FLEXREC: ADAPTING LLM-BASED RECOMMENDERS FOR FLEXIBLE NEEDS VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern recommender systems must adapt to dynamic, need-specific objectives, yet most are optimized for one static target. Reinforcement-learning post-training gives LLMs strong instruction-following abilities, suggesting a route to aligning them with complex recommendation goals. We study closed-set autoregressive ranking, where an LLM permutes a fixed candidate set given user context and a need instruction. Applying RL here raises two challenges: (i) an item's value changes across objectives, but sequence-level rewards collapse a ranking to one scalar and cannot identify which placements satisfy the current need; and (ii) interaction feedback is sparse and noisy, with reward reliability varying across users, items, and needs, causing inefficient and unstable updates. We propose FlexRec, a post-training RL framework combining (1) a sequential-dependence-aware item-level reward based on swaps within the remaining candidate pool and (2) critic-guided, uncertainty-aware scaling that models reward uncertainty and down-weights low-confidence signals. Across recommendation scenarios and objectives, FlexRec outperms strong baselines and improves NDCG@5 by up to **59%**, Recall@5 by up to **109.4%** in need-specific ranking, and Recall@5 by up to **24.1%** in generalization over the base model.

1 INTRODUCTION

Existing recommender systems are typically optimized for a single objective, such as click-through rate in advertising or purchases in e-commerce platforms (He & Chua, 2017; Guo et al., 2017). However, real-world recommendation often requires the ranking strategy to change with the user's immediate intent, the platform context, or the provider's business goal. We refer to this setting as *multi-need recommendation*: given the same user context and candidate items, the system should produce different rankings under different explicit needs. These needs may be partially aligned but can also conflict, as practical systems often balance objectives whose optimal rankings differ (Jannach, 2023; Nguyen et al., 2017).

LLMs offer a promising direction for flexible recommendation because their instruction-following and generalization abilities allow them to condition on user context, item metadata, and explicit task specifications (Wei et al., 2022; Wang et al., 2025b; Brown et al., 2020; Gao et al., 2021). Reinforcement learning from verifiable rewards (RLVR) (Lambert et al., 2024) has recently emerged as an effective paradigm for aligning LLMs with downstream objectives, and recent work has begun applying it to LLM-based recommendation: Rec-R1 (Lin et al., 2025) optimizes LLM rankers with sequence-level recommendation rewards using GRPO (Shao et al., 2024), while ConvRec-R1 (Zhu et al., 2025b) explores rank-level GRPO for conversational recommendation.

However, adapting LLMs for multi-need recommendation through GRPO introduces two fundamental challenges. First, in multi-need recommendation, item utility varies with the underlying need, so the training signal must identify which individual placements satisfy the current need. However, existing RL methods for LLM-based recommendation systems often operate with sequence-level rewards Shao et al. (2024); Deng et al. (2025); Wu et al. (2025), assigning a single scalar value to the entire ranked list. Yet autoregressive LLM recommendation admits a natural sequential interpretation: the generated ranking can be analyzed as a sequence of item placements, each conditioned on the

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.

previously generated prefix. A single list-level reward therefore fails to capture which item placement is beneficial or harmful, obscuring the fine-grained, need-specific credit assignment required for multi-need adaptation.

Second, interaction annotations are inherently sparse. In realistic scenarios, only a small subset of items receives explicit feedback from the user, while the majority remain unlabeled. Existing work often relies on learned interaction predictors to complete missing supervision (Yang et al., 2025; Wang et al., 2025a), but these estimates inevitably contain error and can vary substantially in reliability. This issue is further amplified in need-aware recommendation, where different needs may depend on different feedback signals, causing reward-estimation reliability to vary. This is especially problematic in RL, where high-variance reward estimates can strongly distort advantage estimation and destabilize policy updates (Zhu et al., 2025a).

To address these challenges under multi-need recommendation, we propose FlexRec, a principled post-training framework for aligning LLM-based recommender systems with dynamic recommendation strategies. Our key observation is that RL for recommendation must simultaneously solve two problems: **fine-grained credit assignment** over a ranked list and **robust optimization** under sparse, noisy reward completion. Our contributions are:

- **Swap-based item-level reward.** We use counterfactual swaps within the remaining candidate pool to evaluate each placement's marginal contribution, producing dense, position-aware supervision for autoregressive LLM ranking.
- **Uncertainty-guided GRPO.** To address the challenge of sparse and noisy reward completion, we train a critic that jointly predicts reward values and their uncertainty, and integrate this confidence signal directly into GRPO updates to down-weight unreliable reward estimates.
- **Multi-need adaptation.** Building on the proposed item-level reward and uncertainty-guided GRPO, we show that a single LLM recommender can be post-trained to follow different recommendation needs. We build three realistic needs on open recommendation datasets: *Maximizing User Interest*, *Explore New Topics*, and *Trend Promotion*, enabling need-conditioned training and evaluation across diverse ranking objectives.

Together, these components enable stable and effective alignment of LLMs for dynamic, need-aware recommendation. Our experiments across diverse recommendation scenarios and user needs demonstrate that FlexRec effectively aligns LLMs as strong recommenders, outperforming both traditional and LLM-based baselines with gains of up to **59.2%** in NDCG@5 and **109.4%** in Recall@5 over base model. We further show that the trained LLM recommender generalizes efficiently to unseen needs. Finally, a single LLM trained jointly on all needs remains competitive across all scenarios, dynamically adapting its reasoning and ranking strategy, making it practical as a universal recommender.

2 Related Work

LLMs for Multi-Need Recommendation Recommender systems have shifted from traditional collaborative filtering and ID-based sequential encoders (e.g., BERT4Rec (Sun et al., 2019) and S3-Rec (Zhou et al., 2020)) toward generative paradigms built on Large Language Models (LLMs). Current approaches typically rely on instruction-tuning (Deng et al., 2025; Lin et al., 2025; Rajput et al., 2023; Wang et al., 2024) or in-context learning (Lyu et al., 2024) to leverage LLMs' semantic reasoning and world knowledge for modeling user behavior. However, most LLM-based recommenders remain *static evaluators* of historical data (Wu et al., 2018; Dhelim et al., 2022; Ma et al., 2024; Zhang, 2023; Yang et al., 2023a): they optimize for a single implicit objective and can struggle when a user's immediate, real-time requirements deviate from their long-term profile. We contribute need-specific post-training datasets and FlexRec, which conditions rankings on user needs at inference.

Reinforcement Learning from Verifiable Rewards Reinforcement Learning from Verifiable Rewards (RLVR) has significantly enhanced LLM reasoning in deterministic domains such as mathematics (Shao et al., 2024), coding (Chen, 2021), and open-domain tasks (Ma et al., 2025). Recent RLVR-based post training methods for recommendation like Rec-R1 use sequence-level metrics such as NDCG (Lin et al., 2025) to guide model training. While effective, this design assigns a single advantage to all tokens in a rollout, providing no explicit distinction between individual

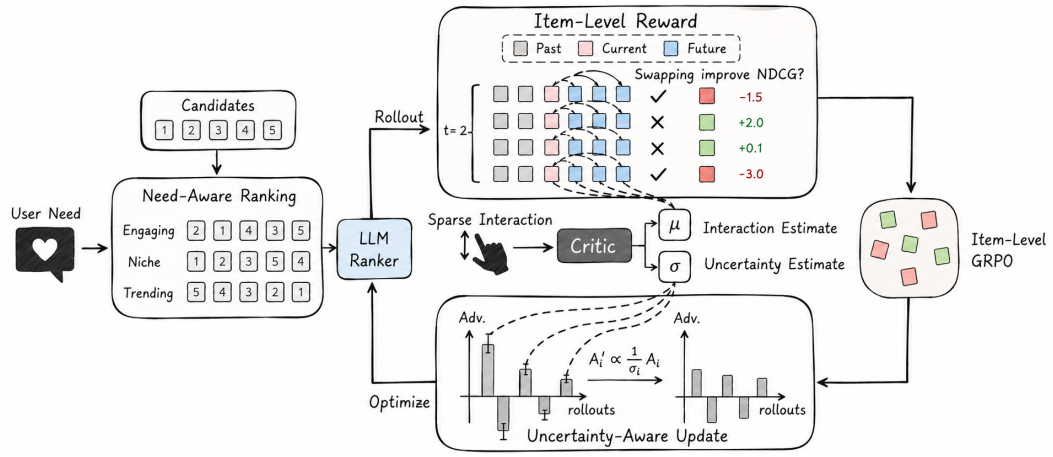


Figure 1: Overall framework of FlexRec. Given a candidate set and an explicit user need, an LLM recommender generates ranked rollouts. An item-level reward is computed by evaluating the marginal contribution of individual item placements via counterfactual swaps (top). A critic predicts both the expected interaction reward and its uncertainty for unobserved interactions (middle). These estimates are used to form uncertainty-aware advantages, down-weighting unreliable signals.

item placements within a sampled response. ConvRec-R1 uses Rank-GRPO, which introduces rank-level credit, causal reward shaping, and importance ratios based on geometric-mean item-token probabilities (Zhu et al., 2025b). Pairwise NDCG swap changes also underlie LambdaRank (Burges et al., 2006). FlexRec's distinction is averaging swaps only with future positions under a fixed prefix and assigning that credit to item tokens in need-conditioned training.

3 METHOD

In this section, we present FlexRec, an RL post-training framework that trains LLM recommenders via item-level GRPO and uncertainty-aware updates under sparse feedback.

3.1 PROBLEM FORMULATION: NEED-SPECIFIC AUTOREGRESSIVE RANKING

We study need-specific ranking, where the model outputs an ordered list of candidate items optimized for a given need. We represent both context and candidate items in natural language and produce rankings via text generation, enabling LLMs to reason over heterogeneous signals (e.g., user history, item metadata, and need instructions).

Let $x \triangleq (U, C, M)$ denote the context, where U is user information (e.g., history), C is the candidate set and M is metadata/task instruction. Let $n \in \mathcal{N}$ denote the need (e.g., engagement, niche content discovery or trend promotion). Each need induces a relevance function over the same candidates, so placement credit depends on the active need.

Given candidates C, the language model generates a need-conditioned ranking:

$$y = (a_1, \ldots, a_K), \qquad a_k \in C \setminus \{a_1, \ldots, a_{k-1}\},$$

where at each step k it selects one item from the remaining candidate pool. This defines an autoregressive decision process $\pi_\theta(a_k \mid a_{<k}, x, n)$ which maps the context, the target need, and the previously selected items $a_{<k}$ to a decision over the remaining candidates.

A need-specific objective evaluates ranking y and decomposes over items:

$$R_n(y; x) \;=\; \sum_{k=1}^{K} d_k r_n(a_k, k; x), \forall n$$

where $r_n(a_k, k; x)$ is the gain of placing item a_k at rank k under context x and need n while d_k is the position-dependant coefficient (e.g: $\frac{1}{log_2(k+1)}$ in NDCG). We adopt NDCG as the main training signal, following prior work (Lin et al., 2025; Zhu et al., 2025b).

