Reinforced Preference Optimization for Reasoning-Augmented Recommendations

Jingtong Gao[1*] **Zeyu Song**[2] **Chi Lu**[2] **Xiaopeng Li**[1]
Derong Xu[1] **Maolin Wang**[1] **Peng Jiang**[2] **Kun Gai**[2] **Qingpeng Cai**[2†] **Xiangyu Zhao**[1†]
[1]City University of Hong Kong [2]Kuaishou Technology
{jt.g,xiaopli2-c,derongxu2-c,Morin.wang}@my.cityu.edu.hk, xianzhao@cityu.edu.hk
gai.kun@qq.com, {songzeyu,luchi,jiangpeng,caiqingpeng}@kuaishou.com

Abstract

Recommender systems are critical for delivering personalized content across digital platforms, and recent advances in Large Language Models (LLMs) offer new opportunities to enhance them with richer world knowledge and explicit reasoning capabilities. With the help of reasoning knowledge, recommendations can better infer users' underlying intents, adapt to evolving preferences, and leverage semantic relationships for improved accuracy and interpretability. However, existing reasoning-based recommendation methods often fail to fully align the LLM's reasoning process with recommendation-specific objectives due to structural disruption during integration and difficulties in translating free-form generation into accurate item predictions. In this paper, we introduce RPORec, a reinforced preference optimization framework that unifies an LLM backbone's reasoning ability with a dedicated recommendation head (Rechead) for precise item retrieval. RPORec comprises two stages: (1) Reasoning-Augmented Recommendation Modeling, where high-quality Chain-of-Thought (CoT) reasoning is generated and used as auxiliary knowledge to guide the Rechead in learning recommendation-specific representations; and (2) Advanced Reasoning Refinement and Alignment, in which the trained Rechead produces verifiable rewards to fine-tune the LLM backbone via reinforcement learning, enhancing reasoning quality, structural consistency, and task relevance. Extensive experiments on public benchmarks and large-scale online deployments show that RPORec consistently outperforms state-of-the-art LLM-based recommendation methods, demonstrating the effectiveness of reasoning-augmented recommendation modeling in real-world systems.

1 Introduction

Recommender systems have become indispensable in modern digital platforms, delivering personalized content in domains such as e-commerce, streaming media, and social networking [17, 15]. By accurately modeling user preferences, they enhance user experience, sustain engagement, and generate substantial commercial value [26, 37]. Recently, Large Language Models (LLMs) have shown remarkable abilities in contextual understanding, complex reasoning, and text generation, enabling the interpretation of user intents and the integration of diverse contextual signals [36, 20]. Consequently, recommendation research is gradually transitioning from small deep models toward LLM-based approaches that aim to combine the extensive world knowledge of LLMs with high-precision preference modeling [38, 32].

[*]This work was completed at Kuaishou Technology.
[†]Corresponding authors.

Preprint.

In recent studies [30, 8], reasoning has emerged as a defining capability that distinguishes LLMs from conventional neural networks. It enables the decomposition of complex problems, multi-hop context integration, and coherent inference beyond surface-level pattern matching. In recommendation [34], such reasoning helps LLM-based models infer user intent, track preference evolution, and leverage world knowledge to uncover semantic relations between users and items. By producing explicit reasoning traces such as Chains-of-Thought (CoTs), LLMs can make recommendations that are not only accurate but also more transparent, interpretable, and adaptive to dynamic user contexts. Reasoning-aware modeling therefore holds strong promise for advancing recommender systems.

Existing reasoning-based recommendation methods can be broadly divided into two categories. (1) Joint optimization methods integrate LLM hidden states with recommendation modules for end-to-end training [2, 33], aiming to exploit latent reasoning ability through task-oriented finetuning. Representative methods such as R^2ec [33] jointly optimize reasoning and prediction through hidden-state coupling. While this design tightly connects reasoning and recommendation signals, directly updating hidden states for downstream recommendation objectives may make explicit reasoning harder to preserve, gradually affecting both interpretability and reasoning quality. (2) Fine-tuned generative methods train LLMs to generate recommendations in natural language [27], potentially with CoT-based inference. Representative methods such as ReRe [27] rely on direct generation for recommendation, while LatentR^3 [35] further shifts reasoning optimization into latent space without preserving explicit CoTs. Although these approaches better preserve or exploit reasoning, aligning free-form text generation for item retrieval remains difficult and may also lead to performance degradation, especially for training-absent items. Even with Semantic IDs (SIDs) [23, 35], the mismatch between recommendation-specific tokenization and the LLM vocabulary creates a persistent semantic gap. Overall, existing methods still face trade-offs among explicit reasoning preservation, optimization stability, and alignment with the structured retrieval objectives of recommendation.

To bridge these gaps, we propose **Reinforced Preference Optimization for Reasoning-Augmented Recommendations (RPORec)**, which combines LLM CoT reasoning with a specialized recommendation head (Rechead) for precise item prediction and retrieval. Instead of coupling the LLM backbone and Rechead through hidden states, RPORec uses textual outputs as the interface, preserving reasoning integrity while enabling task-specific recommendation modeling and direct item retrieval, thereby reducing semantic mismatch. However, constructing such a reasoning-augmented framework remains non-trivial due to three key challenges: (1) jointly optimizing the LLM backbone and Rechead toward recommendation objectives; (2) preserving and improving the backbone's reasoning quality; and (3) filtering noisy reasoning context while retaining useful signals for recommendation.

RPORec tackles these challenges with three key designs. First, we introduce an **iterative optimization pipeline**: we first train the Rechead with a frozen LLM backbone, and then refine the backbone using verifiable reward signals from the Rechead via Reinforcement Learning with Verifiable Rewards (RLVR) [10]. This two-stage design prevents recommendation-specific gradients from directly distorting LLM hidden representations while supporting both accurate recommendation modeling and high-quality reasoning. Second, we develop a **reasoning-augmented recommendation modeling** framework with a dedicated reasoning-aware Rechead that effectively integrates high-quality reasoning signals into a retrieval-based architecture for fine-grained next-item prediction. Third, we propose an **reasoning refinement and alignment** strategy based on carefully designed reward functions to improve both reasoning quality and task alignment of the LLM backbone.

In summary, our contributions are:

- We introduce RPORec, a novel framework that integrates high-quality LLM reasoning text into recommendation, enabling reasoning-augmented decision-making.
- We design an iterative optimization pipeline with reasoning-augmented recommendation modeling and advanced reasoning refinement and alignment strategies to improve reasoning quality and task alignment for improved performance.
- Extensive experiments on public benchmarks and deployment on a global online platform demonstrate that RPORec significantly outperforms state-of-the-art recommendation methods.

