REASONING TO RANK: EXPLOITING LARGE LANGUAGE MODELS FOR RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recommender systems are tasked to infer users' evolving preferences and rank items aligned with their intents, which calls for in-depth reasoning beyond pattern-based scoring. Recent efforts start to leverage large language models (LLMs) for recommendation, but how to effectively optimize the model for improved recommendation utility is still under explored. In this work, we propose Reasoning to Rank, an end-to-end training framework that internalizes recommendation utility optimization into the learning of step-by-step reasoning in LLMs. To avoid position bias in LLM reasoning and enable direct optimization of the reasoning process, our framework performs reasoning at the user-item level and employs reinforcement learning for end-to-end training of the LLM. Experiments on three Amazon datasets and a large-scale industrial dataset showed consistent gains over strong conventional and LLM-based solutions. Extensive in-depth analyses validate the necessity of the key components in the proposed framework and shed lights on the future developments of this line of work.

1 INTRODUCTION

Recommender systems have established themselves as an essential interface for users to navigate the overwhelming scale of modern digital ecosystems, such as e-commerce, social media, online education and many more (Ricci et al., 2022; Adomavicius & Tuzhilin, 2005). However, conceptualizing recommendation merely as a mechanical ranking task underestimates the complexity of the challenge. At its core, recommendation is an inferential decision-making process in which a system integrates evidence from a user's interaction history to estimate latent preferences and current needs (Ricci et al., 2022). Classical matrix factorization captures latent user–item factors (Koren et al., 2009), while sequential recommenders explicitly exploit ordered interaction histories (Kang & McAuley, 2018; Sun et al., 2019). The contrast between statistical association and deliberative inference echoes the distinction between fast and slow modes of judgment (Kahneman, 2011), motivating us to explore whether explicit reasoning can complement preference modeling. Large Language Models (LLMs) combine broad pretrained knowledge and few-shot language capabilities (Brown et al., 2020) with the ability to expose multi-step inference through chain-of-thought prompting (Wei et al., 2022). These capabilities have motivated a growing body of LLM-based recommendation research (Wu et al., 2024; Zhang et al., 2025; Wei et al., 2024; Geng et al., 2022; Bao et al., 2023; Lin et al., 2024; Gao et al., 2025; Kim et al., 2025; You et al., 2025).

In recognition of the potential, the research community has rapidly adopted LLMs to innovate recommendation methodologies (Wu et al., 2024; Zhang et al., 2025; Wei et al., 2024; Geng et al., 2022; Bao et al., 2023; Lin et al., 2024; Gao et al., 2025; Kim et al., 2025; You et al., 2025). One line uses LLMs as auxiliary data or feature augmenters that complement behavior-based recommendation signals (Wei et al., 2024). A complementary line makes the language model itself the recommendation model, either predicting relevance decisions or generating item identifiers and sequences (Geng et al., 2022; Bao et al., 2023; Zhang et al., 2025; Liu et al., 2024a; Lin et al., 2024). Recent studies further explore review-grounded preference reasoning, explicit reasoning traces, and process-supervised optimization to better align LLM recommendation with ranking behavior (Kim et al., 2025; You et al., 2025; Gao et al., 2025).

Despite these advancements, fully exploiting LLMs' power and, especially, realizing the paradigm of reasoning for recommendation into practical effectiveness is still hindered by distinct structural

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

obstacles (Liu et al., 2024b; Turpin et al., 2023). First, LLMs are notorious victims of position bias, where the order of input disproportionately influences the model's output. This sensitivity breaks the permutation invariance required for robust item ranking, thereby distorting the assessment of true user preference in recommendation (Bito et al., 2025). Second, many approaches rely on carefully crafted prompts to guide LLMs to generate linguistically plausible reasoning traces but lack a principled mechanism that connects reasoning with the utility of the resulting recommendation.

In this work, we propose Reasoning to Rank (R2Rank), a framework that unifies Chain-of-Thought (CoT) reasoning with ranking optimization through an end-to-end reinforcement learning solution (Wei et al., 2022; Schulman et al., 2017). R2Rank decouples the recommendation process into independent pointwise inferences: for each candidate item, the model first generates a rationale grounded in the user's interaction history and item information, and then projects this reasoning into a scalar relevance score – a design that intrinsically mitigates position bias in LLMs. To connect semantic reasoning with the optimization of ranking utility, we employ a Plackett-Luce probabilistic surrogate to map these scores into a differentiable ranking distribution (Plackett, 1975). This yields candidate-specific score gradients that back-propagate through the reasoning hidden states into the LLM, while a shared listwise return trains the sampled reasoning tokens through PPO. Furthermore, to address the scarcity of explicit reasoning in standard recommendation datasets, we introduce a cold-start Supervised Fine-Tuning (SFT) stage using teacher-generated rationales. Its self-check is inspired by iterative self-feedback and refinement (Madaan et al., 2023), while our supervision and filtering procedure is task-specific. This holistic design ensures that the model's reasoning is not merely descriptive, but actively optimized to drive the recommendation performance in a permutation-invariant manner.

Viewed in this way, recommendation is a reasoning process that identifies preference evidence, relates it to a candidate, and turns it into a ranking decision. Our contribution is to couple candidate-level evidence extraction with list-level optimization: pointwise inference removes arbitrary candidate presentation order from each assessment, while the scores jointly define a distribution over ranked prefixes at the target cutoff. The realized NDCG of a sampled prefix supplies a shared reward for jointly training the scoring head and the rationale policy, rather than separate per-item rewards. SFT, Plackett–Luce, and PPO serve as implementation mechanisms for this coupling; the novelty lies in optimizing the reasoning process itself for cutoff-level ranking utility.

We conducted extensive experiments on three Amazon datasets and a large-scale industrial advertising dataset to compare R2Rank with conventional recommenders (Kang & McAuley, 2018; Sun et al., 2019; He et al., 2020) and competitive language-model baselines (Guo et al., 2025; Zhang & Wang, 2023; Bao et al., 2023; You et al., 2025). The results show that R2Rank achieves the best or near-best performance across datasets on NDCG metrics, validating the effectiveness of optimizing LLM reasoning for listwise recommendation utility. We further perform ablation studies to verify the contributions of key components in R2Rank, and conduct in-depth analyses on cross-domain generalization, robustness across item-frequency strata, and sensitivity to user history length, demonstrating that R2Rank remains effective and stable across diverse settings.

2 Related work

We organize our discussion of related work into two parts: the recent efforts that leverage LLMs for recommendation and the developments of LLM post-training.

• **Large Language Models in Recommender Systems** LLMs have recently been introduced into recommender systems as a flexible substrate for semantic understanding, instruction following, and multi-step reasoning, enabling recommendation to be formulated and solved through natural-language interfaces (Wu et al., 2024). Existing studies can be summarized into several paradigms. Some treat LLMs as prompt-based recommenders that directly select or rank candidates from a user context; zero- and few-shot prompting can underperform conventional recommendation methods (Liu et al., 2023), while long contexts introduce a separate sensitivity to where relevant information appears (Liu et al., 2024b). Others use LLMs as data or feature augmenters to enrich recommendation signals before downstream graph-based modeling (Wei et al., 2024).

A complementary line trains or prompts the language model itself to produce recommendation decisions, either via generative identifier/sequence prediction or via scoring and ranking over can-

didates (Geng et al., 2022; Bao et al., 2023; Zhang et al., 2025; Liu et al., 2024a; Lin et al., 2024; 2025). In parallel, an emerging line uses explicit reasoning, review-grounded preference inference, or process-level supervision to better connect model behavior with recommendation quality (Kim et al., 2025; Gao et al., 2025; You et al., 2025). While these results suggest that reasoning can strengthen preference inference, broadly applicable objectives that consistently translate reasoning improvements into listwise ranking gains remain underexplored.

• **Post-training for Large Language Models** Post-training has become a standard recipe for adapting pretrained LLMs, typically consisting of supervised fine-tuning (SFT) and preference-based optimization. Representative approaches include policy-gradient RL methods (e.g., PPO) and direct preference optimization methods (e.g., DPO), as well as more recent group-based RL variants (e.g., GRPO) (Ouyang et al., 2022; Schulman et al., 2017; Rafailov et al., 2023; Guo et al., 2025). These methods primarily align LLMs at the response level, optimizing how a single generated output matches a preference signal. In contrast, our post-training follows a learning-to-rank perspective: the optimization target is inherently listwise, aiming to align model behavior with ranking quality rather than solely improving standalone responses.