3.2 Challenge of Multi-Need Recommendation

Under the formulation above, multi-need recommendation is equivalent to optimizing a family of need-specific objectives $\{R_n\}_{n\in\mathcal{N}}$ over a shared autoregressive ranking space. Because the active need changes the item utility $r_n(a_k, k; x)$, the same placement can receive different credit under different objectives. Learning from these objectives therefore presents two challenges. First, standard listwise supervision provides only the final scalar $R_n(y; x)$, collapsing the effects of all K autoregressive selections and obscuring which placements satisfy the active need. Second, when item utilities are unobserved and must be estimated from sparse user–item interactions, the reliability of these estimates can vary across contexts, items, and needs.

3.3 Swap-based Item-level Reward

We first address coarse list-level supervision with a reward that respects the prefix and remaining candidate pool of autoregressive ranking.

Properties of Autoregressive Ranking. We formalize three fundamental properties of autoregressive ranking with language models. These insights characterize the structure of the ranking problem and guide the design of effective item-level reward signals.

Insight 1: Sequential Dependence. The contribution of a_k is defined only relative to the remaining pool $\mathcal{C}_k = C \setminus \{a_1, \ldots, a_{k-1}\}$, because earlier-ranked items are already fixed and cannot be affected by the choice of a_k. Therefore, any valid credit signal for a_k should depend on how it compares to items in $\mathcal{C}_k$, rather than on its absolute contribution alone.

Insight 2: Swaps as Atomic Actions. Let a ranking be a permutation $y = (a_1, \ldots, a_K)$. For any pair of positions $1 \le k < j \le K$, define the *swap operator* $\tau_{k,j}$ that exchanges the items at ranks k and j, and denote the swapped ranking by

$$y^{(k\leftrightarrow j)} \triangleq \tau_{k,j}(y).$$

We note that any ranking output can be obtained by composing pairwise swaps. Therefore, we view a single swap as an atomic local edit of a ranking and attribute changes in listwise quality to swapping.

Insight 3: Marginal Contribution via Counterfactual Relocation. Item-level credit should be defined through counterfactual rankings in which the position of an item is changed while the rest of the list is held fixed. Using the swap operator from insight 2, the natural local counterfactual for the item at rank k is to exchange it with a lower-ranked item $j > k$, yielding $y^{(k\leftrightarrow j)}$. We therefore quantify the contribution of a_k by the change in objective:

$$\Delta_{k,j}(y; x) \triangleq R_n\Big(y^{(k\leftrightarrow j)}; x\Big) - R_n(y; x),$$

and define the item-level reward as the expected improvement under an atomic swap with candidates over the remaining positions.

Reward Design Motivated by the previous insights, we define item-level credit through counterfactual swaps. Given a predicted ranking $y = (a_1, \ldots, a_K)$ and a need-specific listwise objective $R_n(y; x)$, let $y^{(k\leftrightarrow j)}$ denote the ranking obtained by swapping positions k and j ($j > k$). For $k < K$, define the mean change caused by a swap as

$$\Delta_k(y; x) \triangleq \mathbb{E}_{j\sim\mathrm{Unif}([k+1:K])}\left[R_n\big(y^{(k\leftrightarrow j)}; x\big) - R_n(y; x)\right],$$

which compares a_k only against future items. Averaging all feasible swaps gives the exact negative expectation for this fixed rollout:

$$r_k^{\mathrm{CS}} = \frac{1}{K-k}\sum_{j=k+1}^{K}\left[R_n(y; x) - R_n\Big(y^{(k\leftrightarrow j)}; x\Big)\right], \qquad k < K. \tag{1}$$

Positive credit means that a uniformly chosen future swap decreases the objective on average. The last item has no remaining alternative, so $r_K^{\mathrm{CS}} = 0$ (theoretical analysis of the reward in Appendix I).

Time Complexity Compared with standard GRPO, FlexRec only adds the construction of swap-based item-level rewards, which introduces an $\mathcal{O}(K^2)$ overhead for K ranked items. Even with large K (10^5) that recommendation system might face in practice, this extra overhead introduce less than 0.1% extra time to the training process.

Item-Level GRPO Given item-level rewards, we compute advantages by normalizing across rollouts within each prompt group. Let g index a prompt group and let $y^{(i)} = (a_1^{(i)}, \ldots, a_K^{(i)})$ denote the i-th rollout in group g. Let $r_{g,i,k}^{\mathrm{CS}}$ be the item-level reward (Eq. (1)) computed on $y^{(i)}$. We pool all item rewards in group g as

$$\mathcal{R}_g \triangleq \{\, r_{g,i,k}^{\mathrm{CS}} \mid i = 1, \ldots, G_g,\ k = 1, \ldots, K \,\},$$

and compute group-specific statistics

$$\mu_g \triangleq \mathbb{E}_{r \sim \mathcal{R}_g}[r], \qquad \sigma_g \triangleq \sqrt{\mathrm{Var}_{r \sim \mathcal{R}_g}[r]}.$$

The item-level advantage is then

$$A_{g,i,k}^{\mathrm{item}} \;=\; \frac{r_{g,i,k}^{\mathrm{CS}} - \mu_g}{\sigma_g + \varepsilon},$$

which we assign to all tokens corresponding to item $a_k^{(i)}$ in the generated sequence. Unlike standard GRPO, which normalizes sequence-level rewards across rollouts, our normalization is performed over **all item rewards across all rollouts**. This is well-defined because our design yields comparable item-level rewards across varying prefixes and remaining candidate pools, enabling meaningful normalization across contexts. For non-item tokens such as reasoning traces or formatting tokens, we use the standard sequence-level GRPO advantage computed from the sequence-level reward. Refer to Appendix G for more implementation details and Table 7 for ablation studies.

3.4 Uncertainty-Aware GRPO

Sparse user–item interactions require RLVR methods to impute missing rewards, typically with a lightweight critic Lobel et al. (2020); He et al. (2017); Guo et al. (2017); He & Chua (2017). Because limited supervision makes these estimates noisy and unevenly reliable, we propose uncertainty-aware GRPO to prevent spurious estimates from misleading policy updates.

Concretely, we train a critic to jointly predict user–item interaction rewards and an input-dependent **predictive variance** (refer to Appendix H for architecture and training details). Given context x, candidate item a_k and active need n, the critic outputs a mean reward estimate $\hat{r}_n(a_k; x)$ and predictive variance $\hat{\sigma}_n^2(a_k; x)$. We do not seek to isolate predictor uncertainty alone: under sparse behavioral feedback, prediction error is entangled with genuine variation in user responses. The variance head models conditional variability of the engagement target. We therefore use variance as a heuristic reliability proxy to guide RL updates.

Following the sequence-level objective $R_n(y; x)$ defined above, uncertainty in the estimated item-level contributions propagates to the sequence-level reward. Ignoring cross-item covariance, we aggregate the predicted variances using the objective's position weights:

$$v_i \triangleq \sum_{k=1}^{K} d_k^2\, \hat{\sigma}_n^2(a_k^{(i)}; x),$$

where d_k is the position-aware weight at rank k. Different LLM rankings receive different variance estimation, as uncertain item-level rewards contribute different levels of uncertainty depending on their ranked positions.

Uncertainty-aware Update. Policy updates depend on an action's reward relative to a baseline; in GRPO, this baseline is the mean reward across sampled rollouts. A noisy critic can distort this comparison. For example, an item with a high but uncertain predicted reward may be placed at the top of a ranking, the resulting advantage dominating policy updates. We therefore use predicted uncertainty as a reliability signal to down-weight such rollouts, reducing their influence while preserving the signs of their advantages.

Let A_i denote the standard sequence-level advantage for rollout i, and let v_i be the estimated variance of its sequence-level reward. To down-weight high-variance and thus unreliable updates, we reweigh the advantage by inverse variance:

$$c_i \triangleq \frac{1}{v_i + \varepsilon}, \quad c_i \leftarrow \min\left(\frac{c_i}{\frac{1}{|g(i)|}\sum_{j \in g(i)} c_j}, 1\right).$$

This reweighting mitigates erroneous updates from a noisy critic: spurious, overconfident reward estimates can induce large and harmful advantages. Finally, we apply the (normalized and clipped) weight to the advantage:

$$\widetilde{A}_i = c_i A_i.$$

The same c_i scales every item-token and non-item-token advantage in a response, reflecting the reliability of that response as a whole (more details are available in Appendix H.1).

4 EXPERIMENTS

In this section, we describe the datasets used in our work and how we construct a diverse suite of needs. We then show that FlexRec is both an efficient learner when trained for a single need and a strong generalist that transfers effectively across needs, enabling a universal LLM ranker (see Appendix C for training details).

Datasets: In this work we focus on the following datasets: KuaiRec, ML-1M and ESCI.

- **KuaiRec** (Gao et al., 2022). A short-video recommendation dataset with dense user–item interactions and rich user/video metadata; to better reflect sparse implicit feedback, we uniformly subsample 10% interactions per user.
- **MovieLens-1M** (Harper & Konstan, 2015). A classic rating-based benchmark capturing long-term user preferences over movies.
- **ESCI** (Reddy et al., 2022). A product-search benchmark where the model ranks a fixed set of items relevant to a textual query.

These benchmarks cover a wide range of recommendation scenarios, spanning short-form content, long-term preferences, and query-driven ranking (Appendix A provides details of dataset construction).

4.1 NEED-SPECIFIC OBJECTIVE DESIGN

To address the lack of public benchmarks that explicitly annotate user needs in recommendation, we construct a suite of need-specific objectives on top of each dataset. These needs span realistic scenarios, from maximizing user interest, to surfacing new topics a user is likely to appreciate, to promoting items that are trending on the platform (details are reported in Appendix B).

Maximizing Interest Ranks items by user preference signals, so items the user is most likely to enjoy and engage with receive the highest scores.

Explore New Topics Prioritizes niche or underexposed items that match the user's interests but come from topics/genres the user has not recently interacted with, so items with strong interest alignment and low historical exposure are scored highest.

Trend Promotion Balances personalization with temporal popularity, so items that are both popular and relevant to the user in the recent time window (e.g., the past day) receive the highest scores.

4.2 BASELINES

We compare the following approaches to closed-set ranking:

Traditional Rankers: BERT4Rec (Sun et al., 2019) is bidirectional Transformer sequential recommender trained with an MLM (Masked Language Modeling) objective. We use the standard RecBole implementation.[1] **STAR** (Lee et al., 2025) uses LLM-based embeddings for sequential

[1] https://github.com/RUCAIBox/RecBole/

Table 1: Performance on **Maximizing Interest** across KuaiRec and MovieLens-1M. FlexRec achieves superior single-need ranking performance compared with baselines.