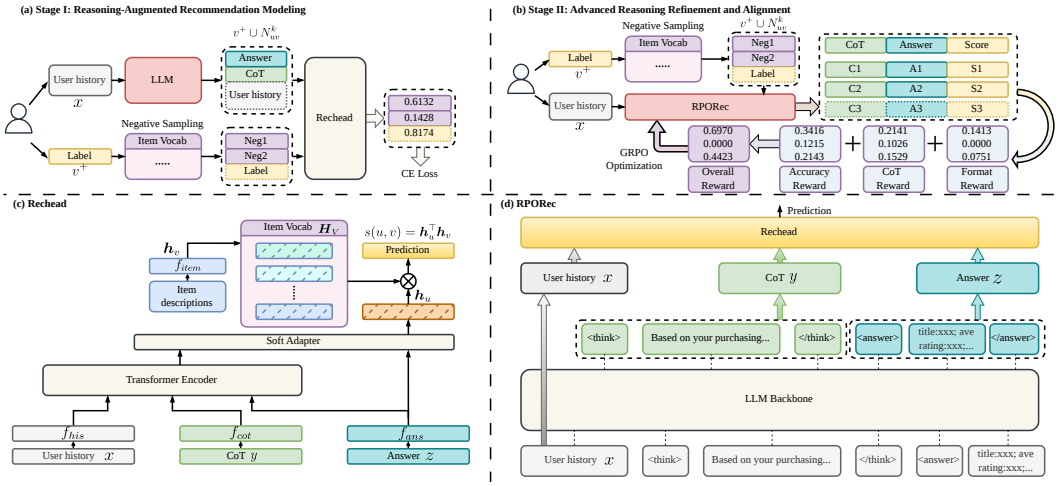


Figure 1: Overview of RPORec. The full structure of RPORec is depicted in (d) with detailed Rechead structure in (c). The optimization processes of stage I and II are illustrated in (a) and (b).

2 RPORec Framework

In this section, we provide an overview of RPORec and detail its key components. We also provide the preliminary knowledge of GRPO and LLM for Recommendations in Appendix A.

As shown in Figure 1(d), RPORec consists of a reasoning-aware LLM backbone and a reasoning-augmented Rechead (Figure 1(c)) to improve recommendations through explicit text-based reasoning and direct task-specific item retrieval. The framework follows an iterative optimization paradigm with two stages: Reasoning-Augmented Recommendation Modeling (Figure 1(a)), which learns the recommendation head (Rechead), and Advanced Reasoning Refinement and Alignment (Figure 1(b)), which further improves and aligns the LLM backbone for recommendation tasks. This design jointly preserves reasoning coherence and prediction precision in downstream recommendation.

Specifically, RPORec couples two core components:

- **Rechead**: provides reasoning augmented recommendation through modeling high-quality LLM reasoning CoTs as auxiliary information for recommendation task-specific modeling and prediction.
- **LLM Backbone**: generates structured outputs containing a reasoning segment with Chain-of-Thoughts (CoTs) and a predicted answer segment describing recommended items with their titles and attributes as shown in Figure 3.

During training, the framework operates in two iterative stages:

- **Stage I – Reasoning-Augmented Recommendation Modeling.** The LLM backbone is frozen, and the Rechead is trained to model recommendation signals from user histories together with the generated CoTs and answer segments.
- **Stage II – Advanced Reasoning Refinement and Alignment.** The Rechead is then frozen and used as a stable reward source to refine the LLM backbone via reinforcement learning with multiple reward signals.

Together, the two stages establish a reasoning-augmented recommendation framework that improves recommendation quality through high-quality CoT reasoning.

2.1 Reasoning-Augmented Recommendation Modeling

Leveraging reasoning knowledge for recommendation is non-trivial: directly optimizing recommendation objectives on LLM hidden states may disrupt intrinsic reasoning, while naively encoding CoTs for generic recommenders introduces substantial noise. We therefore propose a lightweight reasoning-augmented recommendation head (Rechead), which treats CoTs as an auxiliary signal in a user-item representation-based retrieval framework without requiring text- or token-level alignment. As shown in Figure 1(a) and (c), Rechead contains four text encoders, a Transformer encoder block,

and a soft adapter layer that controls CoT contribution and suppresses irrelevant reasoning in the latent space.

2.1.1 Rechead Structure

As shown in Figure 1(c), Rechead takes the user history x, the CoT y, and the final answer z (e.g., predicted item title and attributes). Each text is embedded by an encoding function f composed of pre-trained small sentence transformers and feed-forward networks:

$$\boldsymbol{r}_x = f_{his}(x), \quad \boldsymbol{r}_y = f_{cot}(y), \quad \boldsymbol{r}_z = f_{ans}(z). \tag{1}$$

where $\boldsymbol{r}_x, \boldsymbol{r}_y, \boldsymbol{r}_z \in \mathbb{R}^d$. Items are also encoded from their textual descriptions by a pre-trained small sentence transformer f_{item} into dense vectors $\boldsymbol{h}_v$, which are stored in the item vocabulary matrix $\boldsymbol{H}_V \in \mathbb{R}^{|V| \times d}$.

To handle entangled CoTs and answers, or missing answers, Rechead selects a primary representation according to whether a reliable answer z is available:

$$\boldsymbol{r}_{sel} = \begin{cases} \boldsymbol{r}_z, & \text{if the answer } z \text{ is parsed successfully,} \\ \boldsymbol{r}_x, & \text{otherwise.} \end{cases} \tag{2}$$

This allows Rechead to fall back to direct history modeling when the explicit answer is malformed or absent.

Although CoTs may contain useful reasoning knowledge, their length and limited controllability introduce substantial noise, making raw CoTs ineffective for recommendation. We therefore model interactions among user history, the CoT, and the primary representation with a lightweight transformer encoder, which filters irrelevant content and produces a reasoning-augmented representation:

$$\boldsymbol{r}_{rea} = \mathrm{TransformerEncoder}([\boldsymbol{r}_x;\, \boldsymbol{r}_y;\, \boldsymbol{r}_{sel}])\,. \tag{3}$$

A gating network then adaptively regulates the contribution of the reasoning-augmented representation to the final representation $\boldsymbol{h}_u \in \mathbb{R}^d$:

$$\gamma_0 = \gamma \cdot \sigma(f_{gate}([\boldsymbol{r}_{sel} \parallel \boldsymbol{r}_{rea}]) - 0.5), \quad \boldsymbol{h}_u = \gamma_0 \cdot \boldsymbol{r}_{rea} + \boldsymbol{r}_{sel}, \tag{4}$$

where $\gamma \in (0, 1)$ is a hyperparameter, σ is the sigmoid function, and $\parallel$ denotes concatenation. This gate curbs noisy or off-topic CoTs while preserving useful relational cues as auxiliary augmentation.

