4.3.3 Efficiency and scalability of attribute-oriented retrieval tools. As elaborated in Section 3.3, the attribute-oriented retrieval tools are designed to adeptly follow diverse attribute choices. Figure 6 compares the number of trainable parameters and FLOPs on the ML-1M and Amazon-Book datasets. Here, 'forzen+$a1$' represents our attribute-oriented retrieval tool with frozen backbone parameters, while 'full+$a1$' denotes its full fine-tuning variant.

We have the following observations: (i) The SASRec$_{BERT}$ model is considerably larger than the original SASRec. This increase in size can be attributed to the enriched semantic information present in the BERT embedding. (ii) The trainable parameter count for 'full+$a1$' aligns closely with that of SASRec. Essentially, this is akin to training an entirely new model. Such an approach becomes impractical as the number of attributes escalates. (iii) While both 'forzen+$a1$' and 'full+$a1$' exhibit identical FLOPs, the former boasts a significantly reduced count of trainable parameters compared to the latter. Moreover, since our 'forzen+$a1$' employs the shared backbone of SASRec, storage is primarily reserved for the additive parameters from the attribute-specific encoder and the dense layer.

4.4 Surprises and Limitations (RQ3)

ToolRec benefits from LLMs to reason about users' preferences across attributes, enabling multi-round interaction processes and improved recommendations.

However, language models are not inherently designed for recommendation tasks. Below, we discuss the failures and limitations this gives rise to.

4.4.1 Unanticipated outcomes: Beyond conventional failures. For a rigorous comparison with existing baselines, we employed the Memory strategy (cf. Section 3.4). This strategy ensures that returned items align with the dataset directory, prompting a re-run if discrepancies arise.

However, beyond the traditional recommendation paradigm, some samples that are typically classified as "failures" deserve a second look.

Figure 7 shows an example. At *Thought 5*, when our LLM-based surrogate user, $\hat{u}$, is unsatisfied with the retrieved movies, it typically evaluates the unsatisfactory attributes and uses external tools to uncover additional options, thereby refining the recommendation.

Yet, in this scenario, rather than using external tools or settling for the current list of candidates, $\hat{u}$ decides to suggest "some popular movies", providing a top-N recommendation on its own. Since we never trained the LLM on any of our recommendation datasets, it is evident that not all of the movies it suggests are found in the ML-1M dataset. As a result, this action is labeled as a "failure", prompting a re-run. However, from another perspective, this could be viewed as a successful recommendation; the only limitation is our inability to evaluate it within the current dataset constraints.

4.4.2 Influence of LLM selection on recommendation performance. Following the experimental setup described in Section 4.1, we replace the base LLM with two alternative LLMs: Vicuna1.5-13B-16k [5] (Vicuna in short), an open-source chatbot fine-tuned on

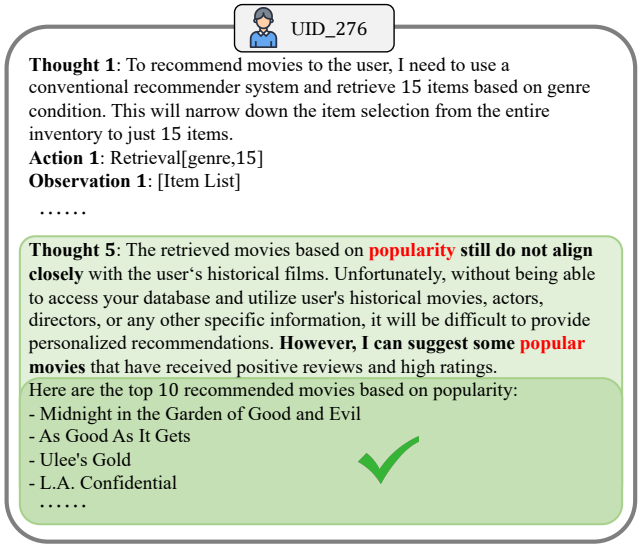
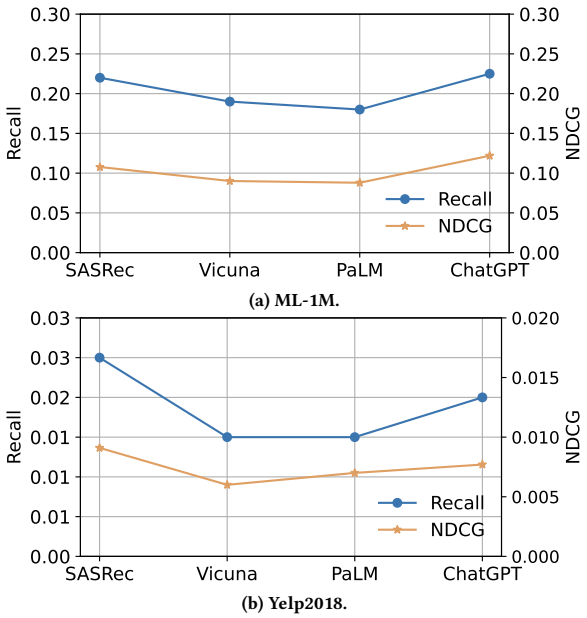


Figure 7: Case study of parsing errors in output samples. Highlighted samples have been manually verified to exist in the real world.