	KuaiRec					MovieLens-1M				
Maximizing Interest	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**
Scoring-based Rankers										
BERT4Rec (Sun et al., 2019)	0.415	0.479	0.706	0.182	0.578	0.502	0.560	0.796	0.128	0.231
STAR (Lee et al., 2025)	0.384	0.457	0.697	0.170	0.541	0.500	0.562	0.797	0.129	0.234
Raw critic	0.379	0.453	0.694	0.167	0.512	0.489	0.548	0.791	0.126	0.240
Zero-Shot LLMs										
GPT-4o	0.376	0.451	0.695	0.154	0.505	0.508	0.564	0.800	0.139	0.244
Qwen2.5-3B-Instruct	0.375	0.451	0.693	0.160	0.515	0.497	0.553	0.794	0.131	0.245
Post-Trained LLMs (Sparse)										
Qwen2.5-3B-Instruct										
+ TALLRec (Bao et al., 2023)	0.507	0.558	0.730	0.264	0.729	0.569	0.609	0.821	0.184	0.330
+ Rec-R1 (Lin et al., 2025)	0.391	0.466	0.700	0.174	0.545	0.554	0.600	0.817	0.170	0.325
+ ConvRec-R1 (Zhu et al., 2025b)	0.389	0.462	0.698	0.174	0.520	0.492	0.549	0.792	0.131	0.245
+ ReasonRank (Liu et al., 2025)	0.385	0.458	0.697	0.172	0.546	0.499	0.559	0.796	0.132	0.239
+ BLADE-N (Chen et al., 2026)	0.383	0.455	0.696	0.169	0.541	0.496	0.556	0.795	0.131	0.238
+ FlexRec	**0.597**	**0.635**	**0.788**	**0.335**	**0.840**	**0.615**	**0.665**	**0.844**	**0.235**	**0.368**
Δ vs Qwen2.5-3B-Instruct	+59.2%	+40.8%	+13.7%	+109.4%	+63.1%	+23.7%	+20.3%	+6.3%	+79.4%	+50.2%

recommendation and achieves strong performance without task-specific training. We follow the official implementation in the original paper. **Raw critic** ranks candidates directly by the engagement critic's predicted mean, evaluating the reward predictor itself as a ranker (Appendix H).

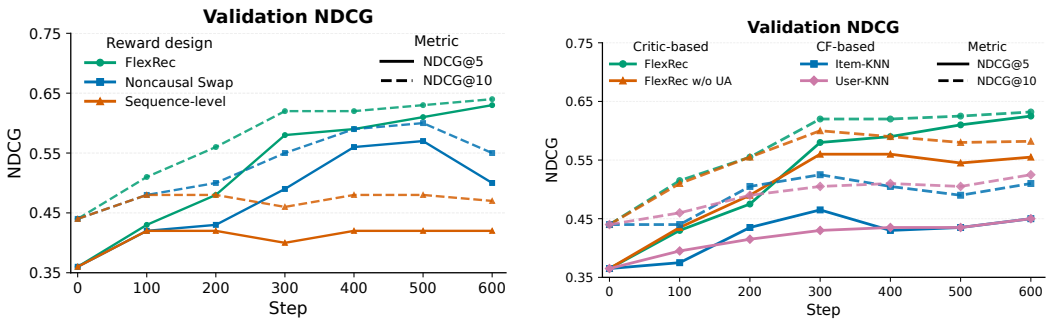
Post-trained LLMs: TALLRec (Bao et al., 2023) applies LoRA fine-tuning using supervised recommendation data; we use the optimal ranked list as the target. **Rec-R1** (Lin et al., 2025), **ReasonRank** (Liu et al., 2025), and **BLADE-N** (Chen et al., 2026) use GRPO with sequence-level rewards: NDCG, a combination of NDCG/recall/rank-biased overlap, and a Bayesian transform of NDCG rewards, respectively. **ConvRec-R1** (Zhu et al., 2025b) extends Rec-R1 with rank-based GRPO for closed-set conversational recommendation, grouping identical rank predictions across rollouts. We adopt 25 warm-up SFT steps for all post-training methods.

4.3 Results

Table 2: Need generalization results (train: **Maximizing Interest**, test: **Explore New Topics**) indicate that FlexRec transfers strong ranking abilities effectively across needs.

	KuaiRec					MovieLens-1M				
Models	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**
Traditional Rankers										
BERT4Rec (Sun et al., 2019)	0.720	**0.784**	0.900	**0.165**	**0.480**	0.731	0.800	0.922	0.144	0.237
STAR (Lee et al., 2025)	0.698	0.751	0.909	0.111	0.318	0.719	0.791	0.919	0.098	0.158
Zero-Shot LLMs										
GPT-4o	0.637	0.703	0.892	0.059	0.142	0.716	0.791	0.918	0.111	0.148
Qwen2.5-3B-Instruct	0.706	0.737	0.911	0.140	0.377	0.729	0.798	0.922	0.133	0.216
Post-Trained LLMs (trained on Max-Interest)										
Qwen2.5-3B-Instruct										
+ TALLRec (Bao et al., 2023)	0.700	0.748	0.901	0.147	0.399	0.729	0.798	0.921	0.142	0.205
+ Rec-R1 (Lin et al., 2025)	0.730	0.755	0.915	0.160	0.429	0.726	0.796	0.921	0.126	0.201
+ ConvRec-R1 (Zhu et al., 2025b)	0.717	0.767	0.915	0.163	0.429	0.729	0.798	0.921	0.136	0.216
+ ReasonRank (Liu et al., 2025)	0.706	0.756	0.911	0.127	0.366	0.726	0.797	0.921	0.128	0.205
+ BLADE-N (Chen et al., 2026)	0.700	0.753	0.910	0.114	0.330	0.730	0.798	0.922	0.135	0.220
+ FlexRec	**0.733**	0.776	**0.919**	**0.165**	0.440	**0.748**	**0.844**	**0.944**	**0.165**	**0.267**
Δ vs Qwen2.5-3B-Instruct	+3.8%	+5.3%	+0.9%	+17.9%	+16.7%	+2.6%	+5.8%	+2.4%	+24.1%	+23.6%

FlexRec is an Effective Single-Need Recommender Table 1 and 9 reports performance on the **Maximizing Interest** need for KuaiRec, MovieLens-1M and ESCI. We compare against a broad set of baselines, including traditional rankers, zero-shot prompted LLMs, and RL post-training methods



(a) Reward design.

(b) Critic and uncertainty-aware update.

Figure 2: Validation NDCG during FlexRec training on KuaiRec. The causal swap-based item-level reward improves over list-level reward and noncausal swap rewards (left); the critic with uncertainty-aware scaling is stronger and more stable than CF completion or critic-only training (right).

for LLM rankers. FlexRec consistently outperforms these alternatives by leveraging fine-grained item-level rewards and stabilizing training with uncertainty-aware updates, enabling efficient adaptation of LLM rankers to user needs (Appendix M shows results on larger model sizes). Moreover, FlexRec outperforms direct ranking by the raw critic's predicted scores, indicating that the learned policy goes beyond simply reproducing the critic's ordering.

FlexRec Generalizes Across Needs Tables 2 and 6 show zero-shot generalization performance of baseline models and post-trained LLMs. FlexRec showcases need transfer: trained only on **Maximizing Interest**, our method consistently improves zero-shot performance over the backbone on **Explore New Topics** and **Trend Promotion**. This suggests that fine-grained, need-aware credit assignment helps the model learn reusable ranking principles beyond a single objective. Gains vary by metric: Recall@5 and MRR@5 improve on both new-topic datasets and MovieLens-1M trending, but decrease on KuaiRec trending. Notably, GPT-4o is competitive on MovieLens-1M under **Trend Promotion**, but FlexRec still achieves the best overall generalization among open-source post-trained counterparts across datasets and settings.

FlexRec is a Universal Multi-Need Ranker. Figure 3 and Appendix J further show that one jointly trained FlexRec model can serve all three needs and switch its ranking behavior through the need instruction. FlexRec as a principled training framework provides an effectively way to align large language models to diverse user needs.



Figure 3: Joint need-conditioned training on KuaiRec improves performance across all three needs.

4.4 Ablation Studies

Item-level Reward Formulation Table 7 shows the performance of different reward constructions under both GRPO and PPO. *Independent contribution* uses an item's individual contribution as the reward and is independent of the current candidate pool. *Swap-based contribution (prefix-agnostic)* computes rewards via counterfactual swaps, but allows swaps with both previously fixed prefix items and future unpredicted items, rather than restricting comparisons to the remaining candidate pool at the current autoregressive step. FlexRec further normalizes item-level rewards across all ranks and rollouts for each prompt, which share the same candidate set and need-specific objective. Our ablation shows this choice matters: normalizing separately at each rank reduces performance. Across RL algorithms and reward formulations, our swap-based item-level reward with pooled normalization yields substantial gains, offering an effective need-aware reward for training multi-need LLM rankers.

Uncertainty-aware Update Table 8 provides ablation studies on two design choices made in FlexRec: using a learned critic to provide reward signals, and jointly predicting uncertainty to modulate training updates. Traditional recommendation systems use collaborative filtering signals (Item/User KNN) to impute potential interactions. However, they are not optimized to serve as training rewards and provide no estimation of uncertainty, which can yield noisy supervision and brittle updates. In contrast, the learned critic (refer to Appendix H for details) provides a stronger reward signal, as evidenced by the large gain in performance compared with CF-based methods. Moreover, using its predicted uncertainty to weight training updates improves performance compared with constant or shuffled weights, supporting the benefit of precisely downweighting uncertain rollouts.

4.5 Discussion

Adaptive Reasoning Across Needs. Qualitative examples in Appendix J.2 show that FlexRec adapts its rationales to the active need. For Explore New Topics, it identifies historically underexposed genres/topics; for Trend Promotion, it uses recent popularity signals to prioritize timely yet relevant items. These need-conditioned rationales suggest that FlexRec learns not only better rankings but also more interpretable recommendation strategies.

Impact of Item-level Reward. The training dynamics in figure 2a show that the swap-based item-level reward improves over sequence-level GRPO and sequential-dependence-agnostic swaps. This supports our core design: in autoregressive ranking, useful credit assignment should evaluate each item relative to the remaining candidate pool under the active need, rather than assigning a coarse sequence-level signal or allowing counterfactuals that violate the fixed-prefix structure.

Impact of Uncertainty-aware Updates. Figure 2b further shows that critic-based rewards outperform KNN-style collaborative reward completion, while removing uncertainty-aware scaling leads to less stable later-stage training. This suggests that the learned critic provides useful reward completion under sparse feedback, and that variance-based downweighting helps prevent unreliable estimates from dominating RL updates.