Finally, the recommendation score is computed by a dot product in the shared embedding space, and the item with the highest score is retrieved:

$$s(u, v) = \boldsymbol{h}_u^\top \boldsymbol{h}_v. \tag{5}$$

2.1.2 Rechead Optimization

As illustrated in Figure 1(a), we precompute the CoT segment y and answer segment z from the LLM for each training sample $[x, v^+]$ to construct the training data for Rechead. During inference, predictions are made over the entire item space $\boldsymbol{H}_V$. Exhaustive scoring is expensive during training, so we adopt negative sampling [18] and compute the Cross-Entropy (CE) loss [19] on k sampled negatives N_{uv}^k together with the positive item v^+ for each training instance:

$$\mathcal{L}_{\mathrm{rec}} = -\log \frac{\exp\big(s(u, v^+)\big)}{\sum_{v \in \{v^+\} \cup N_{uv}^k} \exp\big(s(u, v)\big)}. \tag{6}$$

This formulation enables efficient recommendation training.

In summary, Rechead uses CoTs as auxiliary reasoning signals, stabilizes prediction through adaptive primary selection, and aligns user vectors with pre-encoded items via efficient contrastive learning. This yields a parameter-efficient and robust module for scalable retrieval under noisy reasoning.

2.2 Advanced Reasoning Refinement and Alignment

With Rechead established for task-specific prediction, we next optimize the LLM backbone. Directly using a pre-trained LLM is insufficient because it is often misaligned with recommendation objectives and tends to produce lengthy, low-quality reasoning chains [13]. We therefore fine-tune the backbone with GRPO, using the frozen Rechead as a verifier that provides recommendation-specific feedback. Three complementary reward functions jointly optimize output format, reasoning quality, and recommendation accuracy.

2.2.1 Format Reward

To ensure valid Rechead inputs, we adopt a standard format reward [10]. The LLM backbone is prompted to generate "<think>y</think><answer>z</answer>", and the format reward is defined as: Specifically, $r_{fmt} = 1.0$ if the format is correct and $r_{fmt} = 0.0$ otherwise. To discourage extraneous content outside the "think" and "answer" tags, we further introduce a penalty-based reward $r_{clean} = \max(0, 1 - \frac{L_{out}}{\kappa})$, where L_{out} denotes the length of text outside the designated tags, and κ=100 is a hyperparameter controlling the penalty strength.

2.2.2 Accuracy Reward

To align the LLM with recommendation tasks, we design a ranking-based reward. Given a user u with one positive item v^+ and k sampled negatives N_{uv}^k, we compute an NDCG-based reward over the candidate set $\{v^+\} \cup N_{uv}^k$ using the Rechead scores $s(u, v)$ (Eq. 5). The reward is defined from the NDCG value of the ground-truth item as $r_{ndcg} = \mathrm{NDCG}\,@k\,(\mathrm{rank}\,(v^+))$, where $\mathrm{rank}()$ denotes the rank of v^+ over the candidate set.

2.2.3 CoT Reward

The CoT reward is a central component of RPORec, because explicit CoTs serve as the key interface between the LLM backbone and Rechead throughout the framework. Accordingly, beyond format correctness and recommendation accuracy, we explicitly optimize the quality of the reasoning trajectory itself, which also distinguishes RPORec from similar methods that primarily optimize predictions or latent representations. To encourage concise, recommendation-centric reasoning, we use a frozen LLM backbone as a summarizer that maps the original CoT y to a short rationale $\hat{y}$ with designed compression prompts. After obtaining $\hat{y}$, we evaluate the robustness and quality of the original CoT y. Specifically, we measure semantic consistency using a pre-trained sentence transformer $e(\cdot)$ and encourage compression via a length-based reward, with $r_{sim} = \mathbf{1}\{\cos(e(y), e(\hat{y})) > \delta\}$ and $r_{comp} = \mathrm{clip}(\frac{|\hat{y}|}{|y|}, 0, 1)$. where $r_{sim} = 1$ when the cosine similarity between the encoded $\hat{y}$ and y exceeds the threshold δ, and r_{comp} is the clipped compression ratio. The similarity reward r_{sim} encourages the original CoT y to retain its core semantics after summarization, while the compression reward r_{comp} favors concise trajectories that resist further simplification, penalizing redundancy. Together, they steer the LLM backbone toward semantically meaningful, information-dense CoTs that would originally require multiple rounds of summarization and simplification.

To further improve CoT quality from a latent decision-making perspective, we introduce an entropy-based reward. Recent work [31] suggests that token entropy E_t serves as an indicator of generation uncertainty, and that high-entropy tokens—especially those in the top 20%—are particularly critical to the semantic quality of the generated trajectory. This suggests that high-quality reasoning should retain a small number of informative, uncertain decision points without becoming globally unstable.

To characterize CoT quality via entropy E, for a generated CoT T, we define the mean entropy $E_\mu = \frac{1}{|T|}\sum_{t \in T} E_t$ and the top-20% entropy average $E_{20\%}$. The entropy reward is then defined as $r_{\text{ent}} = E_{20\%} - E_\mu$, rewarding trajectories with salient high-information tokens (i.e., higher $E_{20\%}$) while penalizing excessive overall uncertainty (i.e., higher E_μ).

Finally, we aggregate these signals for GRPO training while requiring a non-zero format reward r_{fmt} to enforce strict format generation:

$$r = r_{fmt} \cdot (\alpha_0 r_{fmt} + \alpha_1 r_{clean} + \alpha_2 r_{ndcg} + \alpha_3 r_{sim} + \alpha_4 r_{comp} + \alpha_5 r_{ent}). \tag{7}$$

2.3 Iterative Optimization

To align the components in RPORec with the recommendation objective while avoiding unstable rewards from simultaneously optimizing multiple modules with a non-stationary Rechead, we adopt a two-stage iterative strategy. We first freeze the LLM backbone and train Rechead for the downstream recommendation task. We then freeze Rechead and use it to provide stable, task-specific feedback for LLM reasoning refinement and alignment. The overall optimization procedure of RPORec is summarized in Appendix B.