3 METHODOLOGY

We present the recommendation setting, the R2Rank framework, and its training through reinforcement learning and structured SFT.

3.1 PRELIMINARIES

For each user $u \in \mathcal{U}$, we represent the user's observed interaction history as a chronologically ordered sequence of items, $H_u = \langle v_1, v_2, \ldots, v_L \rangle$ with $v_\ell \in \mathcal{I}$, where each interacted item v_ℓ is depicted with descriptive signals, such as timestamp, textual item description and other forms of meta data. When available, we denote the basic information (e.g., profile attributes) of a user as b_u. The recommendation context for user u is then defined as $c_u = (b_u, H_u)$. Given a set of recommendation candidates $X_u = \{x_1, \ldots, x_K\} \subset \mathcal{I}$ for user u, a model (denoted as π_θ) outputs a ranking τ_u over X_u to maximize the recommendation utility $\rho(\tau_u)$.

In this paper, we focus on Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002) as the utility metric $\rho(\cdot)$, but the training framework can be easily adopted to all other ranking-based metrics. Specifically, denote the user-provided relevance feedback on τ_u as $R_u = \{r_1, \ldots, r_K\}$, where r_k is a graded relevance label for item x_k (e.g., 0/1 for binary feedback), NDCG is computed as,

$$\text{NDCG}(\tau_u) = \frac{\text{DCG}(\tau_u)}{\text{DCG}(\tau_u^*)}$$

where $\text{DCG}(\tau) = \sum_{k=1}^{K} \frac{2^{r_{\tau(k)}}-1}{\log_2(k+1)}$ and τ_u^* stands for the ideal ranking of X_u, i.e., items ranked according to R_u.

3.2 FRAMEWORK OVERVIEW

Generating a ranked list directly from c_u and X_u exposes predictions to irrelevant candidate order (Bito et al., 2025). Longer histories and candidate lists also make relevant evidence harder to use (Liu et al., 2024b), while the cost of full self-attention grows quadratically with sequence length (Vaswani et al., 2017).

As illustrated in Figure 1, R2Rank separates candidate-level preference reasoning from ranking aggregation. For each item $x_i \in X_u$, the LLM produces structured reasoning y_i conditioned on c_u,

$$y_i \sim \pi_\theta(\cdot \mid c_u, x_i), \qquad i = 1, \ldots, K. \tag{1}$$

The rationale is trained to connect evidence in c_u with the candidate's attributes rather than rely on item-agonistic shortcuts.

Let $h_k = h_\theta(c_u, x_k, y_k)$ be the last-token hidden state of y_k. We map it to a relevance score and rank by

$$\tau_u \;=\; \text{argsort}\big(\{s_k = f_\phi(h_k)\}_{k=1}^{K}\big), \tag{2}$$

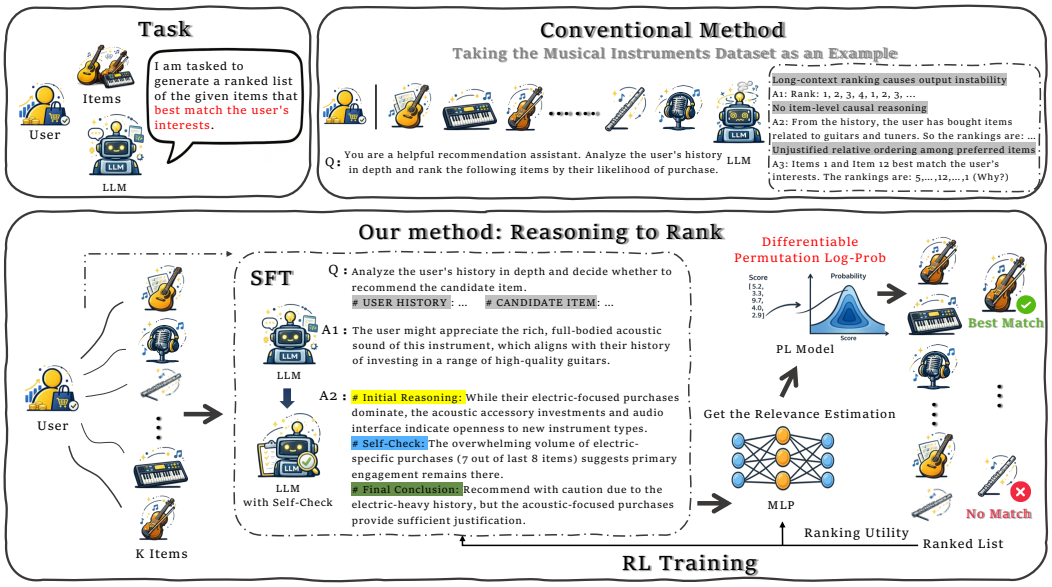


Figure 1: Overview of R2Rank. Candidate-wise rationales produce relevance scores, and listwise reinforcement learning optimizes the resulting ranking.

where f_ϕ is attached to the final layer of π_θ. Scores depend on both θ and ϕ: the ranking loss passes through f_ϕ and the non-detached h_θ into the backbone. Independent assessments remove candidate presentation order from each prompt and keep the per-candidate context bounded. They require K forward passes, which can be batched and parallelized.

3.3 Reinforcement Learning under Plackett-Luce Surrogate

Once τ_u is presented to the user, user feedback R_u can be collected to measure the utility of τ_u. The deterministic inference pipeline in R2Rank is summarized below,

$$(c_u, X_u) \xrightarrow{\pi_\theta} \{y_i, h_i\}_{i=1}^K \xrightarrow{f_\phi} \{s_i\}_{i=1}^K \xrightarrow{\text{argsort}(\mathbf{s})} \tau_u \xrightarrow{R_u} \rho(\tau_u). \tag{3}$$

The hard sorting operation and ranking metrics such as NDCG are non-differentiable. During training, we therefore replace sorting with a Plackett–Luce (PL) distribution over ranked prefixes (Plackett, 1975). Monte Carlo NDCG supplies the return, while differentiable PL log-probabilities and the policy-gradient estimator (Sutton et al., 1999) provide the score and token-policy gradients described below.

• **Plackett-Luce Surrogate.** Given $\boldsymbol{s} = \{s_k\}_{k=1}^K$, let $M = K_{\text{eval}} = \min(K, 10)$ and let $\tau_{1:M}$ denote an ordered prefix of M distinct candidates. The Plackett–Luce (PL) model assigns this prefix the probability

$$P_M(\tau_{1:M}|\boldsymbol{s}) = \prod_{m=1}^{M} \frac{\exp\left(s_{\tau(m)}\right)}{\sum_{j \notin \{\tau(1),\ldots,\tau(m-1)\}} \exp(s_j)}. \tag{4}$$

At training time, we sample $\tau_{1:M}$ from P_M and evaluate its NDCG@M; items not selected into the prefix remain in the denominator until prefix sampling ends. At inference, we rank all candidates deterministically by descending score. The discrete sample is handled with the log-derivative estimator rather than differentiated through directly.

• **End-to-End Reinforcement Learning.** Let $Y = (y_1, \ldots, y_K)$ and $\pi_\theta(Y) = \prod_{k=1}^K \pi_\theta(y_k \mid c_u, x_k)$. Because $s_k = f_\phi(h_\theta(c_u, x_k, y_k))$, the joint training distribution factorizes as

$$q_{\theta,\phi}(Y, \tau_{1:M}) = \pi_\theta(Y) P_M(\tau_{1:M} \mid \boldsymbol{s}_{\theta,\phi}(Y)). \tag{5}$$

Algorithm 1 RL training for R2Rank

Require: Minibatch $\mathcal{B}$; current policy π_θ; frozen rollout policy $\pi_{\theta_{\text{old}}}$; scoring head f_ϕ; clipping ratio ϵ.

for each instance $(u, c_u, X_u = \{x_1, \ldots, x_K\}) \in \mathcal{B}$ **do**
  for $k = 1$ to K **do**
    Sample $y_k \sim \pi_{\theta_{\text{old}}}(\cdot \mid c_u, x_k)$; cache its old-policy token log-probabilities
    Teacher-force fixed y_k under π_θ to obtain current token log-probabilities and h_k; set $s_k \leftarrow f_\phi(h_k)$
  end for
  Sample $\tau_{1:M} \sim P_M(\cdot|\boldsymbol{s})$ using Equation 4
  Evaluate $\rho(\tau_{1:M}) \leftarrow \text{NDCG@}M(\tau_{1:M})$
  Accumulate token-policy objective $\mathcal{J}_{\text{PPO}}(\theta)$ using Equation 7
  Accumulate score-path objective $\mathcal{J}_{\text{PL}}(\theta, \phi) \leftarrow \rho(\tau_{1:M}) \log P_M(\tau_{1:M}|\boldsymbol{s})$
end for
Update θ by ascending $\mathcal{J}_{\text{PPO}}(\theta) + \mathcal{J}_{\text{PL}}(\theta, \phi)$
Update ϕ by ascending $\mathcal{J}_{\text{PL}}(\theta, \phi)$

