AGENTICREC: A Recommendation-Oriented Agentic Framework with Progressive Tool-Integrated Reasoning Optimization

Anonymous ACL submission

Abstract

Recommender agents built on Large Language Models offer a promising paradigm for personalized recommendation. However, existing agents typically suffer from a misalignment between their tool-integrated reasoning trajectories and recommendation feedback, limiting their ability to distinguish fine-grained user preferences. To address these challenges, we propose AGENTICREC, an agentic recommendation framework that formulates recommendation as a tool-integrated reasoning process over a recommendation-oriented tool suite. Built upon this framework, we further develop a dedicated two-stage training paradigm tailored for recommender agents. In the first stage, we introduce Recommendation-Oriented Trajectory Activation, optimize the agentic recommendation ability under implicit feedback. In the second stage, Progressive Preference Refinement further refines the agent through bidirectional preference reasoning over self-bootstrapped hard pairs, progressively sharpening preference boundaries. Theoretical analysis and extensive experiments demonstrate the effectiveness of AGENTICREC. Our code is available at https://anonymous.4open.science/r/AgenticRec-FB16.

1 Introduction

Recommender Systems are indispensable for mitigating information overload in the digital era (Gao et al., 2021; Wu et al., 2024). Recently, Large Language Models (LLMs) have significantly affected recommendation (Wu et al., 2024; Peng et al., 2025) due to their strong natural language understanding and reasoning capabilities (DeepSeek-AI, 2025; OpenAI, 2026), enabling semantic matching between complex user intents and diverse item descriptions. However, recommendation is ultimately based on large-scale implicit feedback and collaborative signals, which cannot be fully captured by language priors alone (Wu et al., 2024; Wang et al., 2024a).

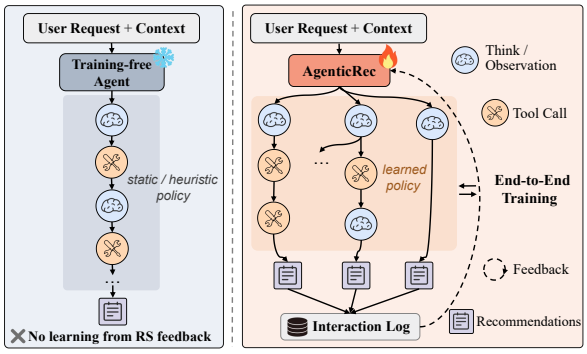


Figure 1: Compared to existing training-free recommender agents like RecMind (Wang et al., 2024b) and InteRecAgent (Huang et al., 2025c) (left), AGENTICREC (right) learns recommendation-driven tool use and reasoning from real-world interaction logs.

Early attempts adapt LLMs to recommendation via fine-tuning, typically formulating the task as next-token prediction through implicit-feedback supervision (Geng et al., 2022; Bao et al., 2023; Liao et al., 2024). Although effective, they remain largely one-shot predictors that output recommendation results from the given prompt. Moreover, real-world systems often involve incomplete and heterogeneous information: ambiguous user intent (Xu et al., 2025b), missing item information (Zhang et al., 2019), and collaborative signals not fully accessible from text alone (Liao et al., 2024).

To overcome these limitations, recommender agents have recently emerged as a promising paradigm, serving as the central hub of recommendation systems (Huang et al., 2025c) or being applied in conversational recommendation scenarios (Fang et al., 2024), leveraging agents' strong planning, reasoning, and memory capabilities to enable more interactive, context-aware, and proactive recommendation (Peng et al., 2025; Huang et al., 2025a).

Along this direction, recent work has explored

recommender agents (Zhang et al., 2025c; Peng et al., 2025) that equip LLMs with structured decision loops (e.g., ReAct (Yao et al., 2023)) and tools to perform *"Think and Act before Recommendation"* for evidence-aware recommendation (Wang et al., 2024b; Huang et al., 2025c; Zhao et al., 2024). Despite their promising progress, existing recommender agents still face two challenges:

- **Misalignment between Tool-Integrated Reasoning Trajectories and Recommendation Feedback:** A common line of existing methods are driven by generic language priors or handcrafted prompt heuristics (Wang et al., 2024b; Huang et al., 2025c) (the left part of Fig. 1), leaving their reasoning trajectories not optimized under recommendation feedback. Since tool use is inherently intertwined with reasoning, not only *when/what* to invoke but also *how* the retrieved evidence is integrated into the final decision must be determined via reasoning, posing a challenge to the design.

- **Insufficiency in Resolving Fine-Grained User Preferences:** Existing methods typically lack a mechanism to progressively sharpen fine-grained preference boundaries on hard, highly confusable candidates. In practice, implicit feedback provides sparse supervision, and inaccurate recommendations may arise from subtle preference differences among top-ranked candidates, which are difficult to correct using coarse list-level signals alone. As a result, agents may learn globally reasonable behaviors, yet remain brittle in challenging cases.

Motivated by these challenges, we propose AGENTICREC, a framework that formulates agentic recommendation and optimizes it under recommendation feedback via a dedicated two-stage training paradigm tailored to it. To support evidence-grounded recommendation decision making, we design a suite of recommendation-oriented tools, including user profile analysis, item information retrieval, behavioral statistics, and collaborative information access, which are specifically tailored to the agent's tool-integrated reasoning recommendation workflow. Built upon this framework, we develop a two-stage optimization paradigm. In the first stage, we introduce Recommendation-Oriented Trajectory Activation, which enables end-to-end optimization of tool-integrated reasoning trajectories under implicit recommendation feedback. Specifically, we optimize the entire decision trajectory through tool-integrated policy optimization. In the second stage, we further propose Progressive Preference Refinement (PPR), a self-bootstrapped refinement strategy that progressively sharpens fine-grained preference boundaries among highly confusable candidates. Specifically, PPR mines hard competitors from the agent's own ranking violations and performs bidirectional preference reasoning to transform challenging recommendation errors into refinement supervision signals.

Our contributions are summarized as follows:

- We propose an agentic recommendation framework that integrates a suite of carefully designed recommendation-oriented tools into the reasoning process to support evidence-grounded recommendation, together with a tool-integrated policy optimization for training recommender agents.

- Within the proposed framework, we develop a two-stage training paradigm consisting of Recommendation-Oriented Trajectory Activation for end-to-end optimization of tool-integrated reasoning trajectories, and Progressive Preference Refinement for progressively sharpening fine-grained preference discrimination among highly confusable items.

- Theoretical analysis and extensive experiments on multiple recommendation benchmarks demonstrate the recommendation effectiveness of AGENTICREC.

2 Related Work

LLMs for Recommendation. LLMs have been explored for recommendation through prompting (Gao et al., 2023; Hou et al., 2024) and task-unified fine-tuning (Geng et al., 2022; Bao et al., 2023; Zhang et al., 2025b; Liao et al., 2024). Recent studies further introduce reasoning to enhance multi-step recommendation decisions (Zhang et al., 2025a; Fang et al., 2025; Lin et al., 2025; You et al., 2025; Liu et al., 2025b).

Recommender Agents. Beyond treating LLMs as black-box predictors, existing studies have introduced the LLM-agent paradigm into recommendation, which can be categorized into three directions (Peng et al., 2025): *simulation-oriented agents* (Zhang et al., 2024a,b; Yoon et al., 2024; Wang et al., 2025) that emulate user behaviors for analysis and data generation, *interaction-oriented agents* (Fang et al., 2024; Nie et al., 2024; Zhu