3 Experiments

In this section, we conduct experiments on three public datasets to investigate the following questions:

- **RQ1:** How does RPORec perform in comparison with different recommendation baselines?
- **RQ2:** What's the contribution of the designed modules and structures in making recommendations?
- **RQ3:** Could RPORec actually amplify LLM's reasoning quality and its effect in recommendation modeling?

In the following subsections, we first describe the evaluation settings. Afterward, we would answer the corresponding questions based on a brief review of our experiment results.

3.1 Experimental Setup

We evaluate RPORec on three Amazon datasets: Musical Instruments, CDs and Vinyl, and Video Games, against representative traditional, LLM-based, and RL-based baselines. Following prior work [27, 33] and industrial experiences, a small-size LLM, Qwen3-0.6B [29], is applied as the LLM backbone for RPORec and all LLM-based baselines to meet the efficiency requirements of recommendation tasks. Detailed dataset statistics, preprocessing settings, baseline descriptions, and implementation details are given in Appendix C.

3.2 Overall Performance (RQ1)

This subsection gives an overall comparison between RPORec and different baselines. The results are depicted in Table 1. From this we can conclude that:

Table 1: Performance comparison of RPORec and baselines. Boldface denotes the highest performance, and underline indicates the second-best performance. "*" indicates the statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over the second-best baseline.

Method	Musical Instruments						CDs and Vinyl						Video Games					
	H@5	N@5	H@10	N@10	H@20	N@20	H@5	N@5	H@10	N@10	H@20	N@20	H@5	N@5	H@10	N@10	H@20	N@20
GRU4Rec	0.0101	0.0063	0.0187	0.0091	0.0320	0.0125	0.0044	0.0028	0.0061	0.0034	0.0088	0.0041	0.0145	0.0096	0.0237	0.0126	0.0347	0.0153
Caser	0.0121	0.0076	0.0215	0.0106	0.0368	0.0144	0.0045	0.0029	0.0066	0.0036	0.0099	0.0044	0.0150	0.0100	0.0236	0.0127	0.0360	0.0158
SASRec	0.0149	0.0096	0.0252	0.0128	0.0394	0.0163	0.0092	0.0056	0.0145	0.0073	0.0212	0.0093	0.0227	0.0145	0.0364	0.0191	0.0563	0.0244
TIGER	0.0158	0.0099	0.0243	0.0123	0.0381	0.0158	0.0071	0.0043	0.0105	0.0052	0.0156	0.0065	0.0165	0.0082	0.0245	0.0114	0.0382	0.0154
BIGRec	0.0132	0.0082	0.0228	0.0117	0.0367	0.0152	0.0058	0.0035	0.0095	0.0047	0.0123	0.0051	0.0183	0.0091	0.0267	0.0124	0.0359	0.0145
D^3	0.0162	0.0101	0.0248	0.0126	0.0395	0.0164	0.0074	0.0045	0.0105	0.0052	0.0164	0.0068	0.0201	0.0100	0.0272	0.0127	0.0399	0.0161
S-DPO	0.0167	0.0104	0.0239	0.0122	0.0392	0.0163	0.0115	0.0070	0.0198	0.0099	0.0310	0.0129	0.0229	0.0114	0.0334	0.0156	0.0468	0.0189
SPRec	0.0226	0.0141	0.0281	0.0143	0.0436	0.0181	0.0130	0.0079	0.0216	0.0108	0.0325	0.0135	**0.0328**	**0.0168**	0.0453	0.0211	0.0562	0.0227
R^2ec	0.0217	0.0135	0.0306	0.0156	0.0486	0.0202	0.0114	0.0069	0.0190	0.0095	0.0320	0.0133	0.0296	0.0147	0.0403	0.0188	0.0475	0.0192
ReRe	0.0215	0.0134	0.0318	0.0162	0.0484	0.0201	0.0140	0.0085	0.0224	0.0115	0.0315	0.0131	0.0300	0.0149	0.0442	0.0206	0.0548	0.0235
LatentR^3	0.0220	0.0137	0.0295	0.0155	0.0465	0.0195	0.0148	0.0090	0.0224	0.0112	0.0354	0.0147	0.0285	0.0142	0.0418	0.0198	0.0617	0.0249
RPORec	**0.0248***	**0.0155***	**0.0348***	**0.0178***	**0.0531***	**0.0220***	**0.0190***	**0.0101***	**0.0288***	**0.0131***	**0.0517***	**0.0185***	0.0326	0.0162	**0.0478***	**0.0223***	**0.0655***	**0.0275***
Improve (%)	9.73%	9.93%	9.43%	9.88%	9.26%	8.91%	28.4%	12.22%	28.57%	13.91%	46.05%	25.85%	-	-	5.52%	5.69%	6.16%	10.44%

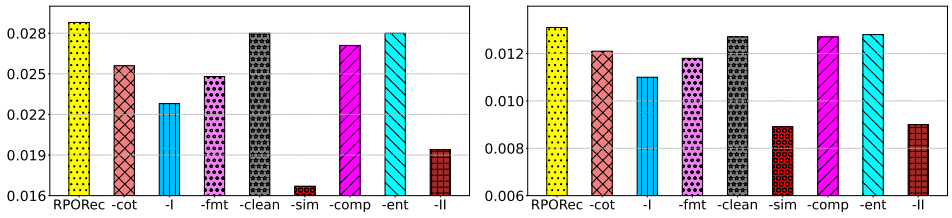
- Traditional models (GRU4Rec, Caser, SASRec) provide strong baselines, with SASRec performing best due to self-attention. Yet without explicit reasoning, they struggle to capture nuanced and evolving user preferences.
- Joint optimization approaches (TIGER, BIGRec, D^3) yield moderate gains, especially on sparse datasets CDs and Vinyl. This suggests that direct token-level optimization can distort reasoning, while TIGER's semantic ID generation fails to fully exploit LLM's pretrained reasoning knowledge.
- Fine-tuned generative and RL-based methods (S-DPO, SPRec, R^2ec, ReRe, LatentR^3) perform better overall, confirming the value of reasoning-aware modeling. However, ReRe and LatentR^3 may incur performance loss during aggregation, especially when candidate items are absent from training, while hidden-state coupling in R^2ec and latent-only optimization in LatentR^3 may make explicit reasoning harder to preserve. This trend is consistent with the design of RPORec, which preserves explicit CoTs while decoupling them from final retrieval through a specialized recommendation head.
- RPORec outperforms all baselines across datasets and most metrics, validating the combination of explicit CoT reasoning and a specialized recommendation head. By preserving reasoning in text and using iterative optimization with verifiable rewards, RPORec better aligns LLM reasoning with recommendation objectives and produces stronger, more interpretable recommendations.