(a) ML-1M.

(b) Yelp2018.

Figure 8: Ranking performance across different LLMs.

Llama2 [40]; and PaLM [6], a commercial LLM developed by Google AI. However, both variants yielded subpar outcomes, either failing to adhere to task instructions, misconstruing tool usage, or misinterpreting user preferences (some representative failures are cataloged in the Appendix). Notably, even though Vicuna and PaLM use the same prompt template as ChatGPT, they are unable to consistently generate recommendations across all test users, despite several attempts. This suggests that ChatGPT has better reasoning capabilities than both Vicuna and PaLM.

Beyond reasoning, a ranking experiment was conducted (cf. Section 4.3.1, w/ multi) to examine the open-world knowledge of various LLMs. As revealed in Figure 8, on datasets like ML-1M (and similar outcomes on Amazon-Book), both Vicuna and PaLM underperformed SASRec, whereas ChatGPT exhibited superior results. These findings align with insights presented in LLMRank [16]. However, all three models lagged behind SASRec on the Yelp2018 dataset, suggesting a possible limitation of LLMs in contexts like local businesses, where open-world knowledge is limited and of more limited use than on the other datasets.

5 CONCLUSION

In this work, we zoomed in on the tool-learning capacities of LLMs, using them as controllers to guide the exploration of item spaces in a recommendation scenario. Specifically, by treating LLMs as surrogate users, they can adeptly capture the nuances of a current context alongside user preferences. Subsequently, we employ attribute-oriented tools for precise item retrieval. We developed two types of attribute-oriented tools: rank tools and retrieval tools, each fetching the corresponding candidate items. To enhance accurate item retrieval, items that appear in the process are verified and stored using the memory strategy. Extensive experiments on real-world datasets rich in knowledge demonstrate the effectiveness and rationality of ToolRec.

The idea of using LLMs for simulation, either under the hood as in our approach or for counterfactual explorations while interacting with users, holds great potential for combining the strengths of LLMs and recommendation models. In the short term, companies operating their own recommender systems may find it impractical to switch to LLM-based RSs. However, ToolRec could enhance recommendation performance by integrating LLMs with their current systems. In the long-term, users are increasingly relying on LLMs for various daily tasks, including recommendations. ToolRec does not require extensive fine-tuning of the LLM, which can lead to additional costs and potential delays due to outdated information. Furthermore, the results from ToolRec are more reliable than those from zero-shot or few-shot LLM recommendations, as they are augmented by traditional recommendation system outputs.

Achieving strong recommendation performance hinges on a dataset with rich semantic knowledge and the robust capabilities of the LLM. In future work, we plan to incorporate recommendation knowledge into LLMs to enhance domain-specific tool learning, such as retrieval-augmented generation or fine-tuning LLMs, potentially reducing the reliance on rich semantic knowledge. Additionally, we intend to explore different types of tools, including search engines and databases, along with a self-reflection strategy, to achieve even more personalized recommendations.

Acknowledgments. This research is supported by the National Science and Technology Major Project (2023ZD0121102), National Natural Science Foundation of China (92270114, 62302321), the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No 101070212.

REFERENCES

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback.

[2] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In *IJCNLP (1)*. 675–718.

[3] 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 *RecSys*. 1007–1014.

[4] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System. In *ICML*. 1052–1061.

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling Language Modeling with Pathways. *J. Mach. Learn. Res.* 24 (2023), 240:1–240:113.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. 4171–4186.

[8] Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, Brian Chu, Zexi Chen, and Manoj Tiwari. 2023. Leveraging Large Language Models in Conversational Recommender Systems. *CoRR* abs/2305.07961 (2023).

[9] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2024. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. *ACM Trans. Inf. Syst.* 42, 1 (2024), 14:1–14:27.

[10] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. *CoRR* abs/2303.14524 (2023).

[11] Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. 2023. OpenAGI: When LLM Meets Domain Experts. In *NeurIPS*.

[12] 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 *RecSys*. 299–315.

[13] Shijie Geng, Juntao Tan, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2023. VIP5: Towards Multimodal Foundation Models for Recommendation. In *EMNLP (Findings)*. 9606–9620.