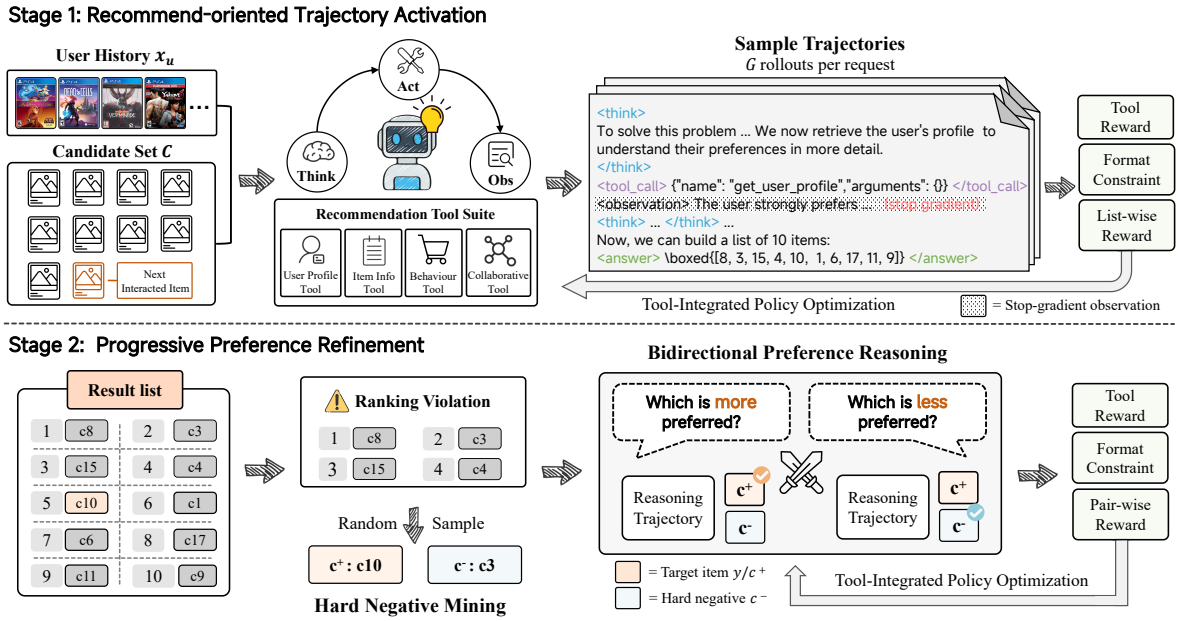


Figure 2: Overview of AGENTICREC.

et al., 2025) that handle users' natural-language requests, and *recommender-oriented agents* (Wang et al., 2024b; Huang et al., 2025c; Wu et al., 2025; Huang et al., 2025b; Cai et al., 2025) that directly leverage the general intelligence and tool-use ability of LLMs to provide recommendations, which is the focus of this paper. Recommender agents can not only improve recommendation quality and interpretability, but also serve as AI assistants in real-world applications, showing great potential to reshape users' online shopping experiences.

3 Problem Formulation

Following prior recommender agent setting (Wang et al., 2024b; Xu et al., 2025a), given a user interaction history $x_u = [v_1, v_2, \ldots, v_t]$ and a candidate set $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, the recommender agent aims to generate a ranked list $r_K = [c_{\sigma(1)}, c_{\sigma(2)}, \ldots, c_{\sigma(K)}]$ of the Top-K items that are most likely to match the user's preferences, where $\sigma(\cdot)$ denotes the ranking order over candidate items and $K \leq n$.

Our goal is to learn an agent policy π_θ that maximizes the expected list-wise utility under real-world feedback y:

$$\max_{\theta} \mathbb{E}_{(x_u, \mathcal{C}, y) \sim \mathcal{D}} \big[R(r_K, y) \big],$$

where $R(\cdot)$ measures the ranking quality induced by implicit feedback (e.g., NDCG@K).

4 AGENTICREC

In this section we introduce AGENTICREC, an agentic recommendation framework which couples a tool-integrated reasoning workflow with a recommendation-oriented tool suite (Sec. 4.1). Built upon this framework, we develop a two-stage training paradigm: (1) Recommendation-oriented Trajectory Activation (Sec. 4.2) end-to-end activates the agentic recommendation ability; (2) Progressive Preference Refinement (Sec. 4.3) further sharpens fine-grained preference discrimination through bidirectional reasoning over self-bootstrapped hard negatives. Fig. 2 provides an overview of AGENTICREC.

4.1 Overall Framework

To support evidence-grounded recommendation decision making, we formulate AGENTICREC as a Tool-Integrated Reasoning agentic recommendation workflow that interleaves deep reasoning with designed recommendation-oriented tool suite. This enables the agent to go beyond the fixed linguistic knowledge of LLMs and autonomously explore across different task scenarios.

Tool-Integrated Reasoning Recommendation Following the ReAct paradigm (Yao et al., 2023), the workflow forms a closed loop that interleaves reasoning and tool invocation for recommendation. The agent iteratively generates *Think* tokens, optionally executes an *Act* by selecting a tool, receives

its *Observation (Obs)*, and finally outputs a *Recommendation (Rec)* list:

$$\textit{Think}_1 \rightarrow \textit{Act}_1 \rightarrow \textit{Obs}_1 \rightarrow \cdots \rightarrow \textit{Rec}.$$

Recommendation-Oriented Tool Suite Drawing on practical experience from recommender-system applications, we define a recommendation-specific tool suite. Notably, our framework allows additional tools to be seamlessly integrated. We briefly introduce the main tools below, with more details provided in Appendix E.

(1) User Profile Tool. It retrieves an LLM-pre-generated profile summarizing user information and long-term preferences, such as favored categories and interests inferred from historical interactions. It can also be extended to update profiles with newly observed feedback.

(2) Item Information Tool. It provides query interfaces for item attributes, such as descriptions and prices, helping the agent understand candidate items on demand during reasoning.

(3) Behavioral Statistics Tool. It extracts features helpful for recommendation based on user behavior data, turning manual feature engineering into an agentic process to better utilize vast behavioral records.

(4) Collaborative Information Tool. It maintains user and item embeddings containing collaborative signals, which support user-to-user and item-to-item retrieval, enabling the agent to reference similar user groups and related items for recommendation. Such collaborative reasoning is difficult to infer from LLM priors alone.

4.2 Recommendation-Oriented Trajectory Activation

As the first training stage, Recommendation-oriented Trajectory Activation (RTA) aims to activate the agentic recommendation ability via tool-integrated policy optimization, where recommendation-specific rewards are propagated through the full reasoning and tool-invocation trajectory through reinforcement learning.

Reward Design We design the reward function by considering three aspects: recommendation quality, output format, and tool-use control. Formally, the reward is defined as:

$$R(r_K, y) = \begin{cases} R_{\text{rec}}(r_K, y) & \text{if valid}(r_K) \text{ and } y \in r_K, \\ -0.5 & \text{if valid}(r_K) \text{ and } y \notin r_K, \\ -1 & \text{otherwise}, \end{cases} \tag{1}$$

where R_{rec} denotes the recommendation reward, and $\text{valid}(r_K)$ indicates that the output satisfies all validity constraints. The detailed designs are described as follows.

(1) Recommendation Reward. We model R_{rec} by computing the NDCG score of the user's ground-truth interacted item in the recommendation list, so as to align the ranking objective with user preference signals. Formally,

$$R_{\text{rec}}(r_K, y) = \text{NDCG@}K(r_K; y). \tag{2}$$

If $y \notin r_K$, the recommendation is considered unsuccessful and receives the penalty -0.5 in Eq. 1.

(2) Format Constraint. Invalid outputs (e.g., invalid items or tools) lead to execution failures in real-world applications. We therefore assign a penalty of -1 when the format is incorrect, enforcing strict validity of generated recommendation.

(3) Tool-call Bonus. To encourage tool use while avoiding reward hacking, we add a small bonus only for high-quality tool-use trajectories. After computing R_{rec}, we add $+0.1$ if the agent achieves Hit@1 and $N_{\text{tool}}(\tau) > 0$. Here, $N_{\text{tool}}(\tau)$ denotes the number of tool invocations in τ.

(4) Tool-call Budget. To control latency and avoid over-reliance on external evidence, we limit each trajectory to at most 10 tool interactions. Exceeding this budget makes the trajectory invalid and receives the invalid-format penalty.

Policy Optimization Algorithm For each training instance $(x_u, \mathcal{C}, y) \sim \mathcal{D}$, we employ GRPO (Shao et al., 2024) as our tool-integrated policy optimization algorithm with rewards in Eq. 1, which estimates advantage using a group G of rollouts.

Dynamic Sampling Since early training often yields uniformly low rewards and thus noisy updates, inspired by the dynamic sampling strategy in DAPO (Yu et al., 2025), we drop training instances whose rollout group contains no informative trajectories, i.e., all sampled trajectories receive negative rewards.

4.3 Progressive Preference Refinement

After the RTA stage, a globally effective policy for recommendation has been activated in the agent. To further strengthen the agent's capability, we introduce Progressive Preference Refinement (PPR) as a self-bootstrapped refinement stage.

We mine hard negatives from items incorrectly ranked above the ground-truth item, and then train the agent with bidirectional preference reasoning on the resulting hard pairs. This bidirectional thinking asks the agent not only to identify what the user is likely to prefer, but also to explicitly reason about what the user is less likely to choose. In this way, PPR moves beyond merely fitting positive implicit feedback, and progressively sharpens fine-grained preference boundaries.