Its expected-utility gradients decompose into the token-policy path and the hidden-state score path:

$$\begin{aligned}\nabla_\theta \mathbb{E}_q[\rho] &= \mathbb{E}_q[\rho \nabla_\theta \log \pi_\theta(Y) + \rho \nabla_\theta \log P_M(\tau_{1:M} \mid \boldsymbol{s}_{\theta,\phi}(Y))]\,, \\ \nabla_\phi \mathbb{E}_q[\rho] &= \mathbb{E}_q[\rho \nabla_\phi \log P_M(\tau_{1:M} \mid \boldsymbol{s}_{\theta,\phi}(Y))]\,. \end{aligned} \tag{6}$$

The sampled tokens Y, sampled prefix $\tau_{1:M}$, and realized reward ρ are treated as stop-gradient quantities in these estimators. During optimization, the trainer re-evaluates the fixed sampled tokens to obtain h_θ with an attached computation graph. Thus, the second term updates ϕ and back-propagates through h_θ to update θ; it is distinct from, rather than a duplicate of, the first score-function term (Sutton et al., 1999).

For policy-based optimization of the language model π_θ, we replace the first term in Equation 6 by a clipped PPO surrogate. Denote the token sequence generated for candidate x_k as $y_k = (o_{k,1}, \ldots, o_{k,T_k})$. Using the language of Markov Decision Process, the state at position t is the tuple $(c_u, x_k, o_{k,<t})$ and the action of policy π_θ is the choice of token $o_{k,t}$. However, the corresponding reward is delayed until the final ranking is assessed by the user. For the complementary token-generation path, we use the clipped Proximal Policy Optimization (PPO) surrogate (Schulman et al., 2017) to constrain the magnitude of policy updates.

Specifically, we use the following clipped objective with a zero baseline, i.e., the realized Monte Carlo return $\rho(\tau_{1:M})$ serves directly as the advantage estimate:

$$\mathcal{J}_{\text{PPO}}(\theta) = \sum_{k=1}^{K} \sum_{t=1}^{|y_k|} \min\Big(\gamma_{k,t}(\theta)\rho(\tau_{1:M}),\ \text{clip}(\gamma_{k,t}(\theta), 1-\epsilon, 1+\epsilon)\rho(\tau_{1:M})\Big), \tag{7}$$

where ϵ is the clipping threshold and $\gamma_{k,t}(\theta)$ is the token-level probability ratio,

$$\gamma_{k,t}(\theta) = \frac{\pi_\theta(o_{k,t} \mid c_u, x_k, o_{k,<t})}{\pi_{\theta_{\text{old}}}(o_{k,t} \mid c_u, x_k, o_{k,<t})},$$

in which θ_{old} denotes the frozen policy snapshot from which the reasoning content $\{y_k\}_{k=1}^K$ is sampled. Its cached token log-probabilities are treated as constants: PPO gradients pass only through the numerator π_θ, whereas PL gradients pass through $s_k = f_\phi(h_\theta)$ to both ϕ and θ. The rollout snapshot remains fixed for the collected batch and is refreshed only when the rollout actor is synchronized.

The LLM therefore receives two complementary gradients: the PPO term follows the probability of the sampled reasoning tokens, while the PL term follows the candidate scores through their last-token hidden states. The scoring head receives the PL term only.

3.4 Self-reflective SFT initialization

We warm-start the reasoning policy with a lightweight SFT stage to establish a consistent output structure before listwise RL. The template has three parts: an initial rationale, a self-check that

revisits the evidence in c_u and identifies potential issues, and a concise conclusion ending with a machine-readable `Recommend`/`Not Recommend` decision.

For user–item instances from a user-disjoint auxiliary subset, we query DeepSeek-R1 (Guo et al., 2025) with this template and retain responses whose format and final decision agree with the required schema and observed label. These responses provide the SFT supervision. This stage initializes the reasoning policy; the ranking objective is optimized in the subsequent RL stage. Data construction and filtering details are given in Appendix A, and the full template and design rationale in Appendix E.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

4.1.1 DATASETS AND METRICS

We evaluate on three Amazon datasets (Hou et al., 2026)—*Musical Instruments*, *Movies & TV*, and *Video Games*—and an industrial dataset from a large advertising platform (Table 1). We use user-disjoint train/validation/test splits of 8:1:1 and fixed candidate sets of $K{=}20$ (one observed positive and 19 negatives), reporting NDCG@$\{1, 5, 10\}$. All methods share the same splits and candidate sets. Preprocessing, SFT data accounting, negative sampling, and significance-testing details are provided in Appendix A.

Table 1: Statistics of datasets used in the experiments.

Dataset	#Users	#Items	#Interactions
Musical Instruments	13,031	10,326	49,171
Movies & TV	19,606	30,809	83,047
Video Games	12,417	7,457	42,305
Industrial	99,890	1,321,290	2,051,891

4.1.2 BASELINES

We compare with (i) conventional recommenders: **SASRec** (Kang & McAuley, 2018), **BERT4Rec** (Sun et al., 2019), and **LightGCN** (He et al., 2020); (ii) prompted or instruction-tuned language-model recommenders: **DeepSeek-R1** (Guo et al., 2025), **Prompt4NR** (Zhang & Wang, 2023), **RecRanker** (Luo et al., 2025), and **TALLRec** (Bao et al., 2023); and (iii) reasoning-augmented recommenders: **R^{2}EC** (You et al., 2025) and **LatentR^3** (Zhang et al., 2026). Appendix B provides their descriptions and implementation choices.

4.1.3 BACKBONE AND TRAINING VARIANTS

We use Qwen-2.5-Instruct (1.5B and 3B); all trainable LLM baselines use the same 3B backbone with their standard training recipes. Our default **SFT & RL** initializes structured reasoning before optimizing NDCG@10, with **RL-only** as the initialization ablation. Unless stated otherwise, ablations and in-depth analyses use the 3B model. Appendix C provides hyperparameters, hardware, and training time.

4.2 MAIN RESULTS

Table 2 reports performance on the three Amazon datasets and the industrial dataset. We highlight four observations about the effectiveness of R2Rank.

Training for ranking matters beyond model scale alone. With a 3B backbone, R2Rank achieves the highest NDCG@10 on three of four datasets, outperforming both DeepSeek-R1 (671B) on these domains and all trainable LLM baselines using the same backbone. General reasoning ability therefore does not automatically translate into strong recommendation ranking. A listwise LLM ranker must infer preferences, compare candidates, and generate an ordered list within one prompt. R2Rank instead learns to assess each candidate against the user's history and trains these assessments together through the final ranking reward, aligning reasoning with recommendation utility.

The advantage is more evident when collaborative signals are sparse. The largest margin appears on Industrial, where R2Rank reaches 0.818 NDCG@10 versus 0.560 for LatentR^3. Its much larger item catalogue and fewer interactions per item provide less repeated collaborative evidence. Assessing

Table 2: Recommendation performance (NDCG) on Amazon and Industrial datasets.

Method	Musical Instruments			Movies & TV			Video Games			Industrial		
	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10
Traditional Recommendation Models												
SASRec	0.257	0.416	0.468	0.107	0.205	0.258	0.223	0.332	0.366	0.072	0.115	0.201
BERT4Rec	0.218	0.328	0.422	0.065	0.148	0.228	0.166	0.267	0.304	0.095	0.168	0.229
LightGCN	0.248	0.402	0.455	0.102	0.198	0.247	0.215	0.320	0.356	0.051	0.143	0.238
LLM-based												
DeepSeek-R1 (671B)	0.198	0.341	0.512	0.310†	**0.520**†	**0.619**	0.290†	0.440†	0.561	0.292	0.438	0.500
Prompt4NR	0.173	0.305	0.421	0.171	0.352	0.464	0.161	0.289	0.432	0.102	0.241	0.446
RecRanker	0.268	0.395	0.441	0.195	0.355	0.420	0.228	0.362	0.425	0.155	0.305	0.428
TALLRec	**0.302**	0.412	0.458	0.156	0.268	0.354	0.261	0.384	0.412	0.126	0.281	0.402
R^2EC	0.289	0.408	0.449	0.288	0.417	0.502	0.241	0.358	0.468	0.239	0.301	0.498
LatentR^3	0.263	0.462	0.507	0.270	0.440	0.508	0.273	0.465	0.512	0.308	0.425	0.560
Ours												
R2Rank (1.5B-Instruct)	0.234	0.453	0.514	0.257	0.426	0.486	0.279	0.487	0.549	0.550†	0.715†	0.759
R2Rank (3B-Instruct)	0.279	**0.495**	**0.564**	**0.338**	**0.520**	0.581	**0.330**	**0.544**	**0.596**	**0.650**†	**0.812**†	**0.818**

candidate attributes against behavioral history offers an additional basis for estimating relevance in this setting. The frequency analysis in Appendix D.2 supports this interpretation: the largest gains occur on items with zero occurrences in the main RL-interaction training corpus. Section 4.5 further relates this pattern to transfer and the use of behavioral evidence.