[14] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on Knowledge Graph-based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* 34, 8 (2020), 3549–3568.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[16] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian J. McAuley, and Wayne Xin Zhao. 2024. Large Language Models are Zero-Shot Rankers for Recommender Systems. In *ECIR (2)*, Vol. 14609. 364–381.

[17] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. ChatDB: Augmenting LLMs with Databases as Their Symbolic Memory. *CoRR* abs/2306.03901 (2023).

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

[19] Wenyue Hua, Yingqiang Ge, Shuyuan Xu, Jianchao Ji, Zelong Li, and Yongfeng Zhang. 2024. UP5: Unbiased Foundation Model for Fairness-aware Recommendation. In *EACL (1)*. 1899–1912.

[20] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to Index Item IDs for Recommendation Foundation Models. In *SIGIR-AP*. 195–204.

[21] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations. *CoRR* abs/2308.16505 (2023).

[22] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[23] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tennenholtz. 2022. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *CoRR* abs/2205.00445 (2022).

[24] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *EMNLP*. 3102–3116.

[25] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Large Language-Recommendation Assistant. In *SIGIR*.

[26] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2023. How Can Recommender Systems Benefit from Large Language Models: A Survey. *CoRR* abs/2306.05817 (2023).

[27] Sheshera Mysore, Andrew McCallum, and Hamed Zamani. 2023. Large Language Model Augmented Narrative Driven Recommendations. *RecSys* (2023), 777–783.

[28] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. ACL, 1532–1543.

[30] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool Learning with Foundation Models.

[31] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *CoRR* abs/2307.16789 (2023).

[32] Zhaopeng Qiu, Xian Wu, Jingyue Gao, and Wei Fan. 2021. U-BERT: Pre-training User Representations for Improved Recommendation. In *AAAI*, Vol. 35. 4320–4327.

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[35] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *NeurIPS*.

[36] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In *NeurIPS*.

[37] 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 *CIKM*. ACM, 1441–1450.

[38] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. In *ICCV*. 11854–11864.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[40] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut

Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[41] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. *CoRR* abs/2304.03153 (2023).

[42] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.* 18, 6 (2024), 186345.

[43] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD*. 950–958.

[44] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. 2023. RecMind: Large Language Model Powered Agent For Recommendation. *CoRR* abs/2308.14296 (2023).

[45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.

[46] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering News Recommendation with Pre-trained Language Models. In *SIGIR*. 1652–1656.

[47] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *SIGIR*. 726–735.

[48] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2024. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. *ACM Trans. Inf. Syst.* 42, 4 (2024).

[49] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A Survey on Large Language Models for Recommendation. *CoRR* abs/2305.19860 (2023).

[50] Yunjia Xi, Weiwen Liu, Jianghao Lin, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, and Yong Yu. 2023. Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models. *CoRR* abs/2306.10933 (2023).

[51] Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2023. OpenP5: Benchmarking Foundation Models for Recommendation. *CoRR* abs/2306.11134 (2023).

[52] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. MM-REACT: Prompting ChatGPT for Multimodal Reasoning and Action. *CoRR* abs/2303.11381 (2023).

[53] Zhengyi Yang, Jiancan Wu, Yanchen Luo, Jizhi Zhang, Yancheng Yuan, An Zhang, Xiang Wang, and Xiangnan He. 2023. Large Language Model Can Interpret Latent Space of Sequential Recommender. *CoRR* abs/2310.20487 (2023).

[54] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. 2023. Generate What You Prefer: Reshaping Sequential Recommendation via Guided Diffusion. In *NeurIPS*.

[55] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*.

[56] An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024. On Generative Agents in Recommendation. In *SIGIR*.

[57] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian J. McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. AgentCF: Collaborative Learning with Autonomous Language Agents for Recommender Systems. *CoRR* abs/2310.09233 (2023).

[58] 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. *CoRR* abs/2305.07001 (2023).

[59] Qi Zhang, Jingjie Li, Qinglin Jia, Chuyuan Wang, Jieming Zhu, Zhaowei Wang, and Xiuqiang He. 2021. UNBERT: User-News Matching BERT for News Recommendation.. In *IJCAI*. 3356–3362.

[60] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing Large Language Models with Long-Term Memory. In *AAAI*. 19724–19731.

[61] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the Minecraft: Generally Capable Agents for Open-World Enviroments via Large Language Models with Text-based Knowledge and Memory. *CoRR* abs/2305.17144 (2023).

[62] 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. *CoRR* abs/2308.07107 (2023).

[63] Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained Language Model based Ranking in Baidu Search. In *KDD*. ACM, 4014–4022.