3.3 Ablation Study (RQ2)

To assess the contribution of RPORec's components, we conduct an ablation study with the following variants on CDs and Vinyl dataset and report H@10 and N@10:

- **RPORec:** The complete model.
- **-cot:** Removes CoT inputs and reasoning-augmented modeling in the Rechead during Stage I.
- **-I:** Removes the entire Stage I training, and performs item retrieval directly based on the LLM backbone outputs.
- **-fmt / -clean / -sim / -comp / -ent:** Removes the corresponding reward term from Stage II.
- **-II:** Removes the entire Stage II, without fine-tuning the LLM backbone.

Based on the ablation results in Figure 2(a) and 2(b), we could draw the following conclusions:



(a) H@10. (b) N@10.

Figure 2: Ablation study.

- **Stage I components.** Removing CoT-aware modeling (**-cot**) causes a clear performance drop, showing that Stage I benefits substantially from explicit CoT utilization in Rechead rather than relying only on the backbone outputs.
- **Role of Rechead.** Removing Stage I and Rechead (**-I**) markedly degrades performance, confirming that free-form backbone outputs alone are insufficient for accurate retrieval and that Rechead is essential for bridging reasoning and recommendation.
- **Stage II rewards.** Removing any reward term in Stage II lowers performance, showing their complementarity. In particular, **-sim** causes the largest drop, indicating that similarity reward is crucial for keeping CoT optimization semantically grounded; without it, r_{comp} reward may still shorten reasoning, but is more likely to drive it away from valid recommendation-relevant content.
- **Role of Stage II.** Removing the entire Stage II (**-II**) sharply reduces performance, demonstrating that RL-based refinement is indispensable for recommendation alignment.

Moreover, we further evaluate RPORec with Llama3.2-1B [7] on CDs and Vinyl (Table 2) to test RPORec's generalization across different LLM backbones. Results demonstrate consistent performance across the two base models, confirming RPORec's robustness and generalizability.

Table 2: Backbone generalization analysis.

Model	H@10	N@10
RPORec (Qwen3-0.6B)	0.0288	0.0131
RPORec (Llama3.2-1B)	0.0294	0.0128

3.4 Case Study (RQ3)

To provide an intuitive analysis of RPORec 's effect on explicit reasoning, we conduct a case study on the CDs and Vinyl dataset. As the main reasoning enhancement in RPORec comes from the CoT Reward in Section 2.2.3, we compare the LLM backbone's reasoning content and average output length before and after applying this reward. Representative examples are shown in Figure 3. To further assess CoT quality, we use GPT-5.4 [21] as an LLM judge and evaluate generated CoTs from two perspectives: **Information Density**, which measures whether the reasoning is concise yet informative rather than redundant or noisy, and **Recommendation Utility**, which measures whether the reasoning provides evidence that is useful for identifying the correct recommended item; both are scored from 0 to 1. We also provide the averaged output length comparison in Figure 4.

Case study on reasoning CoT quality

Without CoT Reward: <think> Okay, let's see. The user wants me to analyze a brief CD or vinyl item and recommend next purchase. They provided several records with attributes. I need to recommend a new one and summarize it concisely. The example uses a title, average rating, number, and description. Let me check the data again. Looking at the records, the user might prefer something that combines a unique genre with a strong emotional impact. The Great American Music Galaxy album has a 4.5 average, 276 rating, and a description mentioning it's a collection of music from the band's past. The description also mentions the band's early days and the album's release in 2016. That sounds like a solid recommendation. It's a debut album with a focus on the band's history, which could be appealing. I should present this as the answer with those details. </think><answer>title: The Great American Music Galaxy; average rating: 4.7; rating number: 114; description: [2016 release, the debut album by this band, featuring the same musicians as The Band and The Doors. The album was released in 2016, which is a year after The Doors and The Band had released their own albums. This album is a collection of music from the past, including the band's early days.]</answer>

GPT-5.4 Judge Scores.

- **Case-level:** Information Density = 0.34, Recommendation Utility = 0.41
- **Test-set Avg.:** Information Density = 0.31, Recommendation Utility = 0.43

With CoT Reward: <think> The Great American Music Galaxy is a 2016 debut album by the band, featuring their early music and with strong emotional impact that the user may prefer. </think><answer>title: The Great American Music Galaxy; average rating: 4.6; rating number: 257; description: [2016 release, the debut album by this band, featuring the same musicians as The Band and The Doors. The album was released in 2016, which is a year after The Doors and The Band had released their own albums. This album is a collection of music from the past, including the band's early days.]</answer>

GPT-5.4 Judge Scores.

- **Case-level:** Information Density = 0.78, Recommendation Utility = 0.73
- **Test-set Avg.:** Information Density = 0.79, Recommendation Utility = 0.71

Figure 3: Case study on reasoning CoT quality.

From Figure 3, we observe that without CoT reward, the model tends to produce verbose reasoning that repeats item metadata (e.g., release year and background description) and introduces weak speculative statements about user preference, which dilutes recommendation-relevant evidence. After applying CoT reward, the reasoning becomes markedly shorter and more focused, retaining only the core cues that support the recommendation decision. This qualitative change is also reflected by the GPT-5.4 judge scores: for the illustrated case, **Information Density** improves from 0.34 to 0.78 and **Recommendation Utility** improves from 0.41 to 0.73; similar gains are also observed on the test-set averages ($0.31 \rightarrow 0.79$ and $0.43 \rightarrow 0.71$, respectively). Concurrently, as shown in Figure 4, the inference length decreases effectively with training, substantially reducing noise in the reasoning trajectory. This reduction not only improves inference quality but also enhances generation efficiency.

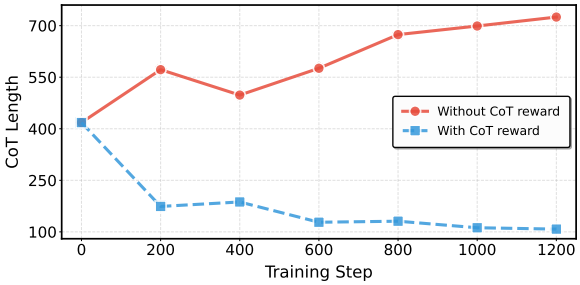


Figure 4: CoT length comparison.