R2Rank improves ranking quality beyond the top prediction. The RL stage optimizes NDCG@10 through the Plackett–Luce surrogate, aligning candidate assessments with the utility of the resulting ranking. The gains also extend to earlier cutoffs: the 3B variant leads at NDCG@5 on all four datasets and at NDCG@1 on three. TALLRec performs best at NDCG@1 on *Musical Instruments*. Thus, the improvement is visible across ranking cutoffs, although optimizing NDCG@10 does not guarantee the best top prediction in every domain.

World knowledge remains relevant for movie recommendation. *Movies & TV* is the exception at NDCG@10, where DeepSeek-R1 achieves 0.619 versus 0.581 for R2Rank. Preference inference in this domain can depend on recognizing sequels, directors, or actors across films. Broader knowledge of these entities may contribute to the larger model's advantage, although the comparison does not isolate that factor. R2Rank still outperforms all baselines using the same 3B backbone, suggesting that training for ranking remains useful even where the gap to a larger model persists.

4.3 Ablation study

We examine the contributions of backbone adaptation, SFT initialization, explicit reasoning, and self-check. All ablations use Qwen2.5-3B-Instruct and the same candidate sets of $K=20$; we report NDCG@10.

4.3.1 Effect of joint training with the LLM

To test whether adapting the backbone improves ranking beyond learning a scoring head, we compare a frozen LLM (**MLP-only**) with joint **LLM+MLP** training under the same **RL-only** setting. Joint training performs better on every dataset (Table 3), supporting adaptation of the backbone to the ranking objective.

4.3.2 Effect of SFT initialization

We compare **RL-only** with **SFT & RL** to test whether structured reasoning initialization improves subsequent ranking optimization. SFT improves NDCG@10 on every dataset (Table 4), showing that initialization provides benefits beyond those obtained from RL alone. The largest absolute gains occur on Industrial (0.109) and Video Games (0.081), suggesting that structured initialization remains useful even when the backbone is subsequently adapted with a ranking reward.

Table 3: Joint training vs. scoring head-only training (Qwen2.5-3B-Instruct, RL-only) where the results are reported under NDCG@10.

Dataset	MLP-only	LLM+MLP
Musical	0.368	**0.541**§
Movies & TV	0.322	**0.562**§
Video Games	0.397	**0.515**§
Industrial	0.528	**0.709**§

Table 4: Effect of SFT cold-start (Qwen2.5-3B-Instruct) where the results are reported under NDCG@10.

Dataset	RL-only	SFT & RL
Musical	0.541	**0.564**†
Movies & TV	0.562	**0.581**‡
Video Games	0.515	**0.596**§
Industrial	0.709	**0.818**§

4.3.3 Effect of explicit CoT output

To test whether generating explicit reasoning during RL improves ranking, we change only the RL prompt to request either a rationale followed by a decision (**With CoT**) or the decision alone (**Without CoT**). With CoT performs better on every dataset (Table 5), supporting the use of explicit reasoning in ranking training. The benefit varies across domains: Video Games shows the largest absolute improvement (0.138 NDCG@10), while Musical Instruments improves by 0.019. This comparison evaluates explicit reasoning as part of the training procedure; Section 4.4 separately examines the information contained in generated rationales with the trained checkpoints held fixed.

4.3.4 Effect of self-check

We test whether revisiting the initial rationale adds value beyond ordinary CoT. On *Musical Instruments*, we remove only `#Self-check`, connect the initial rationale directly to the conclusion, and rerun filtering, SFT, and RL, comparing the resulting plain CoT with No CoT and the full template (Table 6).

Table 5: Effect of explicit CoT output (Qwen2.5-3B-Instruct) where the results are reported by NDCG@10.

Dataset	Without CoT	With CoT
Musical	0.545	**0.564**†
Movies & TV	0.515	**0.581**§
Video Games	0.458	**0.596**§
Industrial	0.758	**0.818**§

Table 6: Effect of the self-check stage on *Musical Instruments* (Qwen2.5-3B-Instruct, NDCG@10).

Output structure	NDCG@10
No CoT	0.545
Plain CoT	0.494
Full template	**0.564**

The full template performs best, while plain CoT falls below No CoT, supporting the self-check stage within the training pipeline. Thus, additional reasoning text alone is insufficient in this comparison; the structure used to generate it also matters. Response length and the examples retained by filtering can also change, so the comparison does not isolate verification alone.

4.4 Effect of rationale content on ranking

Ablating CoT during training changes both the model and its computation budget. We use two experiments to examine whether rationale content influences ranking and whether it can help another model. Both experiments keep the checkpoints and candidate sets fixed. Detailed procedures are provided in Appendix D.7.

4.4.1 RATIONALE SUBSTITUTION

On all four datasets, original rationales outperform shuffled rationales and final answers alone (Table 7), with the scoring head and final answers held fixed. Shuffling exchanges rationales among candidates for the same user while preserving the rationale pool and its total token count. Performance drops below using only the final answer on three datasets, suggesting that matching a rationale to its candidate matters. Using only the final answer changes the input to a frozen model; it differs from removing CoT during training.

Table 7: Rationale substitution with frozen checkpoints. Results are NDCG@10; Original and Shuffled use the same rationale pool for each user.

Dataset	Original	Shuffled rationale	Answer only
Musical Instruments	**0.564**	0.508	0.522
Movies & TV	**0.581**	0.542	0.526
Video Games	**0.596**	0.533	0.551
Industrial	**0.818**	0.682	0.731

4.4.2 RATIONALE TRANSFER BETWEEN MODELS

On *Musical Instruments*, giving R2Rank rationales to DeepSeek-R1 raises NDCG@10 from 0.512 to 0.552 (Table 8). In the reverse direction, R2Rank scores 0.481 with DeepSeek-R1 rationales, compared with 0.564 using its own. The results suggest that rationales provide useful information to another model, while R2Rank performs best with its own rationales. Together, these experiments support the usefulness of rationale content, but do not show that every reasoning step faithfully explains the ranking.

Table 8: Bidirectional rationale transfer on *Musical Instruments* (NDCG@10).

Condition	NDCG@10
DeepSeek-R1	0.512
DeepSeek-R1 + R2Rank rationales	0.552
R2Rank + DeepSeek-R1 rationales	0.481
R2Rank	**0.564**

4.5 IN-DEPTH ANALYSES

Beyond aggregate NDCG, we examine where the gains arise and how R2Rank uses behavioral evidence. Training on *Video Games* and evaluating directly on *Movies & TV* yields 0.551 NDCG@10, while transfer in the reverse direction yields 0.494 (Appendix D.1). In-domain training remains stronger, but the off-diagonal results indicate that part of the learned comparison process transfers across domains. On Industrial, items with zero occurrences in the main RL-interaction corpus account for 68.7% of test positives; R2Rank achieves 0.796 versus 0.509 for LatentR^3 on this subset (Appendix D.2). This descriptive breakdown localizes the largest aggregate gain to a sparse-item regime without by itself establishing a causal effect of sparsity.

The remaining analyses examine how the models use their presented context. Retaining more recent interactions steadily improves industrial performance, whereas R2Rank remains stable across the tested history-length groups on *Musical Instruments* (Appendix D.3). DeepSeek-R1's predicted ranks vary when the same positive candidate is placed at different prompt positions (Appendix D.4), motivating candidate-wise assessment. Finally, the case in Appendix F illustrates how platform and content compatibility can outweigh a candidate's global rating. These analyses characterize model behavior under the evaluated protocols rather than prove that the generated rationale is a causal explanation of each ranking decision.