5 Conclusion

FlexRec adapts an LLM ranker to explicit recommendation needs using swaps within the remaining candidate pool and critic-based reward completion. The strongest evidence is improved interest ranking on both datasets and a gain over sequence-level GRPO on dense KuaiRec, where reward completion is less central. These results support the combined training recipe. Transfer to constructed novelty and trending objectives is mixed, while joint training demonstrates support for the three evaluated needs. These offline proxy tasks motivate further evaluation of instruction dependence, robustness, and serving cost in practical settings.

AI Use Statement

The original manuscript was written manually without LLM assistance. The author used AI to assist with minor Latex formatting and paraphrasing.

REFERENCES

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*, RecSys '23, pp. 1007–1014. ACM, September 2023. doi: 10.1145/3604915.3608857. URL http://dx.doi.org/10.1145/3604915.3608857.

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

Christopher J. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with non-smooth cost functions. In *Advances in Neural Information Processing Systems*, volume 19, 2006. URL https://papers.neurips.cc/paper_files/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html.

Hao Chen, Yuanchen Bei, Qijie Shen, Yue Xu, Sheng Zhou, Wenbing Huang, Feiran Huang, Senzhang Wang, and Xiao Huang. Macro graph neural networks for online billion-scale recommender systems. In *Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 3598–3608, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645517. URL https://doi.org/10.1145/3589334.3645517.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Ruijun Chen, Chongming Gao, Jiawei Chen, Weiqin Yang, and Xiangnan He. Beyond static best-of-N: Bayesian list-wise alignment for LLM-based recommendation, 2026. URL https://arxiv.org/abs/2605.04559.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Sahraoui Dhelim, Nyothiri Aung, Mohammed Amine Bouras, Huansheng Ning, and Erik Cambria. A survey on personality-aware recommendation systems. *Artificial Intelligence Review*, 55(3): 2409–2454, 2022.

Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. Kuairec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, pp. 540–550, 2022. doi: 10.1145/3511808.3557220. URL https://doi.org/10.1145/3511808.3557220.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 3816–3830, 2021.

Yingqiang Ge, Shuchang Liu, Zuohui Fu, Juntao Tan, Zelong Li, Shuyuan Xu, Yunqi Li, Yikun Xian, and Yongfeng Zhang. A survey on trustworthy recommender systems, 2024. URL https://arxiv.org/abs/2207.12515.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. *ArXiv*, abs/1703.04247, 2017. URL https://api.semanticscholar.org/CorpusID:970388.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), 2015. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics, 2017. URL https://arxiv.org/abs/1708.05027.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pp. 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569.

Dietmar Jannach. A survey on multi-objective recommender systems. *Frontiers in Big Data*, 6:1157899, 2023. doi: 10.3389/fdata.2023.1157899. URL https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2023.1157899/full.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Dong-Ho Lee, Adam Kraft, Long Jin, Nikhil Mehta, Taibai Xu, Lichan Hong, Ed H. Chi, and Xinyang Yi. Star: A simple training-free approach for recommendations using large language models, 2025. URL https://arxiv.org/abs/2410.16458.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. Gecko: Versatile text embeddings distilled from large language models, 2024. URL https://arxiv.org/abs/2403.20327.

Jiacheng Lin, Tian Wang, and Kun Qian. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.24289.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. ReasonRank: Empowering passage ranking with strong reasoning ability, 2025. URL https://arxiv.org/abs/2508.07050.

Sam Lobel, Chunyuan Li, Jianfeng Gao, and Lawrence Carin. Towards amortized ranking-critical training for collaborative filtering, 2020. URL https://arxiv.org/abs/1906.04281.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 583–612, 2024.

Qiyao Ma, Menglin Yang, Mingxuan Ju, Tong Zhao, Neil Shah, and Rex Ying. Breaking information cocoons: A hyperbolic graph-llm framework for exploration and exploitation in recommender systems. *arXiv preprint arXiv:2411.13865*, 2024.

Qiyao Ma, Yunsheng Shi, Hongtao Tian, Chao Wang, Weiming Chang, and Ting Yao. From faithfulness to correctness: Generative reward models that think critically. *arXiv preprint arXiv:2509.25409*, 2025.

Phong Nguyen, John Dines, and Jan Krasnodebski. A multi-objective learning to re-rank approach to optimize online marketplaces for multiple stakeholders. *arXiv preprint arXiv:1708.00651*, 2017. URL https://arxiv.org/abs/1708.00651.

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

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. Shopping queries dataset: A large-scale ESCI benchmark for improving product search, 2022. URL https://arxiv.org/abs/2206.06588.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks, 2022. URL https://arxiv.org/abs/2203.09168.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

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, 2019.

Xiaolei Wang, Chunxuan Xia, Junyi Li, Fanzhe Meng, Lei Huang, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Search-based interaction for conversation recommendation via generative reward model based simulated user, 2025a. URL https://arxiv.org/abs/2504.20458.

Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey, 2025b. URL https://arxiv.org/abs/2503.12605.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. Eager: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3245–3254, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jiaxing Wu, Lin Ning, Luyang Liu, Harrison Lee, Neo Wu, Chao Wang, Sushant Prakash, Shawn O'Banion, Bradley Green, and Jun Xie. Rlpf: Reinforcement learning from prediction feedback for user summarization with llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25488–25496, 2025.

Wen Wu, Li Chen, and Yu Zhao. Personalizing recommendation diversity based on user personality. *User Modeling and User-Adapted Interaction*, 28(3):237–276, 2018.

Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*, 2023a.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *Advances in Neural Information Processing Systems*, 36:24247–24261, 2023b.

Zhisheng Yang, Xiaofei Xu, Ke Deng, and Li Li. Enhance large language models as recommendation systems with collaborative filtering, 2025. URL https://arxiv.org/abs/2510.15647.

Gangyi Zhang. User-centric conversational recommendation: Adapting the need of user with large language models. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 1349–1354, 2023.

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

Dingwei Zhu, Shihan Dou, Zhiheng Xi, Senjie Jin, Guoqiang Zhang, Jiazheng Zhang, Junjie Ye, Mingxu Chai, Enyu Zhou, Ming Zhang, Caishuang Huang, Yunke Zhang, Yuran Wang, and Tao Gui. Vrpo: Rethinking value modeling for robust rl training under noisy supervision, 2025a. URL https://arxiv.org/abs/2508.03058.

Yaochen Zhu, Harald Steck, Dawen Liang, Yinhan He, Vito Ostuni, Jundong Li, and Nathan Kallus. Rank-grpo: Training llm-based conversational recommender systems with reinforcement learning, 2025b. URL https://arxiv.org/abs/2510.20150.

A Dataset Details

Data Subsampling The construction uses 5,000 training instances per dataset and need. The supplied held-out evaluation files contain 1,000 queries for each recommendation dataset/need pair except KuaiRec trending, which contains 315. Each query supplies 30 candidates and a fixed relevance vector. Appendix F specifies the target fields and scoring rules; Appendix N describes the additional methods and the separate two-stage diagnostic.

KuaiRec (short-video recommendation). KuaiRec is a fully observed short-video recommendation dataset with dense user–item interactions and rich user/video metadata. Since the raw data is near-dense, we induce sparsity to better match realistic implicit-feedback regimes by uniformly subsampling 10% of interactions per user. We use separate temporal splits for training, validation, and testing. Each ranking instance conditions on the user's most recent $H{=}10$ historical items available before its prediction point and asks the model to rank a candidate set of $C{=}30$ items; the candidate construction procedure is described in Appendix D. The dataset is released under *Creative Commons Attribution Share Alike 4.0 International* license.

MovieLens (rating-based recommendation). MovieLens is a classic benchmark for rating-based recommendation that captures long-term user preferences over movies. We use the same dataset splitting and processing protocol as used in KuaiRec. MovieLens-1M is distributed by the GroupLens Research Group under a research-use license that permits research use with attribution and prohibits redistribution or commercial use without separate permission.

ESCI (product search). ESCI represents product search, where the goal is to retrieve items most relevant to a textual query rather than a user's interaction sequence. We follow the dataset's publicly released train–test split for training and evaluation.[2]. The dataset is released under *Apache License 2.0* license.

B Need Construction

B.1 Maximizing Interest

For KuaiRec, following prior work (Gao et al., 2022; Yang et al., 2023b; Chen et al., 2024), we use the watch ratio—defined as the ratio between the total watch time and the video length—as a proxy for user interest. Watch ratio provides a continuous, fine-grained signal that reflects both user engagement and early abandonment behavior, and is commonly adopted as an implicit relevance measure in short-video recommendation settings. We directly use the watch ratio value as the relevance score when evaluating item utility.

For ML-1M, explicit user feedback is available in the form of discrete ratings ranging from 1 to 5. We treat ratings as relevance labels. The NDCG gain transform is $2^{\text{rating}} - 1$, applied once by the scorer to the raw rating, rather than treating an already exponentiated gain as a new relevance label.

The watch-ratio instruction below describes the KuaiRec task. The MovieLens-1M interest evaluation uses the rating-specific instruction shown next.

Need: Maximizing Interest (KuaiRec)

Rank them to maximize the user's expected watch ratio (likelihood of long/complete viewing) based on their history—because higher watch ratio signals stronger engagement and improves retention. Also keep in mind to recommend items that strongly align with the user's known interests.

[2] https://github.com/amazon-science/esci-data

Need: Maximizing Interest (MovieLens-1M evaluation)

You will receive a list of candidate movies, each with key 'movie_id'. Rank them to maximize the user's expected rating (1–5 stars), leveraging the user's past ratings and topical preferences. Consider signals such as preferred genres and recency effects in the user's history. Balance strong alignment with the user's known tastes and exploration into genres that the user may like. Your output must contain exactly the same set of candidates.

B.2 Explore New Topics

Niche discovery aims to recommend items that expose users to previously unseen topical items while remaining relevant to their interests.

In KuaiRec, we implement niche discovery using video topic tags. For a given user history, we identify the set of topics the user has previously interacted with, and label a candidate video as *niche* if its topic does not appear in this history. Candidates without valid topic annotations are treated as non-niche.

To construct a ground-truth ranking that balances relevance and novelty, we start from a base relevance score obtained via embedding-based retrieval. We then apply a multiplicative niche bonus to candidates labeled as niche,

$$\text{score}_{\text{niche}}(v) = \text{score}_{\text{base}}(v)\big(1 + \alpha\, \mathbf{1}\{v \text{ is niche}\}\big),$$

where $\alpha = 0.5$, giving niche candidates a $1.5\times$ multiplier in the supplied evaluation targets. The resulting scores induce a ranking that prioritizes novel-topic items while preserving relevance among both niche and non-niche candidates. This objective combines historically underexposed topics with similarity-based relevance, providing a controlled test of novelty-sensitive ranking.

For ML-1M, niche discovery is defined analogously by identifying movies whose genres have not appeared in the user's interaction history.

We use the following instruction across datasets:

Need: Niche Discovery

Keep in mind to recommend not only items aligned with the user's interests but also items that are niche (topics the user hasn't interacted with), but that are plausibly relevant and could enrich their experience.