Hard Negative Mining from Ranking Violations Inspired by prior work (Zhang et al., 2013), we exploit the agent's own outputs to automatically mine hard negatives. For each input $(x_u, \mathcal{C})$, we identify the ground-truth positive item $c^+ = y \in \mathcal{C}$ based on user feedback. After the agent generates a ranked list r_K, we check whether c^+ is placed at the top position. If c^+ is not ranked first, we treat this case as a *ranking violation*, since at least one competing item is ranked above the positive, revealing an informative preference boundary.

Concretely, let $\mathrm{rank}_{r_K}(c)$ denote the position of item c in the list r_K (top-K). We construct a hard-negative candidate set by collecting items within the top-K that are ranked above the positive; if the positive item is not in r_K, we use the entire top-K list as the hard-negative pool:

$$H(x_u, \mathcal{C}) = \begin{cases} \{c^- \mid c^- \in r_K,\ \mathrm{rank}_{r_K}(c^-) < \mathrm{rank}_{r_K}(c^+)\}, & \text{if } c^+ \in r_K, \\ r_K, & \text{if } c^+ \notin r_K. \end{cases} \tag{3}$$

Instead of deterministically picking the top-ranked item, we randomly sample one negative item from H to form a hard preference pair (c^+, c^-). This stochastic mining increases negative diversity and avoids overly-adversarial pairs that can yield ambiguous supervision, providing a more stable refinement signal while remaining focused on competitive negatives. The mining procedure requires no external labeling beyond the implicit feedback used to identify c^+.

Bidirectional Preference Reasoning The mined hard pairs expose challenging cases where the current agent fails to distinguish the true preference from strong competitors. The agent not only needs to identify what the user is likely to click, but also to explicitly recognize what the user is less likely to choose under the same context. However, training only on the positive direction provides one-sided supervision: the hard negative is used as a contrastive counterpart, while the reason why it should be rejected remains under-explored. To overcome the limitations of one-sided supervision, we propose *bidirectional preference reasoning*, a training mechanism that turns each hard pair into two complementary tasks, thereby enhancing the agent's versatility in recommendation.

Given a mined hard preference pair (c^+, c^-) under the same user context x_u, we refine the agent via bidirectional preference reasoning from two symmetric directions:

- **Positive Direction:** We first task the agent with the standard objective: identifying the item that the user is more likely to interact with next. This reinforces the agent's ability to recognize matching attributes between the user behavior and the ground-truth item c^+.
- **Negative Direction:** We introduce a counter-perspective task where the agent must explicitly identify the item the user is less likely to choose. This forces the agent to actively deliberate on the specific shortcomings or mismatching features of the hard negative item c^- relative to the user's history, rather than simply treating it as a background non-positive sample.

A prompt example of the PPR task is provided in Appendix A. Both directions share the same structured output format but differ in the queried perspective, thereby providing complementary learning signals for enhancing the agent's capability.

Training Procedure PPR follows the same tool-integrated policy optimization principle as RTA. For each mined pair (c^+, c^-), we construct two preference tasks corresponding to the positive and negative directions, and perform rollouts under the AgenticRec framework. Different from Eq. 1, PPR replaces R_{rec} with a binary pair-wise reward for each direction:

$$R^d_{\text{pair}} = \mathbb{I}[\hat{c}^d = c^d_\star], \quad d \in \{+, -\}, \tag{4}$$

where $c^+_\star = c^+$ for the positive direction, $c^-_\star = c^-$ for the negative direction, and $\hat{c}^d$ is the agent's predicted item under direction d. Other reward components are kept unchanged to ensure valid outputs

and controlled tool invocation during preference refinement.

4.4 Theoretical Justifications

We provide theoretical justifications showing that both training stages of AGENTICREC are well-grounded in improving recommendation performance. Specifically, (1) the RTA gradient estimator with the inclusive group-average baseline yields a directionally unbiased estimate of the expected list-wise utility objective (i.e., NDCG@K), ensuring accurate credit assignment over tool-integrated trajectories; (2) the PPR objective on mined hard pairs minimizes a convex upper bound of the pair-wise ranking error probability, thereby contributing to improved ranking performance. Together, these results indicate that the two stages of AGENTICREC are theoretically sound and jointly contribute to the final result. Formal statements and proofs are deferred to Appendix B.

5 Experiments

5.1 Experimental Setup

Task Setting We design the task setting according to the problem definition in Sec. 3. Specifically, following common practice in prior works (Liao et al., 2024; Chen et al., 2024), each sample contains a candidate set of 20 items, including one ground-truth next item as the positive item and 19 negative items randomly sampled from the item pool. Given the textual input of user behavior records, the model is required to generate a top-10 ranked list from the candidate set.

Datasets and Metrics All training and evaluation samples are constructed from four subsets of the Amazon Reviews 2023[1]: CDs, Instruments, Office, and Games. Recommendation quality is evaluated by NDCG@K (N@K) and HitRate@K (H@K) of the user's actual purchased item in the generated list, where $K \in \{1, 5, 10\}$.

Backbone LLM and Baselines We use Qwen3-4B-Instruct-2507 (Yang et al., 2025) as AGENTICREC's backbone LLM. We compare AGENTICREC with baselines covering conventional sequential recommenders, training-free LLM-based recommenders, and trainable LLM-based recommenders, to evaluate its performance.

Full experimental details are in Appendix C.

[1] https://amazon-reviews-2023.github.io/

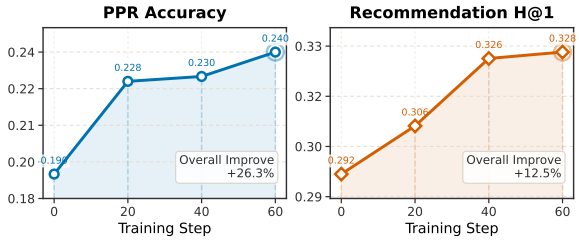


Figure 3: Performance changes during PPR training on the Games dataset, including the agent's preference judgment accuracy on the PPR test set and H@1 on the final recommendation test set.

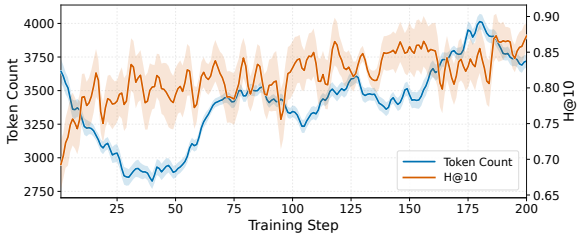


Figure 4: Dynamics of token count and average H@10 of rollout trajectories during first-stage training on the Games dataset.

5.2 Main Results

Table 1 reports the overall comparison on four Amazon subsets. Overall, AGENTICREC achieves the best performance on most datasets and metrics, obtaining the leading results in 19 out of 20 columns. The consistent improvements on both H@K and N@K show that AGENTICREC improves both item matching and ranking quality.

The comparison among different groups shows that the world knowledge and reasoning ability of LLMs can be useful for recommendation, that post-training with recommendation data can further improve the effectiveness of LLM-based recommenders, and that the agent paradigm is also beneficial. By combining these advantages in a tool-integrated agent framework and optimizing it with reinforcement learning, AGENTICREC achieves stronger overall performance, highlighting the potential of trained recommender agents.

5.3 Ablation Study

We conduct an ablation study to examine the effectiveness of the two stage training paradigm and the overall advantage of the agentic framework. As shown in Tab. 2:

- Training consistently improves the performance of both R and TIRR. This verifies that optimizing the model with recommendation-specific signals

Table 1: Evaluation results among baselines and AGENTICREC. The best performance score is denoted in **bold**.