5 CONCLUSION

We presented Reasoning to Rank (R2Rank), which couples candidate-wise reasoning with listwise utility. Structured SFT establishes the self-reflective reasoning format; during reinforcement learning, a Plackett–Luce distribution supplies score-path gradients and PPO trains the sampled reasoning policy from the shared NDCG return. Across four datasets, R2Rank is competitive overall and strongest in the industrial sparse-item regime. Rationale substitution and cross-model transfer further indicate that the generated content carries useful candidate-specific information, although they do not establish complete rationale faithfulness. Future work should study harder candidate pools, cross-candidate calibration, and history-presentation robustness.

AI USE STATEMENT

We used generative AI tools for code debugging, data processing, presentation feedback, language editing, literature searches, and LaTeX formatting. DeepSeek-R1 generated the structured SFT responses described in Section 3.4 and Appendix A. We take responsibility for all text, claims, and artifacts produced with this assistance.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 1007–1014. ACM, 2023. doi: 10.1145/3604915.3608857. URL https://doi.org/10.1145/3604915.3608857.

Ethan Bito, Yongli Ren, and Estrid He. Evaluating position bias in large language model recommendations. *arXiv preprint arXiv:2508.02020*, 2025. doi: 10.48550/arXiv.2508.02020. URL https://arxiv.org/abs/2508.02020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Chongming Gao, Mengyao Gao, Chenxiao Fan, Shuai Yuan, Wentao Shi, and Xiangnan He. Process-supervised LLM recommenders via flow-guided tuning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2025), Padua, Italy, July 13–18, 2025*, pp. 1934–1943. ACM, 2025. doi: 10.1145/3726302.3729981. URL https://doi.org/10.1145/3726302.3729981.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 299–315. ACM, 2022. doi: 10.1145/3523227.3546767. URL https://doi.org/10.1145/3523227.3546767.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648. ACM, 2020. doi: 10.1145/3397271.3401063. URL https://doi.org/10.1145/3397271.3401063.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking LLMs as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.147. URL https://aclanthology.org/2026.acl-long.147/.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining*, pp. 197–206. IEEE, 2018. doi: 10.1109/ICDM.2018.00035. URL https://doi.org/10.1109/ICDM.2018.00035.

Jieyong Kim, Hyunseo Kim, Hyunjin Cho, SeongKu Kang, Buru Chang, Jinyoung Yeo, and Dongha Lee. Review-driven personalized preference reasoning with large language models for recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2025), Padua, Italy, July 13–18, 2025*, pp. 1697–1706. ACM, 2025. doi: 10.1145/3726302.3730055. URL https://doi.org/10.1145/3726302.3730055.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. ReLLa: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM Web Conference 2024*, pp. 3497–3508. ACM, 2024. doi: 10.1145/3589334.3645467. URL https://doi.org/10.1145/3589334.3645467.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. Order-agnostic identifier for large language model-based generative recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1923–1933. ACM, 2025. doi: 10.1145/3726302.3730053. URL https://doi.org/10.1145/3726302.3730053.

Dairui Liu, Boming Yang, Honghui Du, Derek Greene, Neil Hurley, Aonghus Lawlor, Ruihai Dong, and Irene Li. RecPrompt: A self-tuning prompting framework for news recommendation using large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pp. 3902–3906. ACM, October 2024a. doi: 10.1145/3627673.3679987. URL https://doi.org/10.1145/3627673.3679987.

Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. Is ChatGPT a good recommender? a preliminary study, 2023. URL https://arxiv.org/abs/2304.10149. The 1st Workshop on Recommendation with Generative Models at CIKM 2023.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Sichun Luo, Bowei He, Haohan Zhao, Wei Shao, Yanlin Qi, Yinya Huang, Aojun Zhou, Yuxuan Yao, Zongpeng Li, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. RecRanker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Transactions on Information Systems*, 43(5):1–31, 2025. doi: 10.1145/3705728. URL https://doi.org/10.1145/3705728. Article 113.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://doi.org/10.52202/075280-2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. doi: 10.52202/068431-2011. URL https://doi.org/10.52202/068431-2011.

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975. doi: 10.2307/2346567. URL https://doi.org/10.2307/2346567.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. doi: 10.52202/075280-2338. URL https://doi.org/10.52202/075280-2338.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Techniques, applications, and challenges. In Francesco Ricci, Lior Rokach, and Bracha Shapira (eds.), *Recommender Systems Handbook*, pp. 1–35. Springer US, New York, NY, third edition, 2022. doi: 10.1007/978-1-0716-2197-4_1. URL https://doi.org/10.1007/978-1-0716-2197-4_1.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450. ACM, 2019. doi: 10.1145/3357384.3357895. URL https://doi.org/10.1145/3357384.3357895.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pp. 1057–1063. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965, 2023. doi: 10.52202/075280-3275. URL https://doi.org/10.52202/075280-3275.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://doi.org/10.52202/068431-1800.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. LLMRec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 806–815. ACM, 2024. doi: 10.1145/3616855.3635853. URL https://doi.org/10.1145/3616855.3635853.

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. A survey on large language models for recommendation. *World Wide Web*, 27(5):60, 2024. doi: 10.1007/s11280-024-01291-2. URL https://doi.org/10.1007/s11280-024-01291-2.

Runyang You, Yongqi Li, Xinyu Lin, Xin Zhang, Wenjie Wang, Wenjie Li, and Liqiang Nie. R^2ec: Towards large recommender models with reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 62376–62405, 2025. doi: 10.52202/085713-2088. URL https://doi.org/10.52202/085713-2088.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. CoLLM: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340, 2025. doi: 10.1109/TKDE.2025.3540912. URL https://doi.org/10.1109/TKDE.2025.3540912.

Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Reinforced latent reasoning for LLM-based recommendation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eUtIZT2ONS.

Zizhuo Zhang and Bang Wang. Prompt learning for news recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pp. 227–237. ACM, July 2023. doi: 10.1145/3539618.3591752. URL https://doi.org/10.1145/3539618.3591752.

A Data accounting and SFT construction

A.1 Main corpus and auxiliary SFT users

The dataset statistics in Table 1 describe the main experimental corpus. The auxiliary SFT corpus and the main experimental corpus are disjoint at the user and interaction levels: no SFT user or user–item interaction appears in the main training, validation, or test splits. For all three Amazon datasets, the reported interaction counts cover October 2021–October 2023, whereas auxiliary SFT instances are sampled from earlier interactions outside this main window after excluding all main-split users. The SFT interactions are therefore outside the reported interaction counts and also temporally separated from the main Amazon corpus. For the larger proprietary Industrial database, we sample non-overlapping user pools of approximately 5,000 users for SFT and approximately 100,000 users for the main training, validation, and test pool. These separation rules do not impose item-level disjointness; below, an "unseen" item means zero occurrences in the main RL training interactions.

The main users are split at an 8:1:1 ratio. On Musical Instruments, these user-level splits contain 39,164 RL training instances, 5,105 validation instances, and 4,902 test instances; these are instance counts rather than user counts. This distinction separates the unit used for splitting from the number of resulting user–item examples.

A.2 SFT queries and filtering

Standard recommendation datasets do not provide reasoning annotations, so we synthesize SFT supervision from the disjoint auxiliary interactions. For each instance (c_u, x_i), we instantiate the shared template with the user context and candidate description, then query DeepSeek-R1 for a complete self-reflective response. We choose this teacher for its long-form reasoning and self-verification capabilities. The resulting supervision consists of text responses rather than hidden representations, so it does not require architectural matching between the teacher and the target backbone. On Musical Instruments, we query DeepSeek-R1 for 4,000 user–item instances and retain approximately 2,000 responses, corresponding to a keep rate of roughly 50%. The retained set is approximately one twentieth of the RL training set in size. A response is retained only if it follows the required three-part schema and its final `Recommend`/`Not Recommend` decision agrees with the observed label. Discarded responses are not distilled. This filtering checks output structure and final-label agreement; it does not certify every intermediate reasoning statement.

SFT provides a structured reasoning initialization and no listwise ranking targets. Listwise supervision enters during RL. The RL-only comparison in Table 4 isolates the contribution of this warm start: without SFT, R2Rank reaches 0.541, 0.562, 0.515, and 0.709 NDCG@10 on Musical, Movies, Video Games, and Industrial, respectively, compared with 0.564, 0.581, 0.596, and 0.818 after SFT initialization.