B.3 Trend Promotion

Trending recommendation aims to surface items that are both relevant to the user and currently popular across the platform, capturing short-term collective interest and temporal dynamics.

In KuaiRec, we construct trending signals using recent interaction frequency. For each candidate video, we compute its trend score as the number of user interactions occurring within a sliding **one-day** window prior to the query time.

To construct a ground-truth ranking that balances personalization and global popularity, we combine semantic relevance and temporal trend signals. Candidate videos are first retrieved using embedding-based similarity to the user's recent interaction history. Similarity and activity are independently normalized within each candidate pool. For activity count c_v, the activity input is c_v on KuaiRec and $\log(1 + c_v)$ on MovieLens-1M. For either input vector q, we use

$$\text{norm}(q_v) = \frac{q_v - \min_{u \in C} q_u}{\max_{u \in C} q_u - \min_{u \in C} q_u + 10^{-8}}.$$

A constant input vector therefore maps to zero. Let sim_{norm} and $\text{trend}_{\text{norm}}$ denote these normalized similarity and activity inputs, respectively. The final relevance score is computed as the combination,

$$\text{score}_{\text{trend}}(v) = \alpha \cdot \text{sim}_{\text{norm}}(v) + (1 - \alpha) \cdot \text{trend}_{\text{norm}}(v), \tag{2}$$

where $\alpha \in (0, 1)$ controls the trade-off between personalized relevance and global trending popularity (we use $\alpha = 0.7$ across experiments). The resulting scores induce a ranking that prioritizes items that are both timely and aligned with the user's interests.

For ML-1M, trending recommendation is defined analogously using recent 1-day activity.

We use the following instruction across datasets:

Need: Trending Promotion

Keep in mind to recommend not only items aligned with the user's interests but also items that are popular in the last 24 hours (provided as the number of watches in the last day), as long as they are relevant to the user.

B.4 Product Search

Amazon's official ESCI dataset annotates each (query, item) pair into four relevance categories—**Exact (E)**, satisfying all query constraints; **Substitute (S)**, partially matching but usable as a replacement; **Complement (C)**, not matching the query itself but useful alongside an exact item; and **Irrelevant (I)**. We follow these categories and the official graded scoring scheme, mapping them to $\{1.0, 0.1, 0.01, 0.0\}$ for $\{E, S, C, I\}$, respectively.

We use the following instruction for product search:

Need: Product Search

Your task is to rerank these candidate products by how well they match the user's search query. Consider how closely each product's title, description, and attributes align with the query intent. Give higher ranks to products that are exact or highly relevant matches (e.g., correct type, brand, or specifications). Lower the rank of products that are complementary or unrelated to the query.

B.5 Data Subsampling

Dataset sizes are specified in Appendix A.

C RL Training Details

We perform reinforcement learning (RL) post-training on 4× NVIDIA A100 GPUs using the Verl [3] library. Unless otherwise specified, all experiments use full-parameter training with learning rate 1×10^{-6}.

To stabilize optimization and constrain policy drift, we include a KL regularization term against a reference policy with KL loss coefficient 0.01. We additionally apply entropy regularization to encourage exploration and avoid premature collapse, using entropy coefficient 0.005.

For FlexRec rollout generation, we use **vLLM** with a group size of $G = 8$ responses per prompt, temperature 0.5, and top-$p = 0.9$. The training objective is NDCG@30 over all 30 predicted items. We select the ranker checkpoint with the highest validation NDCG@30; the critic has its own validation-MSE selection rule in Appendix H. Additional-baseline settings are reported separately in Appendix N. We do not report an end-to-end throughput or latency comparison.

[3] https://github.com/verl-project/verl

Protocol at a glance. The following settings summarize FlexRec on KuaiRec and MovieLens-1M; the adapted ReasonRank and BLADE-N baselines retain their separately reported configurations in Appendix N.

Component	FlexRec setting
Ranking instance	10 historical items; 30 candidates; explicit need instruction
Data partition	Temporal training, validation and test partitions (Appendix A)
Actor training	Full-parameter updates; learning rate 10^{-6}; 4 NVIDIA A100 GPUs
Rollout sampling	vLLM; $G = 8$; temperature 0.5; top-$p = 0.9$
Ranking objective	NDCG@30 over the complete candidate pool
Token credit	Pooled item advantages on item tokens; sequence GRPO on other tokens; response coefficient on both
Regularization	KL coefficient 0.01; entropy coefficient 0.005
Actor selection	Highest validation NDCG@30
Critic selection	Lowest validation MSE across 30 training epochs
Evaluation	Fixed held-out relevance vectors and per-query macro-averaging (Appendix F)

Matched dense-KuaiRec comparison. For the dense-KuaiRec comparison in Table 10, Rec-R1, ConvRec-R1, and FlexRec use identical random-seed settings, training data and partitions, initial model checkpoint, feedback source, training budget, and evaluation protocol. This comparison covers sequence-level GRPO (Rec-R1), rank-level credit (ConvRec-R1), and the proposed FlexRec method. The separately configured ReasonRank and BLADE-N adaptations in Appendix N are not part of this dense comparison.

D Candidate Generation

We construct the candidate set using a fixed semantic retriever following STAR (Lee et al., 2025), a state-of-the-art training-free retrieval framework based on large language model embeddings. Each item is first converted into a natural-language description using its available metadata, and embedded using the `text-embedding-004` model from the Gemini embedding family (Lee et al., 2024).

Given a user interaction history of length H, candidate items are retrieved by cosine similarity in the embedding space. To account for temporal preference dynamics, we form a query embedding as a temporally discounted average of historical item embeddings, optionally weighted by interaction strength (e.g., ratings or watch ratios), following the original STAR formulation. The candidate set consists of the top-K most similar eligible items, excluding the user history available at the prediction point. The MovieLens-1M interest test uses fixed 30-item pools with observed ratings for every candidate, as verified in Appendix F.

The retriever is kept frozen throughout training and evaluation. This setup reflects realistic recommendation pipelines, where candidate generation is typically performed by a pre-trained, independently maintained retrieval system, while downstream ranking models are optimized on a fixed candidate pool.

E Item Representation

In this section we detail the relevant information we give to the models. We give the same set of metadata for embedding models in retrieval as well as for LLMs in reranking.

Videos (KuaiRec). For short videos, we leverage rich categorical and item annotations provided by the dataset. Each video is converted into a textual description composed of its caption, hierarchical category path, and optional tag information:

```
Video caption: <caption>
Category path: <level_1> > <level_2> > <level_3>
Video tag: <tag>
```

Movies (MovieLens-1M). For movies, we construct a concise textual description consisting of the movie title, release year, and genre information:

```
Movie Title: <title>
Movie Year: <year>
Movie genres: <genre_1>, <genre_2>, . . .
```

ESCI For the ESCI product search dataset, each product is represented using core catalog metadata commonly available in real-world e-commerce systems. Specifically, we convert each item into a textual description by concatenating its product title, bullet-point description, brand, and color attributes:

```
Product title: <product_title>
Product description: <product_bullet_point>
Brand: <product_brand>
Color: <product_color>
```

This representation captures both functional semantics (via title and description) and distinguishing attributes (via brand and color), which are critical signals for product relevance in search scenarios. All fields are expressed in natural language and fused into a single text sequence prior to embedding, enabling a unified semantic retrieval interface consistent with other datasets.

F Evaluation Details

Observed MovieLens test targets. The MovieLens-1M interest test contains 1,000 queries and 30,000 candidate–rating pairs. Every candidate has an observed rating: all 30,000 user–movie IDs and target grades match the official MovieLens-1M rating records (Harper & Konstan, 2015). The 10,000 history entries also match those records, and no user–movie pair appears in both the history and candidate sets of this interest test. Thus the interest-ranking metrics are scored against recorded user ratings. Critic-based completion supplies training rewards for missing feedback; the test grades themselves are fully observed.

Ranking metrics. For KuaiRec and MovieLens-1M, NDCG uses gain $2^z - 1$, applied once to relevance z, and discount $1/\log_2(k+1)$. The denominator is the ideal DCG of the same 30-candidate pool at the reported cutoff. Interest uses watch ratios on KuaiRec and raw movie ratings on MovieLens-1M; new-topic evaluation uses the constructed novelty scores, and trending uses the continuous trend scores. These fixed relevance vectors accompany each held-out query and are not regenerated during scoring.

Recall@5 divides the number of positives in the first five positions by the number in the entire candidate pool. MRR@5 is the reciprocal rank of the first positive within five positions, or zero if none appears. Positives are defined by watch ratio > 1 for KuaiRec interest, rating $= 5$ for MovieLens-1M interest, novelty score > 1 for both new-topic tasks, and trend score > 0.5 for both trending tasks. Metrics are macro-averaged over all queries; zero-positive queries contribute zero Recall/MRR. In MovieLens-1M interest, 244 of the 1,000 queries have no rating-5 candidate. The positive threshold affects Recall/MRR; NDCG always uses the full graded relevance. ESCI additionally reports Precision@5. Appendix B defines its relevance grades, and Appendix N records the additional-baseline settings.

G Sequence-Level Advantages for Non-Item Tokens

Not all tokens in a generated sequence correspond to item predictions. Reasoning, formatting, and control tokens, including delimiters and EOS, belong to the non-item category. They receive the sequence-level reward and its standard group-normalized GRPO advantage.

Let g index a prompt group and let $R_{g,i}$ denote the sequence-level reward of rollout i in group g. We compute group statistics

$$\mu_g^{\text{seq}} \triangleq \mathbb{E}_i[R_{g,i}], \qquad \sigma_g^{\text{seq}} \triangleq \sqrt{\text{Var}_i[R_{g,i}]},$$

and define the sequence-level advantage

$$A_{g,i}^{\text{seq}} = \frac{R_{g,i} - \mu_g^{\text{seq}}}{\sigma_g^{\text{seq}} + \varepsilon}.$$

We assign $A_{g,i}^{\text{seq}}$ to all non-item tokens in rollout i.

The response-reliability weight and item-token assignment are combined with these sequence-level advantages in Appendix H.1.

H Interaction Critic and Uncertainty Estimation

Table 3: Critic prediction diagnostics for watch ratio (KuaiRec) and rating (MovieLens-1M). Pearson correlations compare predicted means with engagement labels (ρ_μ), and predicted variances with squared errors (ρ_{σ^2}). $^*p < 0.05$. Correlation measures association, not variance calibration or the effectiveness of response weighting.

Dataset	MSE ↓	MAE ↓	ρ_μ ↑	ρ_{σ^2} ↑
KuaiRec	0.328	0.400	0.374^*	0.300^*
MovieLens-1M	0.317	0.271	0.510^*	0.132^*

Task Definition and Challenges Given a user's interaction context which consists of historical items and observed engagement signals, we aim to predict the user's potential interaction with a candidate item. This prediction task is inherently challenging, subsuming the reranking problem considered in this work. Accordingly, the goal of the critic is not to produce perfectly accurate engagement predictions, but rather to extract and provide useful learning signals that can be exploited during the reinforcement learning stage.