Model	CDs					Instruments					Office					Games				
	H@1	H@5	N@5	H@10	N@10	H@1	H@5	N@5	H@10	N@10	H@1	H@5	N@5	H@10	N@10	H@1	H@5	N@5	H@10	N@10
Conventional Sequential Recommenders																				
Caser	0.0488	0.2526	0.1487	0.5089	0.2307	0.1825	0.4644	0.3275	0.6384	0.3828	0.1771	0.4374	0.3139	0.5541	0.3511	0.1535	0.3953	0.2813	0.5220	0.3217
GRU4Rec	0.0512	0.2348	0.1403	0.4851	0.2202	0.1960	0.4534	0.3291	0.6446	0.3907	0.1642	0.4324	0.3054	0.5705	0.3490	0.1708	0.3781	0.2817	0.5239	0.3288
SASRec	0.0536	0.2348	0.1413	0.4660	0.2145	0.1813	0.4546	0.3238	0.6323	0.3807	0.1662	0.4270	0.3018	0.5863	0.3525	0.1535	0.3915	0.2762	0.5374	0.3234
ReaRec	0.0357	0.1871	0.1082	0.4576	0.1942	0.1507	0.3786	0.2633	0.6188	0.3400	0.1716	0.4265	0.3049	0.5551	0.3457	0.1459	0.3723	0.2298	0.5873	0.2987
Training-free LLM-based Recommenders																				
LLMRank	0.0501	0.2265	0.1373	0.5995	0.2569	0.0417	0.2745	0.1533	0.6017	0.2584	0.0713	0.3102	0.1872	0.5576	0.2665	0.0403	0.2745	0.1519	0.6660	0.2767
InteRecAgent	0.0782	0.2547	0.1660	0.4857	0.2392	0.2063	0.4735	0.3433	0.6515	0.4004	0.1911	0.4467	0.3211	0.6056	0.3717	0.1782	0.4107	0.3009	0.5599	0.3477
Trainable LLM-based Recommenders																				
TALLRec	0.1144	0.3539	0.2313	0.6007	0.3102	0.0649	0.2683	0.1618	0.5147	0.2406	0.1539	0.3592	0.2543	0.5522	0.3318	0.1362	0.3819	0.2608	0.6468	0.3474
LLaRA	0.2324	0.5172	0.3736	0.7234	0.4394	0.2512	0.5404	0.4043	0.6833	0.4510	0.2416	0.5007	0.3928	0.6496	0.4398	0.2763	0.6199	0.4588	0.7332	0.4952
S-DPO	0.1764	0.4183	0.2988	0.6293	0.3669	0.0723	0.2671	0.1657	0.5220	0.2475	0.1573	0.3528	0.2513	0.5421	0.3522	0.2006	0.4529	0.3292	0.6967	0.4079
ReRe	0.2073	0.4648	0.3371	0.6901	0.4080	0.2584	0.5049	0.3900	0.6765	0.4447	**0.2601**	0.5052	0.3970	0.6962	0.4579	0.2921	0.5528	0.4473	0.7102	0.4976
AGENTICREC	**0.2992**	**0.6472**	**0.4795**	**0.8093**	**0.5324**	**0.2586**	**0.6115**	**0.4393**	**0.8052**	**0.5021**	0.2494	**0.5755**	**0.4123**	**0.7773**	**0.4775**	**0.3282**	**0.6468**	**0.4887**	**0.8157**	**0.5445**

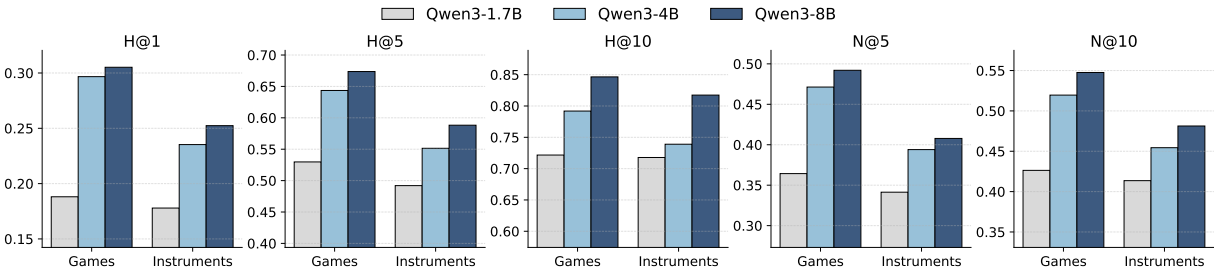


Figure 5: Performance of AGENTICREC with varying backbone sizes.

Table 2: Ablation study of variants of AGENTICREC. R: pure reasoning by LLM, TIRR: tool-integrated reasoning recommendation.

Data	Setting	Variant	H@1	H@5	N@5
CDs	Frozen	R	0.1907	0.4696	0.3303
		TIRR	0.2331	0.5476	0.3955
	Trained	R	0.2324	0.5399	0.3865
		TIRR (RTA)	0.2837	0.6162	0.4606
		TIRR (RTA+PPR)	**0.2992**	**0.6472**	**0.4795**
Instr.	Frozen	R	0.1973	0.5613	0.3818
		TIRR	0.1948	0.5277	0.3673
	Trained	R	0.2328	0.5797	0.4141
		TIRR (RTA)	0.2463	0.6091	0.4321
		TIRR (RTA+PPR)	**0.2586**	**0.6115**	**0.4393**
Office	Frozen	R	0.2147	0.5017	0.3640
		TIRR	0.1963	0.4913	0.3471
	Trained	R	0.2276	0.5374	0.4004
		TIRR (RTA)	0.2326	0.5572	0.4040
		TIRR (RTA+PPR)	**0.2494**	**0.5755**	**0.4123**

can effectively enhance recommendation quality, while the frozen remain suboptimal.

- The two-stage training paradigm is effective. Compared with RTA, RTA+PPR further improves the final recommendation performance, showing that PPR brings real gains on top of RTA. By replaying self-bootstrapped ranking errors and optimizing the agent with bidirectional preference reasoning, PPR refines the agent's preference boundary and transfers this ability to final recommendation.
- The agentic setting mostly achieves better performance than R, validating the advantage of tool-integrated reasoning recommendation. However, TIRR underperforms R on Instruments and Office in the frozen setting, which is expected: without training, tool invocation relies only on language priors and may introduce noisy evidence through suboptimal tool calls. After training, TIRR consistently outperforms R across all datasets, highlighting the necessity of our training method.

5.4 Further Analysis

Effect of PPR Training To further examine the effect of PPR, we track the agent's performance during PPR training in Fig. 3. During PPR training, the agent's preference judgment accuracy on the PPR test set continues to increase. Meanwhile, the agent's H@1 on the final recommendation test set also steadily improves. This verifies that PPR can enhance the model's ability to distinguish fine-grained preferences. It also suggests that the agent implicitly generalizes a more versatile ability, lead-

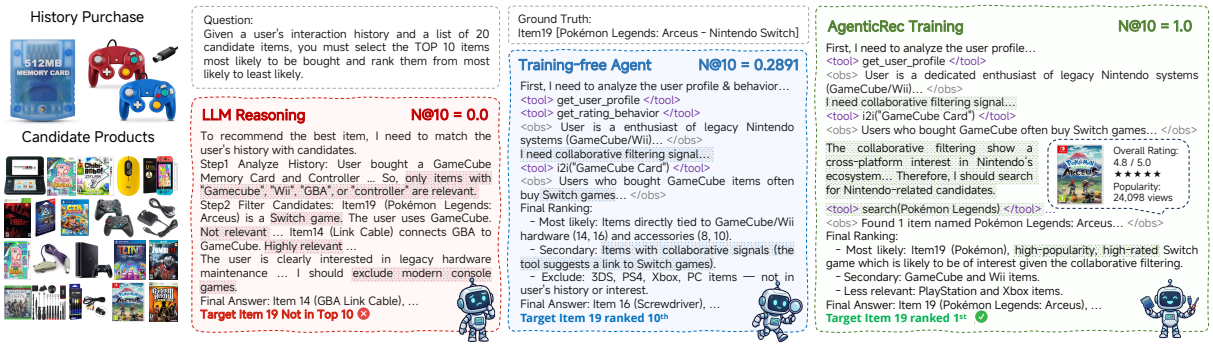


Figure 6: An example showing the effectiveness of AGENTICREC. Textured highlights denote critical reasoning steps that most significantly influence the final ranking and illustrate the differences between methods.

Data	Avg. Latency (s)	Avg. Tool Time (s)
CDs	16.233	0.010
Games	18.260	0.022
Office	20.678	0.019
Instruments	17.074	0.013

Table 3: Average latency statistics for our method.

ing to more accurate recommendations.

Token Count Fig. 4 illustrates the dynamics of token count and average H@10 of rollout trajectories during first-stage training on the Games dataset. During training, the agent is encouraged to produce longer reasoning chains and conduct deeper exploration, leading to recommendations that better satisfy user preferences. Meanwhile, the dialogue token cost remains within an acceptable range. More statistics about token cost are in Appendix D.