A.3 Fixed random-negative candidate construction

The default evaluation protocol contains one observed positive and 19 negatives drawn uniformly without replacement from the item pool, excluding the positive and any item in the user's observed history. The seed is fixed by sample index, and candidate IDs are shared across all methods. These candidates are not constructed by popularity sampling, in-batch sampling, or a retrieval model. The larger-candidate evaluation in Appendix D.6 retains the complete original candidate list and adds 30 distinct history-excluded negatives under a fixed seed.

Amazon reviews do not provide impression positions, and the industrial click records do not retain exposure or display-position metadata. Our comparisons therefore use the observed feedback distribution. Random candidate construction does not remove exposure bias in the logged interactions; this is separate from candidate presentation-order sensitivity in an LLM prompt.

A.4 Preprocessing and evaluation details

For all datasets, we treat observed user–item interactions as implicit positives and construct user sequences by sorting their interactions chronologically. To match the LLM ranking setting and control context length, we truncate each history to the most recent $L{=}20$ interactions. The Amazon time window described above reduces temporal drift and alleviates item availability mismatch. The industrial dataset has a larger item space and interaction volume, with a more pronounced sparse-item regime in the main RL corpus; Appendix D.2 reports the frequency-stratified results.

We evaluate our fixed candidate sets using NDCG@$\{1, 5, 10\}$. Unless stated otherwise, significance markers $^{\dagger}/^{\ddagger}/^{\S}$correspond to $p < 0.05/0.01/0.001$ under a two-sided paired Wilcoxon signed-rank test.

B Baselines

We provide detailed descriptions of each baseline included in Table 2, grouped by paradigm. When official implementations are available, we use the released code and recommended hyper-parameters; otherwise, we conduct limited validation-set tuning to ensure stable performance. All baselines are evaluated under the same candidate-set protocol as R2Rank.

• Conventional recommenders.

- **SASRec** (Kang & McAuley, 2018) is a Transformer-based sequential recommender that applies unidirectional self-attention over item sequences to predict the next item, representing the unidirectional self-attention paradigm.
- **BERT4Rec** (Sun et al., 2019) adopts a bidirectional Transformer encoder trained with a masked-item prediction objective (Cloze task), representing the bidirectional sequential paradigm.
- **LightGCN** (He et al., 2020) performs simplified neighborhood aggregation over the user–item bipartite graph, removing feature-transformation matrices and nonlinear activations while retaining trainable user and item embeddings.

• Prompted and instruction-tuned language-model recommenders.

- **DeepSeek-R1** (Guo et al., 2025) is a general-purpose large reasoning LLM. We include it as a *zero-shot prompted* ranker that directly outputs a ranked list given the user history and the candidate set, representing the upper envelope of task-agnostic LLM reasoning without any recommendation-specific training.
- **Prompt4NR** (Zhang & Wang, 2023) formulates recommendation as a *pointwise* cloze-style prediction, templating the user history and a candidate into a prompt that asks the masked language model to predict a binary verbalizer (e.g., `yes`/`no`) at a `[MASK]` position; candidates are then ranked by the predicted probability.
- **RecRanker** (Luo et al., 2025) instruction-tunes an LLM as a top-k ranker with a *hybrid* training scheme that jointly includes *pointwise* (single-item scoring), *pairwise* (preference between two items), and *listwise* (output of an ordered list over a small set) objectives on top of conventional retrieval models.

- **TALLRec** (Bao et al., 2023) casts recommendation as a *pointwise* binary instruction-following task in which the LLM is asked to output `yes`/`no` for each candidate given the user's liked/disliked histories, with a two-stage tuning recipe (alpaca-style instruction tuning followed by task-specific tuning).

• **Reasoning-augmented LLM recommenders.**

- **R^{2}EC** (You et al., 2025) integrates explicit reasoning traces with recommendation learning via a dual-head architecture that reuses the decoder backbone: a language-modeling head generates the rationale while a recommendation head performs *pointwise* scoring of each candidate from the post-reasoning hidden states, trained with RL-style rewards.
- **LatentR^3** (Zhang et al., 2026) adds a latent-attention layer that generates continuous latent reasoning tokens. It uses an SFT warm-up followed by modified GRPO, with the negative perplexity of the ground-truth item serving as the continuous reward during RL rather than as a pointwise inference score.

C Implementation details

We implement R2Rank on top of Qwen-2.5-Instruct (1.5B and 3B) loaded in `bf16` with FlashAttention-2. The scoring head f_ϕ is a two-layer MLP (hidden size 1024, GELU) that consumes the last-token hidden state of each reasoning segment y_k. The scoring head is randomly initialized and jointly trained with the backbone. The vLLM rollout worker returns the sampled tokens and their cached log-probabilities under its frozen policy snapshot. The trainer teacher-forces those tokens through the current backbone to recompute current token log-probabilities and last-token hidden states with automatic differentiation enabled. Cached old-policy log-probabilities remain detached, while the PL score path is retained through the recomputed hidden states and propagated into the backbone.

Conventional recommenders. SASRec, BERT4Rec, and LightGCN are trained with Cross-Entropy loss, using embedding dimension 64 and validation NDCG@10 for early stopping.

LLM-based baselines. All trainable language-model baselines (Prompt4NR, RecRanker, TALLRec, R^{2}EC, LatentR^3) are fine-tuned on the same Qwen-2.5-3B-Instruct backbone following the training recipes reported in their original papers.

R2Rank (ours). We use learning rate 3e-5 for the scoring head and a scaled backbone learning rate with λ_{llm}=0.14 for the 3B model and 0.18 for the 1.5B model, with gradients clipped at 1.0. For each user we sample K=20 per-candidate reasoning trajectories and draw an ordered PL prefix of length K_{eval}=10, whose NDCG@10 is used as the scalar reward; PPO uses clipping ratio ϵ=0.2. Rollouts are produced by a vLLM worker launched through Ray and synchronized with the trainer every 5 optimizer steps; at inference we output a deterministic ranking by descending `argsort` of the scores. All experiments run on a single node with $8\times$NVIDIA H20 GPUs, with one GPU hosting the vLLM rollout actor and the remaining seven running DDP training with candidate-level parallelism. One RL epoch takes roughly 40 hours for the 3B model and 30 hours for the 1.5B model.

D In-depth analyses

We further examine when R2Rank is effective through cross-dataset transfer, item-frequency stratification, history-length analysis, and candidate-position sensitivity. Together with the case study in Appendix F, these analyses relate the observed ranking behavior to the user-history evidence and candidate attributes available to the model.

D.1 Cross-dataset generalization

We train R2Rank on one Amazon dataset and directly evaluate it on another without further fine-tuning. Table 9 reports NDCG@10, where diagonal entries are in-domain performance and off-diagonal entries quantify cross-domain transfer. The off-diagonal results retain non-trivial ranking quality: for example, training on *Video Games* gives 0.551 on *Movies & TV*, while transfer in the reverse direction gives 0.494. This behavior is consistent with learning a reusable process for comparing candidate

attributes with evidence in the user's history, beyond memorizing domain-specific co-occurrence patterns. In-domain training generally remains stronger, so transfer does not eliminate the value of domain-specific adaptation.

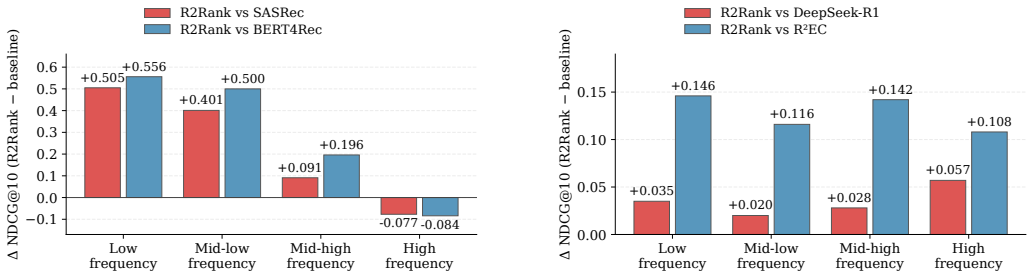
Table 9: Cross-dataset generalization evaluated by NDCG@10. Models are trained on one dataset (rows) and directly evaluated on another (columns) without any other fine-tuning.

Train \ Test	Musical	Video Games	Movies & TV
Musical	0.564	0.491	0.412
Video Games	0.532	0.596	0.551
Movies & TV	0.463	0.494	0.581

D.2 Main-RL item-frequency stratification

To analyze robustness to sparse collaborative evidence, we stratify positive items in the *Video Games* dataset into four equal-sized bins by their frequency in the main RL-interaction training corpus and evaluate recommendation quality within each tier. This stratification does not assert that an item is absent from the auxiliary SFT corpus. Figures 2a and 2b report the NDCG@10 improvement of R2Rank over strong sequential baselines (SASRec, BERT4Rec) and representative LLM-based recommenders (DeepSeek-R1, R^2ec), respectively.