In recommendation settings, such engagement signals (e.g., watch ratio or dwell time) exhibit two properties. First, they are *long-tailed*, with a small fraction of interactions accounting for a disproportionate share of total engagement. Second, they are *heteroscedastic*: the intrinsic noise of user feedback varies substantially across contexts, users, and items. As a result, both the expected engagement level and the uncertainty of this estimate depend on the input context.

Critic Architecture We employ a two-tower MLP-based critic (see 4 for details) that encodes user history and candidate items into a shared embedding space, followed by a lightweight interaction network. The critic outputs both a predicted mean engagement $\hat{\mu}(X)$ and an input-dependent variance $\sigma^2(X)$. The architecture is intentionally simple, as the critic serves as a training signal rather than a primary ranking model.

Table 4: Architecture of the critic model used to estimate engagement mean and uncertainty. where $\hat{u}$ represents (learnt) user embedding and $\hat{c}$ represents (fixed) candidate item embedding.

Component	**Specification**
User history encoder	MLP with ReLU activations
Candidate encoder	Fixed item embeddings
Hidden dimension	256
Interaction features	$\hat{u}, \hat{c}, \hat{u} \odot \hat{c}, \hat{u}^\top \hat{c}$
Output heads	Mean head + log-variance head
Normalization	LayerNorm
Dropout	0.05

Variance-Weighted Gaussian Likelihood To model heteroscedastic engagement noise, we use recent advances in heteroscedastic regression using neural networks (Seitzer et al., 2022). Let Y denote the observed engagement signal and X the input context. The critic predicts a mean $\hat{\mu}(X)$ and variance $\sigma^2(X)$. We minimize the following loss:

$$\mathcal{L}_{\beta\text{-NLL}} = \mathbb{E}_{(X,Y)}\left[\text{sg}\left[(\sigma^2(X))^\beta\right]\left(\frac{1}{2}\log\sigma^2(X) + \frac{(Y-\hat{\mu}(X))^2}{2\sigma^2(X)}\right)\right]. \tag{3}$$

Here sg denotes stop-gradient: its forward value is unchanged, but its derivative is zero. We use $\beta = 1$. Only the outer variance multiplier is detached; gradients still pass through the variance in the

Gaussian NLL and through the mean prediction. This is the gradient convention of β-NLL (Seitzer et al., 2022). Writing $s = \sigma^2(X)$ and $e = Y - \hat{\mu}(X)$, the per-example derivative with respect to s is $\frac{1}{2}(1 - e^2/s)$ at $\beta = 1$. Differentiating the outer multiplier instead would give $\frac{1}{2}(\log s + 1)$, independent of the residual.

H.1 From Critic Signals to Policy Updates

This appendix connects need-specific relevance scores and critic variances to token-level policy updates. It specifies the ranking reward, response weighting, and item/non-item advantage assignment.

Full-list ranking reward. The critic provides estimates for missing user–item feedback. Given the resulting need-specific relevance scores $s_n(a; x)$, the listwise training reward is NDCG@30:

$$R_n(y; x) = \frac{1}{Z_n(x)} \sum_{k=1}^{30} d_k \, g_n(a_k; x), \qquad d_k = \frac{1}{\log_2(k+1)}, \quad g_n(a; x) = 2^{s_n(a;x)} - 1,$$

where $Z_n(x)$ is IDCG@30 for the same candidate set. All $K = 30$ predicted items enter the objective. This listwise reward supplies the sequence-level credit and the counterfactual swap differences used for item-level credit.

Position-weighted response uncertainty. We aggregate the critic variances with squared NDCG position discounts:

$$v_{g,i} = \sum_{k=1}^{30} d_k^2 \, \hat{\sigma}_n^2(a_k^{(i)}; x_g), \qquad u_{g,i} = \frac{1}{v_{g,i} + \varepsilon}, \qquad c_{g,i} = \min\left(\frac{u_{g,i}}{G^{-1} \sum_{j=1}^{G} u_{g,j}}, 1 \right).$$

The ordering matters because the same item variance is paired with a different squared discount at a different rank. Equal aggregate uncertainty within a group still gives $c_{g,i} = 1$ for every response.

Squared discounts are the appropriate coefficients when propagating variance through a weighted sum. Specifically, for uncertain gains L_k with variances s_k^2, fixed IDCG Z, and ignored cross-item covariance,

$$\mathrm{Var}\left[Z^{-1} \sum_k d_k L_k \right] \approx Z^{-2} \sum_k d_k^2 s_k^2.$$

Our aggregate $v_{g,i}$ is used as a response-reliability score. If the critic variance is on the engagement scale, it is not automatically the variance of the nonlinear gain $2^{s_n} - 1$; exact NDCG variance would additionally require gain-scale uncertainty and treatment of the normalizer. Squared position weights alone do not establish calibration.

Response-weight controls. Table 8 compares the proposed normalized, clipped coefficient $c_{g,i}$ with no weighting, a constant coefficient $\bar{c}_g = G^{-1} \sum_i c_{g,i}$ shared by all responses to a prompt, and a random within-prompt permutation of the final coefficients. The latter two controls preserve the mean coefficient, while shuffling also preserves its distribution. They change the coefficient applied to all token advantages while retaining the critic-based rewards and other training settings. The constant coefficient is applied after normalization and clipping, without renormalizing it to one. These controls test response-specific allocation rather than merely reducing the average policy-surrogate scale.

Using the unnormalized inverse-variance coefficient $1/(v_{g,i} + \varepsilon)$ directly to scale advantages led to unstable policy training in our runs. This motivates controlling its numerical scale through the proposed group normalization and upper clipping. This is a qualitative optimization observation; the ranking comparisons in Table 8 evaluate the resulting weighting rule and its controls.

Terminal item and non-item tokens. The final item has no feasible future swap, so its raw reward is $r_{g,i,30}^{\mathrm{CS}} = 0$. It remains in the pooled item-reward statistics and is not gradient-masked:

$$A_{g,i,30}^{\mathrm{item}} = \frac{-\mu_g}{\sigma_g + \varepsilon}.$$

This advantage need not be zero. Reasoning, delimiters, and EOS instead receive the sequence-level GRPO advantage from the NDCG@30 reward (Appendix G).

Combining credit and response reliability. Let $\mathcal{T}_{g,i,k}$ contain the generated token positions representing item a_k. The same response-level reliability weight $c_{g,i}$ scales both forms of credit:

$$\widetilde{A}_{g,i,t} = c_{g,i} \begin{cases} A^{\text{item}}_{g,i,k}, & t \in \mathcal{T}_{g,i,k}, \\ A^{\text{seq}}_{g,i}, & t \text{ is a non-item token.} \end{cases}$$

Thus uncertainty changes the strength of the whole response's update, while the item/sequence assignment determines token-level credit. Writing the generated token as $z_{g,i,t}$, define

$$\rho_{g,i,t}(\theta) = \frac{\pi_\theta(z_{g,i,t} \mid z_{g,i,<t}, x_g, n_g)}{\pi_{\theta_{\text{old}}}(z_{g,i,t} \mid z_{g,i,<t}, x_g, n_g)}.$$

The clipped policy surrogate at that token is

$$\min\Big(\rho_{g,i,t}(\theta)\widetilde{A}_{g,i,t}, \text{clip}(\rho_{g,i,t}(\theta), 1-\epsilon, 1+\epsilon)\widetilde{A}_{g,i,t}\Big).$$

KL and entropy regularization are specified in Appendix C.

Training The critic follows the ranker's temporal training, validation, and test partitions. Input histories use information available before each prediction point. Critic fitting uses training-partition interactions only; validation and test target labels are excluded from fitting. We train for 30 epochs and select the checkpoint with the lowest validation MSE. Held-out test targets are used only for final evaluation and do not enter critic fitting or checkpoint selection.

Results Table 3 reports prediction error and Pearson correlations for the critic. The predicted means correlate positively with observed engagement, while variance/error correlations are 0.300 on KuaiRec and 0.132 on MovieLens-1M. These are descriptive associations; they do not establish variance calibration or explain the effect of response weighting on policy learning. The downstream ranking comparison is reported separately in Table 8.

I Theoretical Analysis of the Swap-based Item-level Reward

We provide a theoretical analysis of the proposed swap-based item-level reward. The goal is to characterize the local properties that make the reward suitable for autoregressive ranking, rather than to claim a global equivalence between the summed item-level rewards and the original listwise objective. Recall that a generated ranking is denoted by $y = (a_1, \ldots, a_K)$ over candidate set C, and its quality under need n is measured by $R_n(y; x)$. In the main text, we write the objective as a decomposable position-dependent reward

$$R_n(y; x) = \sum_{k=1}^{K} r_n(a_k, k; x).$$

For the analysis below, we make the position dependence explicit. For DCG/NDCG-style objectives, we can write

$$R_n(y; x) = \sum_{k=1}^{K} d_k \, \ell_n(a_k; x),$$

where $d_1 \geq d_2 \geq \cdots \geq d_K$ is a non-increasing position discount and $\ell_n(a_k; x)$ is the need-specific gain of item a_k. For NDCG, the query-level normalization constant is fixed for a given (x, C, n) and can be absorbed into ℓ_n.

Consistency with original ranking objective. For any $k < j$, let $y^{(k \leftrightarrow j)}$ denote the ranking obtained by swapping the items at positions k and j. Since the two rankings differ only at these two positions,

$$R_n(y^{(k\leftrightarrow j)}; x) - R_n(y; x) = (d_k - d_j)\big(\ell_n(a_j; x) - \ell_n(a_k; x)\big).$$

Therefore, the negative swap effect used in our reward satisfies

$$-\Big(R_n(y^{(k\leftrightarrow j)}; x) - R_n(y; x)\Big) = (d_k - d_j)\big(\ell_n(a_k; x) - \ell_n(a_j; x)\big).$$

When $d_k > d_j$, this quantity is positive exactly when a_k has larger gain than a_j. If $d_k = d_j$, the swap effect is zero regardless of the gains, including ties in the zero-discount tail of a truncated metric.

Comparison within a fixed prefix. A useful way to interpret the swap-based reward is as a state-conditioned local advantage in the autoregressive ranking process. At step k, the policy state is determined by the context, need, fixed prefix, and remaining candidate pool:

$$s_k = (x, n, a_{<k}, \mathcal{C}_k), \qquad \mathcal{C}_k = C \setminus \{a_1, \dots, a_{k-1}\}.$$

The action is to select an item $a \in \mathcal{C}_k$ for rank k. Our reward compares the selected item with feasible alternatives under the same fixed prefix; its magnitude also depends on the sampled suffix ordering and position discounts.