Latency We report the average latency per recommendation request on a single NVIDIA A800 GPU using vLLM. As shown in Table 3, the latency remains practical for an agentic recommendation framework. Unlike conventional recommendation pipelines optimized for strict real-time ranking, AGENTICREC targets interactive and reasoning-intensive recommendation scenarios, where richer reasoning and evidence-grounded decision making are often preferred over millisecond-level response latency.

Scaling Backbone LLM To examine the scalability of AGENTICREC with respect to model capacity, we conduct experiments using Qwen3 (Yang et al., 2025) across varying sizes (1.7B, 4B, and 8B). As shown in Fig. 5, performance improves consistently as model size increases across datasets and evaluation metrics, indicating that AGENTICREC can effectively utilize increased model capacity to enhance performance.

5.5 Case Study

To intuitively demonstrate the effectiveness of AGENTICREC, we present a sample from the Games dataset in Fig. 6. The user history consists of legacy Nintendo purchases (GameCube accessories), while the actual next item purchased is *Pokémon Legends: Arceus*, a game for the Nintendo Switch. The vanilla reasoning LLM exhibits an over-reliance on literal string matching (e.g., strictly filtering for keywords like *GameCube*), failing to mine latent interests or grasp the intrinsic semantics of items. By invoking tools, the training-free agent utilizes collaborative signals to improve ranking quality; however, without RTA training and PPR refinement, it fails to further explore nuanced differences among items and remains trapped by literal semantics in the final decision, resulting in suboptimal performance.

In contrast, AGENTICREC leverages collaborative signals to astutely infer the user's latent interest in the Nintendo ecosystem. Building on this insight, it proactively invokes tools to acquire detailed specifications of promising candidates, enabling fine-grained differentiation for ranking. Ultimately, it flexibly synthesizes multi-source information to deliver precise recommendations.

6 Conclusion

In this paper, we proposed AGENTICREC, which casts agentic recommendation as tool-integrated reasoning workflow and optimizes it via a two-stage paradigm: Recommendation-Oriented Trajectory Activation activates agentic recommendation ability, while Progressive Preference Refinement sharpens preference discrimination. Experiments confirm the effectiveness of AGENTICREC.

Limitations

Limited by computational resources, we did not conduct experiments with larger-scale LLM backbones, such as 72B-level models. Our preliminary experiments show that AGENTICREC benefits from increased backbone capacity, and we expect stronger abilities to emerge when AGENTICREC is instantiated with larger LLMs. In addition, this work focuses on optimizing recommender agents from offline interaction logs. A promising future direction is to further explore self-evolving recommender agents to improve the agent's adaptability in dynamic recommendation environments.

References

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*, pages 1007–1014.

Shihao Cai, Chongming Gao, Haoyan Liu, Wentao Shi, Jianshan Sun, Ruiming Tang, and Fuli Feng. 2025. Mgfrec: Towards reinforced reasoning recommendation with multiple groundings and feedback. *arXiv Preprint*, https://arxiv.org/abs/2510.22888.

Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. 2024. On softmax direct preference optimization for recommendation. In *NeurIPS*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv Preprint*.

Jiabao Fang, Shen Gao, Pengjie Ren, Xiuying Chen, Suzan Verberne, and Zhaochun Ren. 2024. A multi-agent conversational recommender system. *arXiv Preprint*.

Yi Fang, Wenjie Wang, Yang Zhang, Fengbin Zhu, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Reason4rec: Large language models for recommendation with deliberative user preference alignment. *arXiv Preprint*.

Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and challenges in conversational recommender systems: A survey. *arXiv Preprint*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv Preprint*.

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*, pages 299–315.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

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*, volume 14609, pages 364–381.

Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A. Rossi, Branislav Kveton, Dongruo Zhou, Julian J. McAuley, and Lina Yao. 2025a. Towards agentic recommender systems in the era of multimodal large language models. *arXiv Preprint*, https://arxiv.org/abs/2503.16734.

Jiani Huang, Xingchen Zou, Lianghao Xia, and Qing Li. 2025b. Mr.rec: Synergizing memory and reasoning for personalized recommendation assistant with llms. *arXiv Preprint*, https://arxiv.org/abs/2510.14629.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2025c. Recommender AI agent: Integrating large language models for interactive recommendations. *TOIS*, 43(4):96:1–96:33.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *SIGIR*, pages 1785–1795.

Jiacheng Lin, Tian Wang, and Kun Qian. 2025. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *TMLR*, 2025.

Qidong Liu, Xiangyu Zhao, Yejing Wang, Zijian Zhang, Howard Zhong, Chong Chen, Xiang Li, Wei Huang, and Feng Tian. 2025a. Bridge the domains: Large language models enhanced cross-domain sequential recommendation. In *SIGIR*, pages 1582–1592.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, Yifei Hu, Qigen Hu, Xinchen Luo, Lejian Ren, Zixing Zhang, Qianqian Wang, Kuo Cai, Yunfan Wu, Hongtao Cheng, and 7 others. 2025b. Onerec-think: In-text reasoning for generative recommendation. *arXiv Preprint*.

Guangtao Nie, Rong Zhi, Xiaofan Yan, Yufan Du, Xiangyang Zhang, Jianwei Chen, Mi Zhou, Hongshen Chen, Tianhao Li, Ziguang Cheng, Sulong Xu, and Jinghe Hu. 2024. A hybrid multi-agent conversational recommender system with LLM and search engine in e-commerce. In *RecSys*, pages 745–747.

OpenAI. 2026. Openai GPT-5 system card. *arXiv Preprint*, https://arxiv.org/abs/2601.03267.

Qiyao Peng, Hongtao Liu, Hua Huang, Qing Yang, and Minglai Shao. 2025. A survey on llm-powered agents for recommender systems. In *EMNLP*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Junfei Tan, Yuxin Chen, An Zhang, Junguang Jiang, Bin Liu, Ziru Xu, Zhu Han, Jian Xu, Bo Zheng, and Xiang Wang. 2025. Reinforced preference optimization for recommendation. *arXiv Preprint*, https://arxiv.org/abs/2510.12211.

Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Wu Jian, and Yuning Jiang. 2025. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. *arXiv Preprint*.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573.

Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. 2025. User behavior simulation with large language model-based agents. *TOIS*, 43(2):55:1–55:37.

Qi Wang, Jindong Li, Shiqi Wang, Qianli Xing, Runliang Niu, He Kong, Rui Li, Guodong Long, Yi Chang, and Chengqi Zhang. 2024a. Towards next-generation llm-based recommender systems: A survey and beyond. *arXiv Preprint*, https://arxiv.org/abs/2410.19744.

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 *NAACL*, pages 4351–4364.

Chenghao Wu, Ruiyang Ren, Junjie Zhang, Ruirui Wang, Zhongrui Ma, Qi Ye, and Wayne Xin Zhao. 2025. Starec: An efficient agent framework for recommender systems via autonomous deliberate reasoning. In *CIKM*, pages 3355–3365.

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. 2024. A survey on large language models for recommendation. *World Wide Web (WWW)*, 27(5):60.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025a. iagent: LLM agent as a shield between user and recommender systems. In *ACL*, pages 18056–18084.

Xiaochuan Xu, Zeqiu Xu, Peiyang Yu, and Jiani Wang. 2025b. Enhancing user intent for recommendation systems via large language models. *arXiv Preprint*, https://arxiv.org/abs/2501.10871.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *arXiv Preprint*.

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*.

Se-eun Yoon, Zhankui He, Jessica Maria Echterhoff, and Julian J. McAuley. 2024. Evaluating large language models as generative user simulators for conversational recommendation. In *NAACL*, pages 1490–1504.

Runyang You, Yongqi Li, Xinyu Lin, Xin Zhang, Wenjie Wang, Wenjie Li, and Liqiang Nie. 2025. R^2ec: Towards large recommender models with reasoning. *NIPS*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. In *NeurIPS*.

An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024a. On generative agents in recommendation. In *SIGIR*, pages 1807–1817.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian J. McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024b. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *WWW*, pages 3679–3689.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025a. What, how, where, and how well? A survey on test-time scaling in large language models. *arXiv Preprint*.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR*, pages 785–788. ACM.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025b. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE TKDE*, 37(5):2329–2340.