R2Rank achieves the most pronounced gains on the lowest-frequency tiers, where repeated interaction evidence is scarce. The advantage holds against both SASRec/BERT4Rec and the LLM-based baselines in Figure 2b, showing that the observed gain extends beyond the comparison with conventional recommenders alone. These results are consistent with candidate-level reasoning drawing on item attributes and relevant user-history evidence when collaborative signals are limited. At higher frequencies, conventional recommenders have more interaction evidence available and the observed margins are smaller.



(a) vs. sequential models (Video Games)

(b) vs. LLM-based models (Video Games)

Figure 2: NDCG@10 improvement of R2Rank over sequential and LLM-based baselines across item-frequency bins on *Video Games*. The Industrial breakdown is reported in Table 10.

Industrial training-frequency breakdown. Industrial has 1.55 interactions per item, compared with 2.70–5.67 on the Amazon datasets, and its item catalogue is approximately 43 times larger than the largest Amazon catalogue studied here. We stratify industrial test positives by the number of occurrences of their items in training. Table 10 reports the fractions of test positives and NDCG@10 within each stratum.

Items with zero occurrences in the main RL-interaction training corpus account for 68.7% of industrial test positives and exhibit the largest absolute improvement. Warm items show an 11.3% relative gain, closer to the gains on Amazon. This breakdown helps explain why the aggregate industrial improvement over LatentR^3 is larger: the industrial test set places substantially more weight on zero- and low-frequency items under this definition. The stratification is descriptive evidence about where the gains occur, rather than an isolated causal test of sparsity.

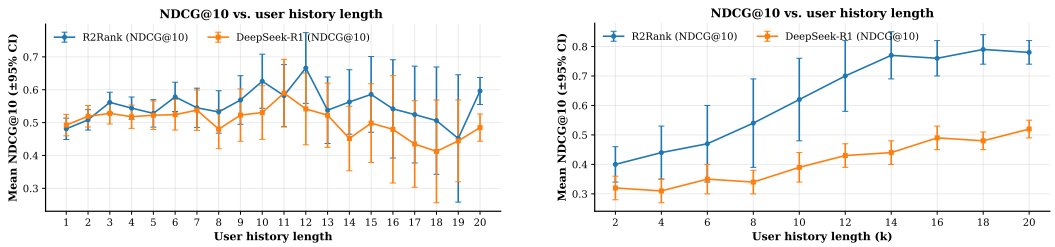
Table 10: Industrial NDCG@10 by the positive item's frequency in the main RL-interaction training corpus. The last column is the absolute improvement over LatentR^3.

Stratum	Share	R2Rank	LatentR^3	Δ
Zero main-RL occurrences	68.7%	0.796	0.509	+0.287
Low-frequency (1–5)	21.6%	0.852	0.611	+0.241
Warm (> 5)	9.7%	0.898	0.807	+0.091

D.3 Influence from user history length

We analyze the impact of user history length by stratifying *Musical Instruments* test instances according to the number of observed historical interactions and reporting mean NDCG@10 under each length category. For *Industrial*, we vary the retained recent-history length as detailed below. Figure 3 shows the results (mean $\pm$ 95% CI) for both datasets.

Across the history-length categories on *Musical Instruments*, R2Rank maintains stable performance and remains competitive with DeepSeek-R1. This pattern is consistent with assessing candidate relevance using pertinent evidence from histories of different lengths, rather than requiring one specific amount of user context.



(a) *Musical Instruments* (b) *Industrial*

Figure 3: Influence of user history length on recommendation quality.

In the industrial dataset, long user histories are the norm: over **95%** of users have more than 20 historical interactions. To evaluate robustness under varying available context in this setting, we randomly sample **100** test instances whose truncated history length equals 20, and repeatedly construct inputs by keeping only the most recent k interactions in user history. We compare R2Rank against DeepSeek-R1 under the same candidate-set evaluation protocol.

As shown in Figure 3b, R2Rank benefits more clearly from longer histories: its performance improves steadily as k increases and remains consistently competitive with DeepSeek-R1 across all settings. Retaining more recent interactions supplies additional behavioral evidence, and the observed gains are consistent with R2Rank using that evidence to assess candidate compatibility. This is relevant to the industrial setting, where long histories are common. These curves assess the complete model rather than isolate individual components.

D.4 Candidate position sensitivity in prompt-based recommenders

A key motivation of R2Rank is to mitigate candidate presentation-order bias in LLM input, where candidate ordering can inadvertently influence the model's output. R2Rank avoids this issue by performing independent user–item assessments: each candidate is evaluated in isolation under the same user context, and the final ranking is induced by sorting the scores produced by a shared scoring head. As a result, a candidate's assessment does not condition on its presentation position in the candidate list.

To illustrate the practical impact of position bias in the baseline LLM solutions, we perform a controlled evaluation on *Musical Instruments* using DeepSeek-R1. We place the ground-truth positive item at different positions in the candidate list (position 1, 10, or 20) while keeping the remaining candidates unchanged, and evaluate the distribution of the predicted rank assigned to the positive

item. Figure 4 shows a clear positional impact: placing the positive item at the first position leads to noticeably better outcomes than placing it at later positions. The relevance labels and candidate identities are fixed, so this shift reflects sensitivity to how the same ranking problem is presented. It motivates separating candidate-level evidence extraction from list presentation, as in R2Rank, while retaining listwise optimization of the resulting scores. This candidate-order issue is distinct from the ordering of interactions inside a user's history, discussed in Appendix D.5, and from exposure bias in logged feedback, discussed in Appendix A. Independent candidate assessment addresses the first issue; it does not remove the other two.

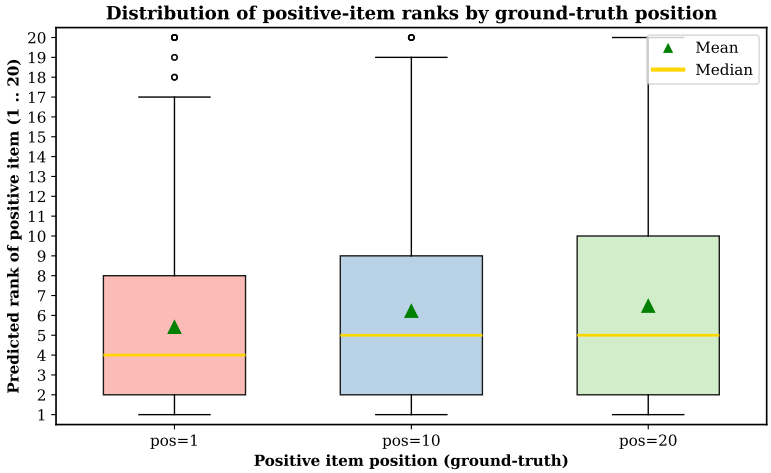


Figure 4: Candidate position sensitivity of DeepSeek-R1 on the *Musical Instruments* dataset.

D.5 Sensitivity to user history order

Modeling the sequential pattern in user's interaction history is known to be important for quality recommendations (Kang & McAuley, 2018; Sun et al., 2019). But the influence from the ordering of items in user history is compounded with position sensitivity in LLMs (Bito et al., 2025): the history is serialized as a token sequence, so reordering it changes the conditioning context and can alter the generated rationale and resulting score.

We probe this impact on the *Video Games* dataset by randomly permuting the interaction history in the testing instances and comparing against the original chronological order. As shown in Table 11, although the mean NDCG@10 is close across settings for R2Rank as well as SASRec/BERT4Rec, the variance of R2Rank performance is much larger than the baselines'. Unfortunately, when examining the detailed results, the input with chronological order does not necessarily led to the best results in R2Rank. Ideally, sensitivity to history order should reflect meaningful temporal effects (e.g., preference drift and recency), not artifacts of how the context is serialized. This motivates future work on reducing context presentation bias, such as order-robust history representations or training-time augmentations that encourage stability across alternative history serializations while preserving genuine temporal signals.

Table 11: Sensitivity to user history order on *Video Games* (NDCG@10). We report the mean, sample standard deviation, and range under shuffled histories, together with the chronological results from Table 2.