For $k < K$, our item-level reward is defined as the negative average swap effect over feasible future positions:

$$r_k^{\mathrm{CS}} = -\frac{1}{K-k} \sum_{j=k+1}^{K} \left[R_n(y^{(k \leftrightarrow j)}; x) - R_n(y; x) \right].$$

Equivalently,

$$r_k^{\mathrm{CS}} = \frac{1}{K-k} \sum_{j=k+1}^{K} (d_k - d_j)\big(\ell_n(a_k; x) - \ell_n(a_j; x)\big).$$

When $\sum_{j>k}(d_k - d_j) > 0$, let

$$\alpha_k = \frac{1}{K-k} \sum_{j=k+1}^{K} (d_k - d_j), \qquad p_{k,j} = \frac{d_k - d_j}{\sum_{\ell=k+1}^{K}(d_k - d_\ell)}.$$

Then

$$r_k^{\mathrm{CS}} = \alpha_k \left[\ell_n(a_k; x) - \sum_{j=k+1}^{K} p_{k,j} \ell_n(a_j; x) \right].$$

This expresses the reward as a comparison with a discount-weighted average of future alternatives in the realized rollout. If the denominator defining $p_{k,j}$ is zero, all those swap effects are zero and $r_k^{\mathrm{CS}} = 0$; the weights $p_{k,j}$ need not be defined. At $k = K$, we use the same zero-reward convention. This interpretation does not imply equal reward distributions across ranks or prefixes.

Connection to local action advantage and optimal local decisions. The same quantity can also be related to the expected listwise reward after making a choice. Let

$$m = |\mathcal{C}_k|,$$

and suppose the model selects item $a \in \mathcal{C}_k$ for rank k. Let ρ denote an exchangeable suffix-completion policy, under which the remaining positions $k+1, \dots, K$ are filled by an exchangeable ordering of $\mathcal{C}_k \setminus \{a\}$. Define the local action value

$$Q_\rho(s_k, a) = \mathbb{E}_\rho[R_n(y; x) \mid s_k, a_k = a].$$

Define a uniform-action baseline over feasible items (not the learned policy's action-weighted state value):

$$V_\rho(s_k) = \frac{1}{|\mathcal{C}_k|} \sum_{b \in \mathcal{C}_k} Q_\rho(s_k, b).$$

Then the local action advantage is

$$A_\rho(s_k, a) = Q_\rho(s_k, a) - V_\rho(s_k).$$

For the decomposable objective above, the only part of $Q_\rho(s_k, a)$ that depends on the selected item a is its gain at position k relative to the expected suffix contribution. Under exchangeable suffix completion, this gives

$$A_\rho(s_k, a) = \alpha_k \left[\ell_n(a; x) - \frac{1}{m} \sum_{b \in \mathcal{C}_k} \ell_n(b; x) \right].$$

Meanwhile, the expected swap-based reward satisfies

$$\mathbb{E}[r_k^{\mathrm{CS}} \mid s_k, a] = \alpha_k \left[\ell_n(a; x) - \frac{1}{m-1} \sum_{b \in \mathcal{C}_k \setminus \{a\}} \ell_n(b; x) \right].$$

Equivalently,

$$\mathbb{E}[r_k^{\mathrm{CS}} \mid s_k, a] = \alpha_k \frac{m}{m-1} \left[\ell_n(a; x) - \frac{1}{m} \sum_{b \in \mathcal{C}_k} \ell_n(b; x) \right].$$

Therefore,

$$\mathbb{E}[r_k^{\mathrm{CS}} \mid s_k, a] = \frac{m}{m-1} A_\rho(s_k, a).$$

This identity holds for $m > 1$ under exchangeable suffix completion and the uniform-action baseline above. It does not identify the advantage under an arbitrary learned autoregressive continuation.

For $\alpha_k > 0$, within a fixed state and under the same completion assumption, the maximizing actions satisfy:

$$\arg\max_{a \in \mathcal{C}_k} \mathbb{E}[r_k^{\mathrm{CS}} \mid s_k, a] = \arg\max_{a \in \mathcal{C}_k} \ell_n(a; x).$$

If $\alpha_k = 0$, all actions receive zero swap reward and tie in local value under this completion model. Exact swap averaging for a fixed rollout does not establish unbiased policy gradients, global objective equivalence, or comparability of pooled rewards under a learned suffix policy.

Rank-wise normalization control. For the additional GRPO row in Table 7, we retain future-only swap rewards and instead compute the mean and standard deviation across rollouts separately at each rank: $A_{g,i,k} = (r_{g,i,k}^{\mathrm{CS}} - \mu_{g,k})/(\sigma_{g,k} + \varepsilon)$. Other training settings and the sequence-level advantages on non-item tokens remain unchanged. NDCG@5 decreases from 0.607 with pooled normalization to 0.460 with rank-wise normalization. Since the terminal raw rewards are all zero, rank-wise centering also gives zero terminal advantage; this control does not independently separate normalization from terminal credit.

Computing the reward with suffix sums. For fixed gains $\ell_k = \ell_n(a_k; x)$ and discounts, define $S_\ell(k) = \sum_{j>k} \ell_j$, $S_d(k) = \sum_{j>k} d_j$, and $S_{d\ell}(k) = \sum_{j>k} d_j \ell_j$. Expanding the swap expression gives

$$r_k^{\mathrm{CS}} = d_k \ell_k - \frac{d_k S_\ell(k) + \ell_k S_d(k) - S_{d\ell}(k)}{K - k}, \qquad k < K.$$

Three reverse cumulative sums compute all rewards in $\mathcal{O}(K)$ arithmetic after the gains are available. This algebraic alternative assumes a fixed normalization for the candidate set; it is not an end-to-end runtime measurement or a claim about the implementation used for the reported runs.

J Joint Multi-Need Training

Figure 3 reports joint need-conditioned FlexRec results on both recommendation datasets. We complement these results with a controlled instruction-switch evaluation and qualitative examples.

J.1 Instruction-Switch Evaluation

We evaluate the same jointly trained checkpoint on the same MovieLens-1M test set, holding user histories, candidate sets and order, item metadata, and decoding settings fixed while changing only the need instruction. Each generated ranking is scored under all three objectives. As shown in Table 5, the matching instruction has the highest mean NDCG@5 for every objective. The margins over the best non-matching instruction are 0.0028, 0.0033, and 0.0540 for interest, new-topic discovery, and trending, respectively. The clearest separation is for trending; the reported intervals for the leading interest and new-topic means overlap. This supports instruction sensitivity, especially for trending, while paired-difference intervals are needed to assess the smaller margins.

Table 5: Instruction switching on the same MovieLens-1M test set using one joint checkpoint. Rows specify the input instruction; columns specify the scoring objective. Each cell shows mean NDCG@5 (top) and its 95% query-bootstrap confidence interval (bottom); the highest mean in each column is bold. Per-cell intervals do not directly quantify uncertainty in paired instruction differences.

Input instruction	Interest	New-topic	Trending
Interest	**0.5352** [0.5235, 0.5469]	0.7935 [0.7870, 0.7999]	0.5424 [0.5332, 0.5513]
New-topic	0.5259 [0.5142, 0.5374]	**0.7968** [0.7904, 0.8033]	0.5406 [0.5313, 0.5498]
Trending	0.5324 [0.5206, 0.5440]	0.7501 [0.7435, 0.7572]	**0.5964** [0.5876, 0.6050]

J.2 Qualitative Analysis on Multiple Needs

The examples show how the model describes its ranking choices under each need instruction. They provide qualitative context for the quantitative results; rationale faithfulness is a separate evaluation question (Ge et al., 2024).

Need: Maximizing Interest

Prompt:
Given user history (in chronological order): ... Rank them to maximize the user's expected watch ratio (likelihood of long/complete viewing) based on their history—because higher watch ratio signals stronger engagement and improves retention. Also keep in mind to recommend items that strongly align with the user's known interests.

Model Response:
<think>
Based on the user's interest in celebrity gossip coupled with a liking for short engaging moments related to popular celebrities, a strong recommendation would include several videos under 'celebrity amusement' category path. The variety might increase the chances for longer engagement. Another notable behavior could involve following content lengths of approximately equal duration to what we see the user engaging normally with, at around 11.22 seconds play duration.
</think>
<answer>: ...

Need: Niche Discovery

Prompt:
Given user history (in chronological order): ... Keep in mind to recommend not only items aligned with the user's interests but also items that are niche (topics the user hasn't interacted with), but that are plausibly relevant and could enrich their experience.

Model Response:
<think>
First, let's look at how videos are categorized and tagged. Users seem interested in aesthetics (specific tags like "beauty"), emotions, entertainment (tags like "amusement" and "short play"), practical life hacks ("life"), and family events (like weddings and birth). However, there were no videos regarding kids or parenting recently noted, which could be niche but relevant.
</think>
<answer>: ...

Need: Trend Promotion

Prompt:
Given user history (in chronological order): ... Keep in mind to recommend not only items aligned with the user's interests but also items that are popular in the last 24 hours (provided as the number of watches in the last day), as long as they are relevant to the user.

Model Response:
<think>
I will start with the most watched video in the last 24 hours ('1429'), which is '6549 views'. This should grab attention as it's very popular right now. Next I'll pick videos with high watches, like '4803', '1642' and others. Then I'll include some trending content with a mix of topics and styles. After that, I'll add newer, higher-watched videos along with those in the browsing history, maintaining order from highest to lowest. Finally, I'll round out the list with lower preference but still relevant videos.
</think>
<answer>: ...

Need: Product Search

Prompt:
Given user query: qscreen fence without holes ... Your task is to rerank these candidate products by how well they match the user's search query. Consider how closely each product's title, description, and attributes align with the query intent. Give higher ranks to products that are exact or highly relevant matches (e.g., correct type, brand, or specifications). Lower the rank of products that are complementary or unrelated to the query.

Model Response:
<think>
I start by examining the user query '!qscreen fence without holes' which seems to indicate a need for a fence that screens off an area but does not require installing holes or special hardware (like grommets or tie downs). Based on this, I'll consider which ASINs would likely match this request- B07DS1YCRZ from ColourTree appears to be a suitable product, offering commercial grade, UV-stabilized fence screening fabric with cable zip ties included. The 'screen' description aligns perfectly with the search criteria.- B08NG85RHL from Orgrimmar is also a heavy-duty privacy screen intended for residential and commercial uses, matching the commercial-grade requirement and screen usage, though its specific size is too large compared to what might be needed for normal fences. </think>
<answer>: ...

K GENERALIZATION RESULTS ON TREND PROMOTION

Table 6: Need generalization results (train: **Maximizing Interest**, test: **Trend Promotion**) show that transfer varies across datasets and metrics.