Yu Zhang, Shutong Qiao, Jiaqi Zhang, Tzu-Heng Lin, Chen Gao, and Yong Li. 2025c. A survey of large language model empowered agents for recommendation and search: Towards next-generation information retrieval. *arXiv Preprint*.

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 *SIGIR*, pages 1796–1806.

Yaochen Zhu, Harald Steck, Dawen Liang, Yinhan He, Nathan Kallus, and Jundong Li. 2025. Llm-based conversational recommendation agents with collaborative verbalized experience. *EMNLP*, pages 2207–2220.

A Prompts Used in AgenticRec

Following are the prompts used in AgenticRec:

System Prompt for AgenticRec

You are a helpful recommendation system assistant equipped with external tools. You can optionally call the tools to assist with the task.

Tools

You may call one or more functions to assist with the user query. You are provided with function signatures within `<tools></tools>` XML tags:

```
<tools>
{tools_schema}
</tools>
```

For each function call, return a json object with function name and arguments within `<tool_call></tool_call>` XML tags:

```
<tool_call>
{
    "name": <function-name>,
    "arguments": <args-json-object>
}
</tool_call>
```

Prompt for Recommendation

Solve the following problem step by step. You now have the ability to selectively invoke tools to gather information to assist your reasoning.
The last part of your response should be in the following format:

<answer>
`\boxed{'The final answer goes here.'}`
</answer>

User question:
Given a user's interaction history and a list of 20 candidate items, you must select the TOP 10 items most likely to be bought and rank them from most likely to least likely.

Answer Format:
1. Output ONLY the `\boxed{[...]}` result with no additional commentary in your final answer.
   - Example: `\boxed{[3, 17, 8, 1, 12, 5, 19, 2, 14, 6]}`
2. Your final output MUST include exactly 10 item indices (no more, no less).
3. Each index must appear exactly once (no duplicates).
4. All indices must be valid (between 1 and 20).

Here is the history of items the user has bought:
`{history_items_str}`

Here is the list of candidate items:
`{candidate_items_str}`

Remember to place the final answer in the last part using the format:
<answer>
`\boxed{'The final answer goes here.'}`
</answer>

Prompt for PPR Training

Solve the following problem step by step. You need to selectively invoke tools to gather information to assist your reasoning.
The last part of your response should be in the following format:

<answer>
\boxed{'The final answer goes here.'}
</answer>

User question:
Given a user's interaction history and two candidate items, you must determine which item the user is [MORE / LESS] LIKELY to buy next.

Answer Format:
1. Output ONLY the \boxed{A} or \boxed{B} result in your final answer.
2. Your answer must be exactly one of: A or B.

Here is the history of items the user has bought:
{history_items_str}

Here are the two options:
Item A: {item_a_info}
Item B: {item_b_info}

Remember to place the final answer in the last part using the format:
<answer>
\boxed{'The final answer goes here.'}
</answer>

B Theoretical Analysis

B.1 Directional Unbiasedness of RTA Gradient

The RTA gradient estimator theoretically remains directionally unbiased up to a positive scaling factor under the group-average baseline:

Proposition 1 (Directional Unbiasedness of RTA Gradient)**.** *The RTA gradient estimator, which utilizes the list-wise ranking metric $R(r_K, y)$ as the reward and the inclusive group average ranking score as the baseline, provides a directionally unbiased estimate of the gradient for the expected list-wise utility objective $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(r_K, y)]$, up to the positive scaling factor $\frac{G-1}{G}$.*

Proposition 1 confirms that the computed gradient direction aligns mathematically with maximizing the expected list-wise utility (i.e., NDCG@K). The inclusive baseline introduces only a positive scaling factor $\frac{G-1}{G}$ when $G > 1$, which acts as a learning rate dampener without altering the optimization direction. This directional unbiasedness is critical for AGENTICREC for two reasons: (1) First, it ensures accurate credit assignment for tool usage, guaranteeing that the "credit" from a high-quality ranking is correctly back-propagated to the specific intermediate reasoning and tool-invocation steps, thereby driving the agent to learn outcome-driven behaviors. (2) Second, it provides stability in sparse feedback environments; by anchoring the gradient update to a group baseline, the optimization focuses on relative list improvement rather than unstable absolute scores, which is essential for convergence when learning from noisy implicit ranking signals.

Proof. In AGENTICREC, the optimization goal is to maximize the expected quality of the generated top-K list r_K given user context x_u. The trajectory τ encompasses the entire reasoning chain, including tool invocations (Act) and observations (Obs), culminating in the final ranking action r_K. Thus, the objective is:

$$\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(r_K, y) \cdot \nabla \log \pi_\theta(\tau)] \quad (5)$$

where $R(r_K, y)$ is the list-wise metric (i.e., NDCG@K) derived from implicit feedback y (Eq. 2).

To address the high variance inherent in list-wise ranking rewards, we employ the GRPO estimator using a group of G sampled trajectories $\{\tau^{(g)}\}_{g=1}^{G}$. The estimated gradient is:

$$\nabla \hat{J}_{GRPO}(\theta) = \frac{1}{G}\sum_{g=1}^{G} (R(r_K^{(g)}, y) - b)\nabla \log \pi_\theta(\tau^{(g)}) \quad (6)$$

where the baseline $b = \frac{1}{G}\sum_{j=1}^{G} R(r_K^{(j)}, y)$ is the average list-wise utility of the sampled group.

With the inclusive group-average baseline, the baseline term does not vanish exactly because b contains the reward of the same trajectory $\tau^{(g)}$. Let $R^{(g)} = R(r_K^{(g)}, y)$ and $z^{(g)} = \nabla \log \pi_\theta(\tau^{(g)})$. Using the likelihood ratio identity, we have:

$$\begin{aligned}
\mathbb{E}_\tau[\nabla \log \pi_\theta(\tau)] &= \int \pi_\theta(\tau) \cdot \nabla \log \pi_\theta(\tau)\, d\tau \\
&= \int \pi_\theta(\tau) \cdot \frac{\nabla \pi_\theta(\tau)}{\pi_\theta(\tau)}\, d\tau \\
&= \int \nabla \pi_\theta(\tau)\, d\tau \\
&= \nabla \left(\int \pi_\theta(\tau)\, d\tau \right) \\
&= \nabla(1) = 0,
\end{aligned} \quad (7)$$

where $\int \pi_\theta(\tau)\, d\tau = 1$ since the integral of any probability distribution over its entire domain must sum to 1.

Therefore, for each g,

$$\begin{aligned}\mathbb{E}[b \cdot z^{(g)}] &= \mathbb{E}\left[\frac{1}{G}\sum_{j=1}^{G} R^{(j)} z^{(g)}\right] \\ &= \frac{1}{G}\mathbb{E}[R^{(g)} z^{(g)}] + \frac{1}{G}\sum_{j\neq g}\mathbb{E}[R^{(j)}]\mathbb{E}[z^{(g)}] \\ &= \frac{1}{G}\nabla J(\theta).\end{aligned} \tag{8}$$

It follows that

$$\begin{aligned}\mathbb{E}[\nabla \hat{J}_{GRPO}(\theta)] &= \frac{1}{G}\sum_{g=1}^{G}\left(\mathbb{E}[R^{(g)} z^{(g)}] - \mathbb{E}[b z^{(g)}]\right) \\ &= \left(1 - \frac{1}{G}\right)\nabla J(\theta) \\ &= \frac{G-1}{G}\nabla J(\theta).\end{aligned} \tag{9}$$

Since $G > 1$, $\frac{G-1}{G}$ is strictly positive. Thus, the gradient direction remains aligned with the true objective of maximizing NDCG@K, while the group-relative subtraction $(R^{(g)} - b)$ reduces variance by focusing on the relative ranking quality within the group. □

B.2 Error Bound Minimization via Bidirectional Preference Reasoning

The bidirectional preference reasoning in PPR stage can theoretically sharpen the fine-grained preference boundaries among highly confusable candidates:

Proposition 2 (Error Bound Minimization via Bidirectional Preference Reasoning)**.** *Optimizing the bidirectional preference reasoning objective on mined hard negative pairs (c^+, c^-) minimizes the upper bound of the pairwise ranking error probability $P\left(rank_{r_K}(c^-) < rank_{r_K}(c^+)\right)$.*

Proposition 2 shows that, by explicitly training on ranking violations (where initially $\Delta s = s(c^+) - s(c^-) < 0$) and applying the bidirectional update, the agent maximizes the score margin Δs. The "Negative Direction" task provides an additional gradient flow explicitly penalizing the salient features of c^- that caused the violation, thereby tightening the error bound more effectively than positive-only supervision alone.