Method	Shuffled mean	Sample std.	Range	Chronological reference
R2Rank	0.594	0.085	0.211	0.596
SASRec	0.365	0.030	0.062	0.366
BERT4Rec	0.303	0.017	0.036	0.304

D.6 Evaluation with larger candidate sets

To test whether the gains persist beyond 20 candidates, we construct strictly nested Random-50 lists on *Musical Instruments*. Every candidate in the original 20-item list is retained, including the observed positive and all 19 original negatives; 30 additional history-excluded items are sampled without replacement under a fixed seed. All methods receive identical candidate IDs at each size. This keeps the original ranking instance intact while increasing the number of alternatives.

Table 12: NDCG@10 on nested candidate sets with K=20 and K=50 for *Musical Instruments*.

Method	$K = 20$	$K = 50$
R2Rank	**0.564**	**0.356**
LatentR^3	0.507	0.298
SASRec	0.468	0.263

All three methods decline as the candidate set grows, but R2Rank remains strongest at both sizes (Table 12). Its NDCG@10 margin over LatentR^3 is 0.057 at K=20 and 0.058 at K=50. The advantage therefore persists in this larger random-negative setting. This experiment evaluates two candidate sizes on one dataset, rather than full-catalogue ranking or candidate pools supplied by a production retriever.

D.7 Rationale substitution and transfer protocols

Rationale substitution. We use the original checkpoints and the same fixed candidate lists on all four datasets. For each user–candidate prompt, we compare three completions: (i) its original rationale and final answer; (ii) a rationale reassigned by randomly permuting the 20 candidate rationales within the same user, followed by the original final answer; and (iii) the final answer alone. Each completion is teacher-forced under the original prompt and rescored using the same frozen backbone, MLP head, and answer-relative hidden-state extraction position.

Shuffling preserves the user, candidate set, rationale pool, and total rationale-token budget within each user. It does not match the length of each individual reassigned rationale. The answer-only condition differs from the **Without CoT** training ablation in Table 5: the former changes only the completion supplied to a frozen model, whereas the latter changes the RL training prompt. Their Musical results, 0.522 and 0.545 respectively, are separate controls.

Bidirectional rationale transfer. On the same fixed candidate lists for *Musical Instruments*, all checkpoints are frozen and no additional training is performed. DeepSeek-R1 first ranks candidates using its original listwise prompt. We then insert the candidate-level rationales generated by R2Rank into that prompt before ranking. In the reverse direction, DeepSeek-R1 generates a rationale for each candidate, which is supplied to the frozen R2Rank backbone and scoring head. The complete R2Rank with its own rationales serves as the reference. Transfer changes input content and length, so it complements the substitution experiment without independently establishing complete rationale faithfulness.

E Prompt template

As discussed in Section 3.4, we enforce a three-part self-reflective output schema via the instruction template in Figure 5. The template is shared across domains and only the placeholders (user context and candidate description) are instantiated with domain-specific fields. These fields may include user profiles, interaction histories, item titles, metadata, or textual content.

A pretrained LLM can produce inconsistent formats or shortcut rationales under long histories and diverse item descriptions. Such variation can increase reward variance and make the representations consumed by the scoring head less consistent. The SFT template therefore establishes a common reasoning pattern before RL, drawing on the verification-oriented behavior of long-reasoning models.

The self-check revisits the initial rationale against the supplied context, looking for extraction errors, unsupported inferences, missing evidence, or over-weighted signals. It is intended to encourage more consistent evidence use and a structured representation near the final decision token for the scoring head. These are design motivations; the contributions of SFT initialization and the self-check component are evaluated in Tables 4 and 6, respectively.

Self-check Prompt Template

You are a recommendation assistant. Analyze the user's history in depth and decide whether to recommend the candidate item.
Requirements: Generate a response with the following three parts in order:
#Initial reasoning: thoroughly analyze relevant evidence from USER CONTEXT and CANDIDATE ITEM.
#Self-check: identify mistakes, missing evidence, uncertainties, or over-weighted signals in the initial reasoning.
#Final conclusion and suggestion: produce a concise final rationale and decision.
Finally output: `<answer>Recommend</answer>` or `<answer>Not Recommend</answer>`.

User Context: {user_profile}, {user_history}
Candidate Item: {item_text}

Figure 5: The self-reflective prompt template adopted in R2Rank.

F CASE STUDY

Figure 6 presents one positive and one negative candidate under the same *Video Games* user context. The condensed rationales follow the initial reasoning, self-check, and final conclusion structure in Figure 5, retaining the evidence relevant to each decision. The user's recent purchases include family-friendly entertainment, a vehicle/racing game, and a Switch carrying case. Together these suggest an interest in accessible family or vehicle-related content and possible Switch use, although an accessory purchase alone does not establish console ownership.

For the positive candidate, *Blaze and the Monster Machines Axle City Racers* on Nintendo Switch, the rationale connects the game's family-friendly racing theme and platform to those recent signals. The self-check qualifies the platform inference and recognizes that the shared "monster" theme is weaker evidence than the broader family/vehicle preference. The final recommendation therefore retains the compatibility judgment while acknowledging the uncertainty in the Switch signal.

For the negative candidate, *Yakuza Kiwami* on PS4, the high rating and favorable crowd feedback do not outweigh the mismatch with the observed context. The self-check emphasizes that the history supplies Xbox and possible Switch evidence, but no evidence of PS4 ownership, and that mature crime-drama content differs from the recent family-friendly purchases. The final assessment gives these user-specific platform and content cues more weight than the candidate's rating or popularity. This example illustrates how the structured rationale can revisit attractive but insufficient signals and relate the decision to user-history evidence. The case is illustrative; the separate self-check ablation is reported in Table 6.

prompt 🤔

(Uses the structured self-check template in Figure~2; below we only instantiate the template with the user context and candidate item fields.)

 # USER HISTORY:
Contains purchase times, purchased items, and ratings (out of 5).
19.7min ago: [PAW Patrol The Movie: Adventure City Calls - Xbox [Digital Code]] (5.0)
14.8min ago: [Monster Truck Championship (XSX) - Xbox Series X] (5.0)
9.2min ago: [Daydayup Switch Carrying Case Compatible with Nintendo Switch/Switch OLED, with 20 Games Cartridges Protective Hard Shell Travel Carrying Case Pouch for Console & Accessories, Black] (5.0)

 # CANDIDATE ITEM 1:
Title: Blaze and the Monster Machines Axle City Racers - Nintendo Switch
User Rating: 4.7
Number of Buyers: 1437
Description: ... a Switch racing game with split-screen multiplayer, stunts, customization, and kid-friendly/accessible options ...

 # CANDIDATE ITEM 2:
Title: Yakuza Kiwami - PlayStation Hits - PlayStation 4
User Rating: 4.8
Number of Buyers: 773
Description: ... a gritty, story-driven action game remade for PS4, featuring mature crime-drama themes and Japanese voice track ...

Positive Case (Recommend) 😍

#Initial reasoning:
The history indicates recent interest in family-friendly titles (PAW Patrol) and vehicles/racing (Monster Truck Championship), plus a strong device signal for Switch via the carrying case; the candidate is a kid-friendly Switch racing game, so platform compatibility and thematic fit are both plausible, supported by strong crowd feedback.

#Self-check:
Key uncertainty is that Switch ownership is inferred from an accessory (not a game purchase), and the evidence for "racing on Switch" is indirect; **the "monster" thematic link is weaker** than the platform/accessory cue and the general preference for family/vehicle content.

#Final conclusion and suggestion:
This is a high-likelihood match driven by (i) inferred Switch engagement and (ii) recent purchases consistent with family entertainment, while acknowledging that the platform signal comes from an accessory rather than explicit Switch gameplay.
<answer>Recommend</answer>

Negative Case (Not Recommend) 😞

#Initial reasoning:
The recent history suggests the user is currently engaging with Xbox and possibly Switch, and prefers family-friendly, accessible titles. The candidate is a PS4-only, mature crime-drama action game with a very different audience. With no evidence of PS4 ownership or interest in this style, it is a poor match despite strong ratings.

#Self-check:
- **Strongest mismatch:** platform (PS4 vs. observed Xbox/Switch evidence).
- **Secondary mismatch:** content/age rating (family-friendly vs. mature crime drama).
- **Uncertainty:** user could own PS4, but there is zero supporting signal in the observed context.

#Final conclusion and suggestion:
The candidate conflicts with both the user's inferred active platforms and recent content preferences; the high rating is insufficient to offset the platform and intent mismatch.
<answer>Not Recommend</answer>

Figure 6: Case study of R2Rank on *Video Games*: positive (Recommend) and negative (Not Recommend) reasoning examples produced under the structured self-reflective format. For brevity, we display condensed reasoning content while retaining the key evidence and self-check signals.