4 Online Application

To validate the effectiveness of RPORec in real-world deployment, we integrated it into a large-scale industrial advertising system and conducted rigorous online A/B testing.

Figure 5 illustrates the online deployment architecture. Given the stringent constraints on computational cost, inference latency, and feature interaction complexity in production environments, we deploy the LLM backbone as a nearline user understanding module, while using task-specific online ranking models as Recheads for fine-grained modeling. Specifically, the RPORec LLM backbone analyzes user profile attributes and historical behavior to predict user interests. We then extract the CoT and answer segments from the generated responses and store them in a key-value (K-V) database. During online serving, the CoT tokens are embedded into dense vectors and incorporated as auxiliary user features into downstream ranking models (i.e., Recheads), together with other online features.

The A/B test ran for 7 days with 10% traffic allocation, covering approximately 40 million users and 2.1 billion ad impressions. The baseline is a state-of-the-art ranking model currently deployed

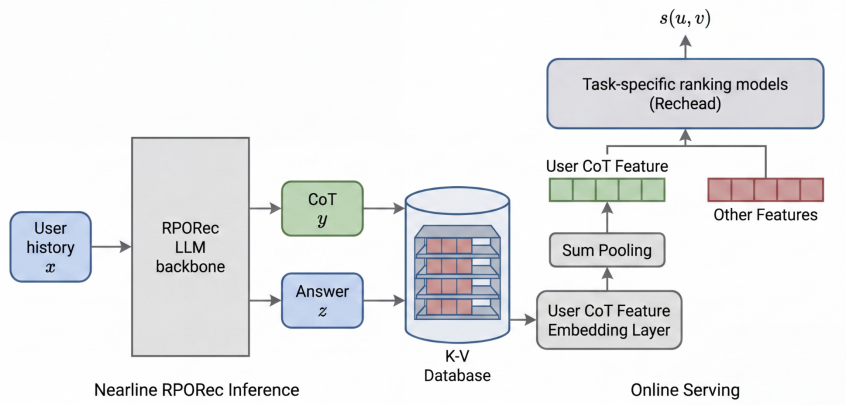


Figure 5: Online deployment architecture of RPORec.

in production, incorporating extensive handcrafted features and GSU-ESU modules [22], with 0.8 billion sparse and 0.2 billion dense parameters. Online applications demonstrate that integrating RPORec achieved a **1.348%** lift in Revenue and a **1.058%** increase in Advertiser Value (ADVV) [5].

5 Related Works

Recommender systems have evolved from collaborative filtering to deep learning [12, 28], and recently entered a new paradigm leveraging LLMs' world knowledge. TIGER [23] pioneers this shift through generative retrieval that autoregressively decodes Semantic IDs (SIDs) from collaborative signals, enabling LLMs to directly generate recommendations. Subsequent work enhances generation quality through debiasing [4] and preference alignment via DPO variants [6, 9] and RLVR-based optimization [27]. With RLVR, LLM-based methods have progressed toward reasoning-aware recommendations. R^2ec [33] employs a dual-head architecture jointly generating reasoning and predictions via hidden-state updates. LatentR^3 [35] optimizes compact latent reasoning in hidden space using modified GRPO without explicit text generation. ReRe [27] directly conducts recommendations through constraint decoding. However, applying reasoning-augmented LLMs poses format and task alignment challenges. Hidden-state methods suffer coarse-grained updates that inadequately refine explicit reasoning, while generative approaches face semantic gaps between LLM outputs (text or pretrained tokens) and recommendation objectives (discrete item IDs or SIDs). RPORec addresses these limitations through a decoupled two-stage framework with an LLM backbone for reasoning and a specialized Rechead for recommendations. This design enables RLVR to align explicit reasoning CoT at the text level, enhancing reasoning quality while effectively leveraging CoT for recommendation modeling and item retrieval in Rechead without semantic gap interference.

6 Conclusion

We introduce RPORec, a reinforcement learning framework that systematically addresses the challenge of integrating high-quality LLM reasoning into recommender systems through three methodological advances. First, our iterative optimization pipeline decouples reasoning generation from recommendation prediction, enabling joint refinement while preserving reasoning integrity and ensuring task-specific alignment. Second, our reasoning-augmented Rechead effectively exploits CoT knowledge through lightweight transformers and adaptive gating mechanisms, filtering noisy content while distilling informative signals for enhanced recommendation precision. Third, we propose advanced reasoning refinement and alignment through RLVR-based optimization, incorporating format adherence, accuracy signals, and CoT quality improvements to produce concise, task-relevant reasoning trajectories for improved performance. Extensive experiments demonstrate that RPORec significantly outperforms state-of-the-art baselines across public benchmarks, while deployment in a large-scale industrial system achieved 1.348% Revenue lift and 1.058% Advertiser Value increase, establishing its effectiveness in delivering superior performance, interpretability, and practical applicability for reasoning-augmented recommendations.

References

[1] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. Llm based generation of item-description for recommendation system. In *Proceedings of the 17th ACM conference on recommender systems*. 1204–1207.

[2] Honghui Bao, Wenjie Wang, Xinyu Lin, Fengbin Zhu, Teng Sun, Fuli Feng, and Tat-Seng Chua. 2025. Heterogeneous user modeling for llm-based recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 145–154.

[3] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems* 3, 4 (2025), 1–27.

[4] Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. 2024. Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation. *arXiv preprint arXiv:2406.14900* (2024).

[5] Zheng Chai, Qin Ren, Xijun Xiao, Huizhi Yang, Bo Han, Sijun Zhang, Di Chen, Hui Lu, Wenlin Zhao, Lele Yu, et al. 2025. Longer: Scaling up long sequence modeling in industrial recommenders. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 247–256.

[6] 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. *Advances in Neural Information Processing Systems* 37 (2024), 27463–27489.

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints* (2024), arXiv–2407.

[8] Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678* (2025).

[9] Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. 2025. Sprec: Self-play to debias llm-based recommendation. In *Proceedings of the ACM on Web Conference 2025*. 5075–5084.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* 645, 8081 (2025), 633–638.

[11] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient Natural Language Response Suggestion for Smart Reply. arXiv:1705.00652 [cs.CL]

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

[13] Enyi Jiang, Changming Xu, Nischay Singh, and Gagandeep Singh. 2025. Misaligning Reasoning with Answers–A Framework for Assessing LLM CoT Robustness. *arXiv preprint arXiv:2505.17406* (2025).