Proof. Let $s(c|x_u)$ be the agent's scoring potential for an item c given context x_u and we use $s(c)$ for short if there is no ambiguity. A ranking violation occurs if $s(c^-) > s(c^+)$ for a hard negative c^-. We define the score difference as:

$$\Delta s = s(c^+) - s(c^-). \tag{10}$$

A ranking error occurs when $\Delta s < 0$.

The fundamental goal of ranking optimization is to minimize the misordering of pairs. The discrete pairwise ranking error is given by the indicator function:

$$\mathcal{L}_{0-1}(\Delta s) = \mathbb{I}(\Delta s < 0) = \begin{cases} 1 & \text{if } \Delta s < 0 \\ 0 & \text{if } \Delta s \geq 0 \end{cases} \tag{11}$$

Directly optimizing $\mathcal{L}_{0-1}$ is intractable due to its discontinuity and zero gradients almost everywhere, which prevents effective gradient-based updates via the agent's policy.

For a pair (c^+, c^-), our bidirectional preference reasoning creates two complementary tasks:

- **Positive Direction:** Maximize likelihood of choosing c^+. Let this probability be:

$$P_{pos} = \frac{e^{s(c^+)}}{e^{s(c^+)} + e^{s(c^-)}}$$

- **Negative Direction:** Maximize likelihood of rejecting c^- (identifying it as "less likely"). This effectively maximizes:

$$P_{neg} = \frac{e^{-s(c^-)}}{e^{-s(c^+)} + e^{-s(c^-)}} = \frac{e^{s(c^+)}}{e^{s(c^+)} + e^{s(c^-)}}$$

Optimizing these tasks via GRPO is equivalent to minimizing the negative log-likelihood (cross-entropy) of the preference probability $P(c^+ \succ c^-)$. For the pair (c^+, c^-), the combined loss $\mathcal{L}_{Bi}$ is proportional to:

$$\begin{aligned}\mathcal{L}_{Bi} &\approx -\log(P_{pos}) - \log(P_{neg}) \\ &= -2\log\left(\frac{1}{1 + e^{-(s(c^+) - s(c^-))}}\right) \\ &= 2\log(1 + e^{-\Delta s}).\end{aligned} \tag{12}$$

The above formulation represents the logistic loss.

We observe that the derived logistic loss function $\mathcal{L}(\Delta s) = 2\log(1+e^{-\Delta s})$ serves as a convex upper bound to the 0-1 ranking error loss $\mathcal{L}_{0-1}$:

- **Upper Bound:** For any Δs, $2\log(1 + e^{-\Delta s}) \geq 2\log(2) \cdot \mathbb{I}(\Delta s < 0)$. Specifically, when $\Delta s < 0$ (an error), the logistic loss grows linearly with the magnitude of the violation ($-\Delta s$), imposing a heavy penalty. When $\Delta s > 0$ (correct ranking), the loss approaches zero smoothly.

- **Convexity:** Unlike the step function $\mathbb{I}(\cdot)$, the logistic loss $\mathcal{L}(\Delta s)$ is smooth and convex. This ensures that the optimization landscape is well-behaved, providing non-vanishing gradients even when the ranking is currently incorrect (i.e., when $\Delta s \ll 0$).

By mining hard negatives where $\Delta s < 0$ and applying the bidirectional update, AGENTICREC effectively minimizes $2\log(1 + e^{-\Delta s})$. Since this function is a tight convex upper bound to the indicator error function, minimizing the bidirectional loss theoretically guarantees the minimization of the empirical pairwise ranking error rate, thereby sharpening the fine-grained preference boundaries among highly confusable candidates. □

C Experimental Details

C.1 Datasets and Preprocess

We conduct experiments on 4 subsets of the Amazon Reviews 2023 corpus (Liu et al., 2025a), including CDs and Vinyl (CDs), Musical Instruments (Instruments), Office Products (Office), and Video Games (Games). To keep the training time tractable, we restrict the interaction data to the period from October 2022 to October 2023. We adopt a common chronological split with a ratio of 8:1:1 for training/validation/testing by timestamp, where interactions are sorted by time and split accordingly. The statistics of these processed datasets are summarized in Tab. 4.

For interaction behavior sequence construction, we follow Liao et al. (2024) to cap the maximum length of user interaction history to 10. For candidate construction, each instance includes one ground-truth next item and 19 randomly sampled negatives, forming 20 candidates in total. When sampling negatives, we exclude all items appearing in the user's historical interaction sequence to avoid overlap with previously interacted items. We then randomly shuffle the candidate items before feeding them to the model, so that the input order does not introduce positional bias.

Table 4: Statistics of the preprocessed datasets.

Dataset	#Users	#Items	#Inters.	Sparsity
CDs	5,437	8,785	13,826	99.71%
Instruments	7,593	5,279	15,746	99.61%
Office	27,130	11,511	47,333	99.85%
Games	6,251	3,003	11,457	99.39%

C.2 Baselines

Conventional Sequential Recommenders include classic baselines Caser (Tang and Wang, 2018), GRU4Rec (Hidasi et al., 2016), and SASRec (Kang and McAuley, 2018), which model user preferences from interaction sequences using convolutional, recurrent, and self-attention architectures, respectively. We further consider ReaRec (Tang et al., 2025), an inference-enhanced recommender that introduces additional reasoning steps at test time.

Training-free LLM-based Recommenders mainly rely on prompting or in-context reasoning without parameter updates. In particular, LLMRank (Hou et al., 2024) directly performs candidate ranking by prompting an off-the-shelf LLM with user history and item descriptions, serving as a strong zero-shot ranking baseline. InteRecAgent (Huang et al., 2025c) represents autonomous recommender agents that leverage an LLM to interact with external tools (i.e., SASRec) for recommendation decision making, while remaining training-free for the LLM backbone.

Trainable LLM-based Recommenders adapt LLMs to recommendation via fine-tuning or reinforcement learning. TALLRec (Bao et al., 2023) improves recommendation by tuning LLMs with recommendation-oriented supervision under limited data. LLaRA (Liao et al., 2024) further integrates textual and ID-style representations to align LLMs with recommendation signals. S-DPO (Chen et al., 2024) introduces a multi-negative, softmax-style DPO objective to better leverage implicit preference data beyond positive-only supervision. ReRe (Tan et al., 2025) applies constrained beam search and preference rewards to improve recommendation result.

C.3 Implementation Details

For AGENTICREC, We use Qwen3-4B-Instruct-2507 (Yang et al., 2025) as the backbone. The training follows the two-stage training with 3 and 1 epochs, respectively, and we set the group size to 8, the maximum generation length to 6,144, the batch size to 64, the learning rate to 1×10^{-6}, and the rollout temperature to 0.6.

For sequential recommendation baselines, we use a batch size of 1,024 and optimize all models with binary cross-entropy loss and a learning rate of 0.001. For ReaRec, the number of reasoning steps

during training is set to 3. For training-free LLM-based baselines, we use GPT-4 as the LLM for all experiments. For trainable LLM-based baselines, to ensure a fair comparison, we adopt Qwen3-4B-Instruct-2507 as the backbone across all methods, the same as AGENTICREC, with a learning rate of 1×10^{-5} and a batch size of 128, while keeping other hyperparameters and configurations as their default settings. Since TALLRec (Bao et al., 2023) was originally designed for point-wise ranking, we adapt it to our setting by reformulating the task as selecting the next item from a given candidate set. Similarly, ReRe (Tan et al., 2025) is adapted to our setting by constraining the model to recommend from a given candidate set. For baselines that explicitly output item relevance scores, we obtain the ranked list by scoring all candidate items and sorting them in descending order. For TALLRec, LLaRA, S-DPO and ReRe, we rank the candidate items according to their joint generation probability.

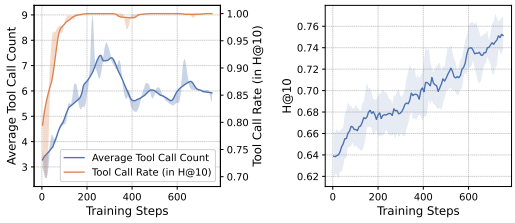
All experiments are conducted on 4 NVIDIA A800 GPUs (80GB memory each).