Generalization Setting	KuaiRec Train: Max-Interest → Test: Trend Promotion					MovieLens-1M Train: Max-Interest → Test: Trend Promotion				
Models	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**	**N@5**	**N@10**	**N@30**	**R@5**	**MRR@5**
Traditional Rerankers										
BERT4Rec (Sun et al., 2019)	0.479	0.520	0.774	0.123	0.085	0.542	0.601	0.820	0.181	0.212
STAR (Lee et al., 2025)	0.579	0.624	0.821	0.164	0.095	0.604	0.655	0.844	0.228	0.280
Zero-Shot LLMs										
GPT-4o	**0.622**	**0.647**	**0.844**	**0.306**	0.112	**0.669**	**0.710**	**0.868**	**0.320**	0.349
Qwen2.5-3B-Instruct	0.540	0.582	0.808	0.230	0.176	0.581	0.627	0.834	0.220	0.280
Post-Trained LLMs (trained on Max-Interest)										
Qwen2.5-3B-Instruct										
+ TALLRec (Bao et al., 2023)	0.493	0.552	0.790	0.152	0.118	0.611	0.648	0.845	0.244	0.322
+ Rec-R1 (Lin et al., 2025)	0.498	0.552	0.791	0.158	0.120	0.607	0.653	0.844	0.252	0.306
+ ConvRec-R1 (Zhu et al., 2025b)	0.472	0.531	0.780	0.115	0.080	0.510	0.573	0.808	0.139	0.184
+ ReasonRank (Liu et al., 2025)	0.549	0.595	0.811	0.217	**0.474**	0.571	0.625	0.830	0.220	**0.587**
+ BLADE-N (Chen et al., 2026)	0.567	0.613	0.818	0.222	0.458	0.558	0.614	0.825	0.204	0.564
+ FlexRec	0.563	0.611	0.816	0.167	0.124	0.630	0.677	0.854	0.269	0.329
% Gain vs Qwen2.5-3B-Instruct	+4.3%	+5.0%	+1.0%	-27.4%	-29.5%	+8.4%	+8.0%	+2.4%	+22.3%	+17.5%

Table 7: Reward and normalization ablations on dense KuaiRec *Maximizing Interest*. The added rank-wise GRPO variant retains future-only swap credit; the proposed variant uses pooled normalization.

Alg.	Reward Formulation	N@5	N@10	N@30	R@5	MRR@5
GRPO	Independent contribution	0.380	0.455	0.695	0.167	0.533
	Swap-based contribution (prefix-agnostic)	0.461	0.514	0.731	0.214	0.716
	Future-only swaps + rank-wise normalization	0.460	0.513	0.730	0.213	0.715
	Sequential-dependence-aware item-level reward (ours)	**0.607**	**0.636**	**0.790**	**0.336**	**0.851**
PPO	Independent contribution	0.390	0.462	0.698	0.175	0.546
	Swap-based contribution (prefix-agnostic)	0.383	0.455	0.695	0.169	0.534
	Sequential-dependence-aware item-level reward (ours)	**0.621**	**0.661**	**0.803**	**0.338**	**0.850**

Table 8: Ablation of reward signals and response weighting on KuaiRec. The critic-based rows train ranking policies using critic rewards. The proposed weighting improves NDCG and Recall@5 over no weighting, constant weighting, and shuffled weighting (results that use the inverse uncertainty directly as weight leads to broken training runs, thus we do not include the results here).

Reward Signal	N@5	N@10	N@30	R@5	MRR@5
CF-based					
User-KNN CF	0.410	0.484	0.707	0.194	0.570
Item-KNN CF	0.417	0.480	0.707	0.208	0.603
Critic-based					
No response weighting	0.566	0.591	0.775	0.280	**0.831**
Constant response weight	0.565	0.590	0.774	0.279	0.830
Shuffled response weights	0.565	0.590	0.774	0.279	0.830
FlexRec: Uncertainty-aware Critic	**0.595**	**0.623**	**0.786**	**0.319**	0.825

L Results on ESCI Search

ESCI (Reddy et al., 2022) evaluates product ranking conditioned on an explicit search query. Table 9 reports results in this setting, testing FlexRec beyond sequential recommendation. FlexRec achieves the highest scores across the reported metrics, supporting its applicability to query-driven product search.

Table 9: Performance on **ESCI** (BERT4Rec not applicable to non-sequential setting), showing that FlexRec is effective not only for sequential recommendation but also for non-sequential ranking tasks such as product search.

Product Search	N@5	N@10	N@30	P@5	R@5
Traditional Rankers					
STAR (Lee et al., 2025)	0.412	0.466	0.634	0.220	0.533
Zero-Shot LLMs					
GPT-4o	0.502	0.530	0.675	0.264	0.647
Qwen2.5-3B-Instruct	0.449	0.493	0.650	0.235	0.585
Post-Trained LLMs					
Qwen2.5-3B-Instruct					
+ TALLRec (Bao et al., 2023)	0.399	0.459	0.626	0.210	0.507
+ Rec-R1 (Lin et al., 2025)	0.504	0.536	0.679	0.268	0.652
+ ConvRec-R1 (Zhu et al., 2025b)	0.397	0.454	0.623	0.213	0.513
+ FlexRec	**0.528**	**0.553**	**0.690**	**0.277**	**0.678**
Δ vs Qwen2.5-3B-Instruct	+17.6%	+12.2%	+6.2%	+17.9%	+15.9%

M Extra Results on KuaiRec

Dense Setting Results Throughout this work, we primarily assume sparse user–item interactions, as is typical in practice. KuaiRec has a dense version, where nearly every user has interacted with nearly every item. In this setting, rewards for user–item pairs are largely observed, reducing the need for a learned critic. Table 10 reports results on KuaiRec-dense. FlexRec exceeds the reported baselines on this dense benchmark, including NDCG@5 of 0.607 versus 0.572 for Rec-R1 (a 6.1%

Table 10: Performance on the **Maximizing Interest** need on dense version of KuaiRec.

Maximizing Interest (KuaiRec)	N@5	N@10	N@30	R@5	MRR@5
Baselines					
BERT4Rec (Sun et al., 2019)	0.415	0.479	0.706	0.182	0.578
STAR (retriever) (Lee et al., 2025)	0.384	0.457	0.697	0.170	0.541
GPT-4o	0.376	0.451	0.695	0.154	0.505
Qwen2.5-3B	0.375	0.451	0.693	0.160	0.515
Post-Trained LLMs (Dense)					
Qwen2.5-3B + SFT	0.585	0.612	0.756	0.333	0.783
Qwen2.5-3B + PPO	0.386	0.458	0.697	0.170	0.541
Qwen2.5-3B + Rec-R1 (Lin et al., 2025)	0.572	0.616	0.769	0.307	0.780
Qwen2.5-3B + ConvRec-R1 (Zhu et al., 2025b)	0.381	0.456	0.696	0.168	0.517
Qwen2.5-3B + GRPO (ours)	**0.607**	**0.636**	**0.790**	**0.336**	**0.851**

relative gain). This supports a benefit from the reward design when feedback completion is less important.

Experiments on Larger Models To examine whether FlexRec scales to larger LLM backbones, we further evaluate Qwen2.5-7B-Instruct on KuaiRec. Since the main text already provides detailed comparisons with post-trained baselines, we focus here on representative non-post-trained baselines and the corresponding 7B backbone. As shown in Table 11, FlexRec substantially improves over the base 7B model across all three needs, including both the training need and zero-shot generalization settings. These results suggest that the benefits of FlexRec are not limited to a specific backbone size, but persist when scaling to larger LLM recommenders.

Table 11: Additional KuaiRec results with a larger 7B backbone.

Model	Maximizing Interest		Generalize: Explore New Topics		Generalize: Trend Promotion	
	N@5	N@10	N@5	N@10	N@5	N@10
BERT4Rec (Sun et al., 2019)	0.415	0.479	0.720	0.784	0.479	0.520
GPT-4o	0.376	0.451	0.637	0.703	**0.622**	**0.647**
Qwen2.5-7B-Instruct	0.354	0.434	0.706	0.757	0.491	0.550
Qwen2.5-7B-Instruct + FlexRec	**0.602**	**0.660**	**0.785**	**0.808**	0.544	0.626
Δ vs Qwen2.5-7B-Instruct	+70.1%	+52.1%	+11.2%	+6.7%	+10.8%	+13.8%

N Additional Baseline Details

Baseline adaptations. The additional methods start from Qwen2.5-3B-Instruct and are trained on maximizing interest for 833 optimizer updates with 30-candidate ranking instances. ReasonRank and BLADE-N adapt the respective RL objectives to our candidate pools without their external SFT or teacher pipelines. These rows compare the adapted objectives under this setup, not complete reproductions of the published systems. Both use seed 42, a prompt batch size of 6, four rollouts per prompt, one training epoch, and a maximum response length of 1,024 tokens, as recorded in their run specifications. Their rollout group size differs from FlexRec's eight; consequently these results are not an isolation of the objective at identical sampled-token budget.

Evaluation. For these evaluations, the scorer applies the gain $2^z - 1$ once to the supplied target z: watch ratio or raw rating for interest, the constructed novelty score for new topics, and the constructed trend label for trending. IDCG uses the ideal ordering of the supplied pool. Binary positives are rating-5 candidates ($z > 4$) for MovieLens-1M interest, $z > 1$ for KuaiRec interest and both new-topic tasks, and $z > 0.5$ for trending. Recall@5 divides top-five positives by all positives in the candidate pool; MRR@5 uses the reciprocal rank of the first positive within five positions. Queries with no positives contribute zero to the binary metrics. All queries enter the macro-average, including the 244 MovieLens-1M interest queries with no rating-5 candidates. Evaluation uses temperature 0.9, top-$p = 0.95$, top-$k = 50$, repetition penalty 1.05, and a 1,024-token response cap for both methods. KuaiRec trending prompts include observed one-day activity counts. All scores are single-run point estimates. Each interest/new-topic test has 1,000 queries; trending has 315 on KuaiRec and 1,000 on MovieLens-1M.

Table 12: Additional two-stage ReasonRank (Liu et al., 2025) diagnostic on KuaiRec: the interest checkpoint at update 833, followed by 705 new-topic updates without interest replay. Prompts, decoding settings, test labels, and scorer are held fixed. Interest/new-topic tests contain 1,000 queries each; trending contains 315. All queries are retained, including 11 interest-test and 3 new-topic-test context overlaps with cumulative training data. This completed two-stage run is separate from joint training and does not include a subsequent trending stage.

Test need	Training	N@5	N@10	N@30	R@5	MRR@5
Interest	Interest	0.385	0.458	0.697	0.172	0.546
	Interest → new topics	0.384	0.458	0.697	0.168	0.539
New topics	Interest	0.706	0.756	0.911	0.127	0.366
	Interest → new topics	0.701	0.754	0.910	0.115	0.335
Trending	Interest	0.549	0.595	0.811	0.217	0.474
	Interest → new topics	0.550	0.600	0.812	0.217	0.466