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

[15] PN Vijaya Kumar and V Raghunatha Reddy. 2014. A survey on recommender systems (RSS) and its applications. *International Journal of Innovative Research in Computer and Communication Engineering* 2, 8 (2014), 5254–5260.

[16] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2024. Matryoshka Representation Learning. arXiv:2205.13147 [cs.LG]

[17] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender system application developments: a survey. *Decision Support Systems* 74 (2015), 12–32.

[18] Haokai Ma, Ruobing Xie, Lei Meng, Fuli Feng, Xiaoyu Du, Xingwu Sun, Zhanhui Kang, and Xiangxu Meng. 2024. Negative sampling in recommendation: A survey and future directions. *arXiv preprint arXiv:2409.07237* (2024).

[19] Anqi Mao, Mehryar Mohri, and Yutao Zhong. 2023. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*. pmlr, 23803–23828.

[20] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

[21] OpenAI. 2026. GPT-5.4. `https://openai.com`

[22] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2685–2692.

[23] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[24] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. `https://arxiv.org/abs/1908.10084`

[25] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[26] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon. com. *Ieee internet computing* 21, 3 (2017), 12–18.

[27] Junfei Tan, Yuxin Chen, An Zhang, Junguang Jiang, Bin Liu, Ziru Xu, Han Zhu, Jian Xu, Bo Zheng, and Xiang Wang. 2025. Reinforced Preference Optimization for Recommendation. *arXiv preprint arXiv:2510.12211* (2025).

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

[29] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] `https://arxiv.org/abs/2505.09388`

[30] Boshi Wang, Xiang Yue, and Huan Sun. 2023. Can ChatGPT defend its belief in truth? evaluating LLM reasoning via debate. *arXiv preprint arXiv:2305.13160* (2023).

[31] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939* (2025).

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

[33] Runyang You, Yongqi Li, Xinyu Lin, Xin Zhang, Wenjie Wang, Wenjie Li, and Liqiang Nie. 2025. R2ec: Towards Large Recommender Models with Reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[34] Jiaqi Zhang, Junliang Yu, Zongwei Wang, Wei Yuan, Tong Chen, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. 2025. Towards Reasoning-Aware Recommender Systems: A Survey in the LLM Era. *Authorea Preprints* (2025).

[35] Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2025. Reinforced Latent Reasoning for LLM-based Recommendation. *arXiv preprint arXiv:2505.19092* (2025).

[36] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023).

[37] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2019. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter* Spring (2019), 1–15.

[38] 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*. 1796–1806.

A Preliminary

This section introduces the two foundations of our study: Group Relative Preference Optimization (GRPO) [25], a representative method in Reinforcement Learning with Verifiable Rewards (RLVR), and general LLM-based recommendation task formulation, which reformulates recommendations as a language modeling task.

A.1 Group Relative Preference Optimization (GRPO)

Group Relative Preference Optimization (GRPO) [25] is an efficient algorithm for aligning LLMs under verifiable reward supervision. For a policy π_θ that generates textual output o given an input x, GRPO samples G candidate responses $\{o_i\}_{i=1}^{G}$ and obtains a scalar reward r_i for each. The reward advantage is normalized within the group G:

$$\hat{A}_i = \frac{r_i - \mu_G}{\sigma_G}, \tag{8}$$

where μ_G and σ_G are the mean and standard deviation of G rewards. At the token level, the ratio between the trainable policy π_θ and a fixed reference model π_{ref} is computed as

$$w_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}{\pi_{\text{ref}}(o_{i,t} \mid x, o_{i,<t})}, \tag{9}$$

where $o_{i,t}$ is the t-th token of o_i. The optimization objective is then

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) =& \mathbb{E}[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min(w_{i,t}(\theta)\hat{A}_i, \\ & \text{clip}(w_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i) - \beta\,\mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]],\end{aligned} \tag{10}$$

where $\mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]$ is the estimated KL divergence between π_θ and π_{ref}, and ε is the clipping ratio. GRPO thus adjusts token probabilities using relative in-group feedback, providing trajectory sampling-based preference alignment, achieving stable policy improvement without explicit value estimation.

Recent work extends GRPO from an averaged token-level perspective by introducing entropy-guided optimization [31]. This approach observes that only a minority of tokens that exhibit high uncertainty are critical "decision tokens" influencing reasoning outcomes. The token entropy is therefore defined as

$$E_t = -\sum_{\hat{o}_t \in vocab} \pi_\theta(\hat{o}_t \mid x, o_{<t}) \log \pi_\theta(\hat{o}_t \mid x, o_{<t}), \tag{11}$$

which measures uncertainty in token selection at token position t with token probabilities on the whole vocabulary ($vocab$). To enhance policy learning efficiency, the reward updates are restricted to the subset of tokens with the highest entropy. Let $\mathcal{S} = \{t \mid E_t \geq \tau_\rho^{\mathcal{B}}\}$ denote the indices of tokens exceeding the entropy threshold $\tau_\rho^{\mathcal{B}}$, which is the top ρ threshold within (mini-)batch $\mathcal{B}$. The updates are thus computed only for the tokens in $\mathcal{S}$ rather than across all token positions. This selective gradient approximation effectively focuses learning on the most informative portions of reasoning trajectories, reducing computational cost and noise from low-entropy tokens. Empirically, such selective reinforcement—optimized with top $\rho = 20\%$ of tokens–achieves comparable alignment performance while improving stability and interpretability. The entropy-guided refinement thus deepens theoretical understanding of GRPO and suggests scalable strategies for reinforcement learning in large language models.

A.2 LLM-based Recommendations

LLM-based recommendation [38, 1] leverages LLMs to perform recommendations through natural-language understanding and generation. Given a user's interaction history $H_u = [v_1, \ldots, v_J]$, each item v_j is described textually, and the history is composed into a prompt x_u and is encoded with the LLM's last hidden state into $\boldsymbol{h}_u$. For a candidate item v, similar representation $\boldsymbol{h}_v$ is obtained from encoding item descriptions x_v, and their matching score can be computed as

$$s(u, v) = \boldsymbol{h}_u^\top \boldsymbol{h}_v. \tag{12}$$

Contrastive learning is then applied to separate positive from negative items during training [9, 33].

$$\mathcal{L}_{\mathrm{CL}} = -\log \frac{\exp\left(s(u, v^+)/\tau\right)}{\sum_{v' \in B_{v^+}} \exp\left(s(u, v')/\tau\right)}, \tag{13}$$

where τ is a temperature parameter, B_{u,v^+} denotes the union of positive and negative samples for the interaction (u, v^+), and v^+ is the ground-truth positive item. When $\tau = 1$, this formulation reduces to the standard Cross-Entropy (CE) loss. However, directly optimizing LLM hidden states with task-specific objectives may disrupt the model's inherent reasoning structures, resulting in degraded reasoning capability and suboptimal recommendation performance.