D Additional Empirical Results

Analysis on Tool Invocation Fig. 7(a) shows the statistics of tool invocation during the first stage training (RTA) on Office. The tool invocation rate among positively rewarded trajectories (orange line) increases rapidly in early training and remains consistently high thereafter, indicating that the policy quickly learns to leverage tools for effective decision-making. Meanwhile, the average number of tool calls (blue line) gradually rises and then stabilizes, suggesting that the model develops a relatively stable tool use strategy rather than indiscriminately increasing tool calls.

Fig. 7(b) shows the change of H@10 on training trajectories during the first stage training, reflecting steady improvements in the learned policy. The synchronized progression between stabilized tool use and increasing H@10 suggests that our training method brings more effective tool planning, gradually leading to better recommendation.

Analysis of Group Size As shown in Fig. 8, increasing the group size in RTA improves recommendation performance by providing richer intra-group comparisons for more stable list-wise credit assignment. However, the gains gradually saturate with larger groups. Thus, a moderate group size offers a practical trade-off between training stability and efficiency.



(a) Tool use (b) Recommendation quality

Figure 7: Training statistics on the Office dataset: (a) tool usage statistics, where the blue line denotes the average number of tool calls per trajectory and the orange line indicates the percentage of positively rewarded trajectories that have tool invocation; (b) recommendation performance measured by H@10 during training.

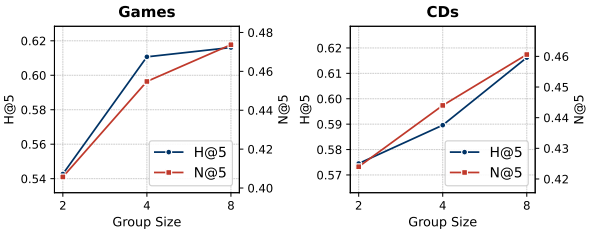


Figure 8: Effect of group size in RTA

Token Cost Fig. 9 reports the token cost statistics of AGENTICREC across datasets. Naive LLM denotes the token consumption when directly using the LLM to reason and generate recommendation results. AGENTICREC denotes the token count during recommendation generation after training under the tool-integrated reasoning recommendation paradigm, including *Think*, *Act*, *Obs*, and *Rec* tokens. Compared with standard LLM reasoning, the token cost of AGENTICREC remains relatively controllable.

E Tool Example

The following are examples of tool invocation in AGENTICREC:

- **User Profile Tool:** Fig. 10 illustrates the user profile tool. We utilize the Qwen3-4B-Instruct-2507 (consistent with our backbone) to pre-generate user profiles. The input consists of the user's interaction history prior to the timestamp of each sample in our processed dataset.
- **Item Information Tool:** As shown in Fig. 11, `item_info_search` tool retrieves detailed information about a specific item based on the dataset's item metadata. Additionally, we construct a candidate analysis function to support another tool `candidates_analyze` that aggregates candidate items by category or price, providing

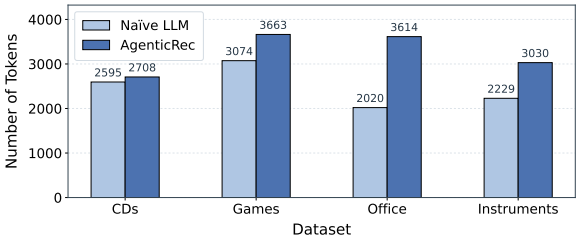


Figure 9: Full token cost statistics.

a structured summary that enables the agent to quickly grasp the composition of the candidate list. This dual-granularity design facilitates high-quality list-wise ranking by combining specific item details with a candidate overview.

- **Behavioral Statistics Tool:** Fig. 12 illustrates the return information of this tool. Upon invocation, it computes statistics on relevant metadata fields to generate statistical summaries of user recent behavior. Specifically, for *Session Behavior* (`get_session_behavior`), the user's dynamic short-term interest patterns are extracted using item metadata from recent sessions; meanwhile, for *Rating Behavior* (`get_rating_behavior`), items rated by the user are categorized into high, neutral, and low groups to serve as explicit preference signals.
- **Collaborative Information Tool:** Fig. 13 illustrates the information returned by the tool. We employ SASRec(Kang and McAuley, 2018) on the training set to model collaborative information and construct the embedding space, which is utilized for collaborative retrieval.

User Profile Tool

▶ **Tool Call**

```
<tool_call>
{
  " name" : " get_user_profile" ,
  " arguments" : {}
}
</tool_call>
```

◀ **Tool Output**

The user is a dedicated enthusiast of Nintendo legacy systems, particularly favoring Switch controllers and related accessories. They consistently prefer high-functionality products with features like vibration, motion control, and macro capabilities.

Figure 10: An example of user profile tool invocation.

Item Information Tool

▶ **Tool Call**

```
<tool_call>
{
    " name" : " item_info_search" ,
    " arguments" : {
        " item_name" : " Plants vs.
            Zombies Garden Warfare 2 -
            Xbox One"
    }
}
</tool_call>
```

◀ **Tool Output**

Found 1 item named "Plants vs. Zombies Garden Warfare 2 - Xbox One" (Categories: Xbox One, Games / Store: Electronic Arts / Price: $16.99)
Overall Rating: 4.6 / 5.0
Popularity: 1625 reviews

▶ **Tool Call**

```
<tool_call>
{
    " name" : " candidates_analyze" ,
    " arguments" : {}
}
</tool_call>
```

◀ **Tool Output**

Candidate Analysis by Category:
- Legacy Systems > Nintendo Systems > Game Boy Systems > Game Boy:
  – Item 13: [EverDrive GBA Mini]
- Legacy Systems > Nintendo Systems > Nintendo DS > Games:
  – Item 4: [Call of Duty: Black Ops]
  – Item 17: [Cake Mania NDS]
- Legacy Systems > PlayStation Systems > PlayStation 2 > Accessories:
  – Item 6: [PS2 Lava Glow 2.4 GHZ RF Wirel...]
  – Item 12: [Wireless 2.4G Controller Gamep...]
- Nintendo Switch > Games:
  ...

Figure 11: An example of item information tool.

Behavioral Statistics Tool

▶ Tool Call

```
<tool_call>
{
    " name" : " get_session_behavior" ,
    " arguments" : {}
}
</tool_call>
```

◀ Tool Output

Below are the extracted interests for the user's most recent 2 sessions, listed chronologically from oldest to newest:
3.3h ago, the user was interested in Games, Legacy Systems.
1.8h ago, the user was interested in PlayStation 4, Games.
This information reflects the user's historical interests, interest evolution, or potential current focus.

▶ Tool Call

```
<tool_call>
{
  " name" : " get_rating_behavior" ,
  " arguments" : {}
}
</tool_call>
```

◀ Tool Output

User's historical rating records are as follows:
Five-star items (=5):
PowerA MOGA XP5-X Plus Bluetooth Controller for Mobile & Cloud Gaming on Android/PC...
Neutral items (>=3):
sunwaytek Syozen Z2 Wireless Mechanical Controller for Nintendo Switch...
Wireless Switch Pro Controller with 9 Color Adjustable LED/Unique Crack/Turbo/Vibration/-Motion Control/Wake-Up...
Dishonored: The Death of the Outsider...
Low-rated items (<3):
PowerA Enhanced - Seafoam Fade, Gamepad, Wired Video Game Controller, Gaming Controller, Xbox Series X|S, Xbox One (Only at Amazon)...

Figure 12: An example of behavioral statistics tool.

Collaborative Information Tool

▶ Tool Call

```
<tool_call>
{
    " name" : " get_similar_items" ,
    " arguments" : {
    " item_title" : " Tiger Woods PGA
        Tour"
    }
}
</tool_call>
```

◀ Tool Output

Based on collaborative filtering, users who enjoy the following items are more likely to purchase [Tiger Woods PGA Tour]:
[1] Nintendo Switch Mario Kart 8 Deluxe Wheel
[2] GIRIAITUS Wii Motion Plus Adapter
[3] Resident Evil 5: Gold Edition...

▶ Tool Call

```
<tool_call>
{
    " name" : " get_similar_users" ,
    " arguments" : {}
}
</tool_call>
```

◀ Tool Output

The following users have similar preferences to the current user. Their recent purchases include:
User [1]: [Madden NFL 10 - Playstation 3], [Dirt - Xbox 360], [NBA Hangtime - PlayStation]...

Figure 13: An example of collaborative information tool invocation.