Another paradigm, generative variants [27], instead let the model autoregressively produce the next preferred item, modeling

$$P(o \mid x_u) = \prod_{t=1}^{T} P(o_t \mid x_u, o_{<t}), \tag{14}$$

where the generated text represents the reasoning trace, predicted item text, or item SIDs. This formulation unifies recommendation with natural-language generation, enabling reasoning-aware and interpretable preference modeling based on the power of LLMs. However, a mismatch between natural language expressions and discrete item identifiers forces a subsequent approximate search or constraint decoding that compromises accuracy. Even with SIDs for generative recommendation, semantic gaps persist due to differences in tokenization and training between the LLM's native vocabulary and the SID space used in recommendation.

Algorithm 1 Iterative optimization algorithm of RPORec

Input: A training dataset $\mathcal{D} = \{(x_j, v_j^+)\}_{j=1}^{|\mathcal{D}|}$, where x_j is the user history and v_j^+ is the ground-truth next item. A pre-trained LLM backbone $\mathbf{\Theta}_{LLM}$ and an initialized Rechead $\mathbf{\Theta}_R$.
Output: Optimized RPORec model with $(\mathbf{\Theta}^*_{LLM}, \mathbf{\Theta}^*_R)$.

Stage I – Reasoning-Augmented Recommendation Modeling:

Generate fixed y_j and z_j for each (x_j, v_j^+) in $\mathcal{D}$ with $\mathbf{\Theta}_{LLM}$ once.
for epoch $= 1$ **to** max_epochs **do**
  for each mini-batch $B \subset \mathcal{D}$ **do**
    Forward the $(x_j, y_j, z_j) \in B$ through Rechead via Equation (1), (2), (3), (4), (5).
    Sample k negative items N_{uv}^k.
    Update $\mathbf{\Theta}_R$ via Equation (6).
  end for
  if Rechead converged **then**
    Obtain and freeze $\mathbf{\Theta}^*_R$, and proceed to Stage II.
    break
  end if
end for

Stage II: Advanced Reasoning Refinement and Alignment:

for epoch $= 1$ **to** max_epochs_RL **do**
  for each mini-batch $B \subset \mathcal{D}$ **do**
    Generate y_j and z_j for each (x_j, v_j^+) with $\mathbf{\Theta}_{LLM}$.
    Forward $(x_j, y_j, z_j) \in B$ through Rechead to obtain $s(u, v)$ in Equation (5).
    Compute r_{fmt}, r_{clean}, r_{ndcg}, r_{sim}, r_{comp}, and r_{ent} as defined in Section 2.2.
    Aggregate to r via Equation (7).
    Update $\mathbf{\Theta}_{LLM}$ via GRPO and [31] to maximize r.
  end for
  if LLM backbone converged **then**
    Obtain $\mathbf{\Theta}^*_{LLM}$
    return $(\mathbf{\Theta}^*_{LLM}, \mathbf{\Theta}^*_R)$
  end if
end for

B Overall Algorithm of RPORec

Here we provide the overall iterative optimization process of RPORec in Algorithm 1. Specifically, we first freeze the LLM backbone and train Rechead for the downstream recommendation task in Stage I. We then freeze Rechead and use it to provide stable, task-specific feedback for LLM reasoning refinement and alignment in Stage II.

C Detailed Experimental Setup

C.1 Dataset

Following previous works [33, 27], we conduct experiments on three datasets from the latest public Amazon source[3], namely Musical Instruments, CDs and Vinyl, and Video Games. Following the temporal-truncation protocol in prior work [3, 35, 33], we start from interactions in the most recent year to collect at least 10k valid items for each dataset. We omit the 5-core filter to preserve natural recommendation behavior. Each user's interaction history is chronologically ordered and truncated to the latest 20 actions. We then split each dataset into training/validation/test sets with a ratio of 8:1:1 and evaluate on the entire item set. The processed dataset statistics are summarized in Table 3.

Table 3: Data statistics.

Dataset	Users	Items	Interactions
Musical Instruments	15,656	10,320	34,373
CDs and Vinyl	7,701	12,024	13,435
Video Games	29,230	10,144	63,502

C.2 Baselines

Here we briefly introduce the baselines used in Section 3.

- **GRU4Rec** [12]: RNN-based session recommender with ranking-oriented loss for sequential user behavior modeling.
- **Caser** [28]: CNN-based model that captures union-level sequential patterns via horizontal and vertical filters.
- **SASRec** [14]: Self-attention model for adaptively selecting relevant items from user histories.
- **TIGER** [23]: Generative retrieval using semantic IDs and Transformer-based sequence prediction.
- **BIGRec** [3]: LLM-based generative recommender fine-tuned to output tokenized item representations.
- **D^3** [4]: Debiases LLM decoding by controlling ghost tokens and improving output diversity.
- **S-DPO** [6]: Softmax-enhanced DPO for ranking, leveraging multiple hard negatives.
- **SPRec** [9]: Self-play framework for DPO training to reduce bias and increase diversity.
- **R^{2}ec** [33]: Unified reasoning-augmented recommender with dual-head design.
- **ReRe** [27]: RL-based generative recommender with verifiable rewards and constrained beam search.
- **LatentR^3** [35]: RL-optimized latent reasoning without explicit chains for efficient LLM-based recommendation.

C.3 Implementation Details

We train RPORec on 4 GPUs. To ensure fair comparison, we set the batch size to 256 for non-LLM methods and 8 for LLM-based methods. All LLM-based methods, including RPORec, use Qwen3-0.6B [29] as the backbone model for efficient generation. The maximum generation length of all LLMs is set to 768 for efficiency. We use static-retrieval-mrl-en-v1 [24, 16, 11] as the sentence

[3] https://amazon-reviews-2023.github.io/index.html

transformer encoder within the Rechead module. For each model, the model-specific hyperparameters are tuned via grid search, while shared hyperparameters remain fixed across experiments. We report the average performance over 10 runs using the optimal hyperparameter configuration.

D Limitations

Although RPORec is effective, there remains room to further improve the quality of the generated reasoning CoTs. For example, introducing additional optimization signals, such as diversity-oriented metrics and hallucination mitigation strategies, may further enhance reasoning quality and potentially yield additional recommendation gains. We consider these directions promising avenues for future work.