F-GRPO: FACTORIZED GROUP-RELATIVE POLICY OPTIMIZATION FOR UNIFIED CANDIDATE GENERATION AND RANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Traditional retrieval pipelines optimize utility through stages of candidate retrieval and reranking, where ranking operates over a predefined candidate set. Large Language Models (LLMs) broaden this into a generative process: given a candidate pool, an LLM can generate a subset and order it within a single autoregressive pass. However, this flexibility introduces a new optimization challenge: the model must search a combinatorial output space while receiving utility feedback only after the full ranked list is generated. Because this feedback is defined over the completed sequence, it cannot distinguish whether a poor result arises from failing to generate a relevant subset or from failing to rank that subset correctly. This credit assignment gap makes end-to-end optimization unstable and sample-inefficient. Existing systems often address this challenge by separating candidate generation from ranking. However, such decoupling remains misaligned with downstream utility because the ranking stage is fundamentally limited by the candidate set it receives. To bridge the optimization gap between candidate generation and ranking, we propose a unified framework that performs both within a single autoregressive rollout and optimizes them end-to-end via factorized group-relative policy optimization (F-GRPO). Our framework factorizes the policy into candidate generation and ranking while sharing a single LLM backbone across both stages, and jointly trains them with an order-invariant coverage reward and a position-aware utility reward. To address the resulting phase-specific credit assignment problem, we use separate group-relative advantages for generation and ranking within a two-phase sequence-level objective. Across sequential recommendation and multi-hop question answering benchmarks, F-GRPO improves ranking quality over GRPO and decoupled baselines, outperforms supervised alternatives, and exceeds zero-shot rerankers on most metrics, with no architectural changes at inference time.

1 INTRODUCTION

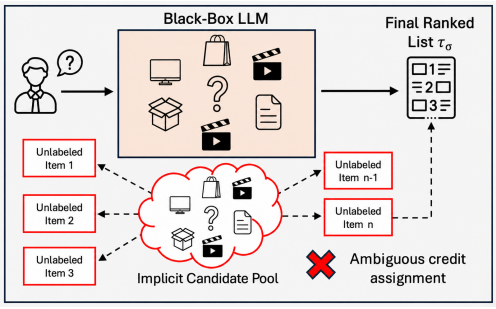
Retrieval and recommendation systems face a coupled *list-to-rank* problem, which requires identifying a slate of relevant candidates and ordering them so the best appear first (Robertson & Zaragoza, 2009; Liu, 2009; Ni et al., 2026; Li et al., 2025a; Huang et al., 2025a; Xie et al., 2025; Yan et al., 2024). Traditionally, this is handled by multi-stage pipelines that retrieve candidates and then rerank them (Nogueira et al., 2020; Glass et al., 2022; Yue et al., 2023; Wu et al., 2025b; Hu et al., 2025b; Xie et al., 2024; Wu et al., 2024a). Large Language Models (LLMs) have broadened this space as stronger rerankers and listwise rankers over retrieved candidate sets (Sun et al., 2023; Hou et al., 2024; Surana et al., 2025). Neither stage can compensate for the other: ranking cannot recover candidates that were never selected, and coverage is wasted unless the best candidates are ranked first.

However, existing LLM-based approaches either conflate or separate these two stages. Some treat the LLM as a direct reranker over a retrieved pool (Sun et al., 2023; Hou et al., 2024; Xia et al., 2025a;b), outputting only a final ranking, so training feedback cannot distinguish coverage errors from ordering errors (Figure 1a). Others decompose the problem into separate modules (Yue et al., 2023; Trivedi et al., 2023; Huang et al., 2025b), such as a retriever followed by a reranker. While effective, this separation introduces additional models and optimizes each stage for its own proxy objective (Gupta et al., 2025) rather than for the coupled list-to-rank decision.

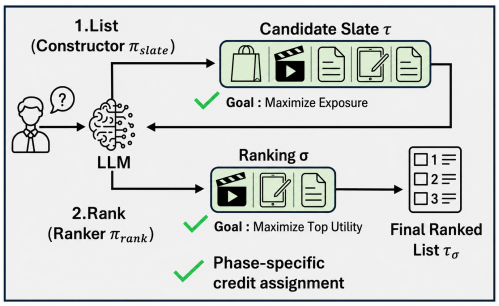
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.

Instead, we make the list-to-rank decision explicit within a single LLM rollout. The model first selects a slate from the candidate pool and then ranks it within the same autoregressive trajectory (Figure 1b). We term this *in-context exploration* because the slate phase explicitly searches the candidate space before committing to a ranking. Keeping both phases in the same context targets the coupled decision directly: the ranking phase conditions on the generated candidates, enabling end-to-end optimization without the separate modules required by staged pipelines (Yue et al., 2023; Trivedi et al., 2023; Wu et al., 2024b).

Training such a rollout with vanilla GRPO raises a credit assignment problem. A single sequence-level reward conflates slate construction and ranking, so the same feedback rewards coverage and ordering (Wu et al., 2025a; Huang et al., 2026b). A rollout that ranks well but misses relevant candidates can still reinforce its slate tokens, and a rollout with good coverage can be penalized for a poor ordering. We therefore propose *F-GRPO*, which extends GRPO (Shao et al., 2024; Guo et al., 2025; Mundada et al., 2026; Surana et al., 2026) with *two-phase sequence-level credit assignment*, in which slate tokens receive a coverage advantage and rank tokens receive a separate ranking advantage. Both rewards come from the same candidate relevance labels, so F-GRPO needs no additional annotation or architectural changes and applies to any list-to-rank task with such labels.



(a) Black-box LLM with implicit candidate pool.



(b) Factorized in-context generation and ranking with phase-specific goals.

Figure 1: **(a)** Black-box ranking conflates candidate selection and ordering, yielding ambiguous credit assignment. **(b)** Factorized in-context generation and ranking with phase-specific goals within a single autoregressive rollout.

We evaluate F-GRPO on two distinct tasks: (a) sequential recommendation (MovieLens (Harper & Konstan, 2015), LastFM (Bertin-Mahieux et al., 2011)) and (b) multi-hop question answering (HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022)) using Qwen3-4B (Team, 2025) and Qwen3.5-2B (Qwen Team, 2026) (§4.2, §4.3). F-GRPO outperforms GRPO in most settings, with the clearest gains at higher cutoffs and where coverage is limiting (e.g., +31.4% relative Recall@3 on LastFM and +13.2% on MuSiQue with Qwen3-4B). On multi-hop QA, F-GRPO also outperforms dedicated zero-shot rerankers on most metrics. A matched ablation changing only the credit-assignment rule shows that separate advantages outperform a shared advantage on every metric across LastFM and MuSiQue, and training-dynamics and error-attribution analyses show that slate coverage and within-slate ranking both matter (§5).

Contributions. Our main contributions are as follows:

- We formalize the list-to-rank decision as a factorized policy over a proposal slate and its ranking, with an objective that jointly optimizes coverage and ranking quality within a single autoregressive rollout (§2).
- We propose F-GRPO, a two-phase sequence-level GRPO method assigning a separate group-relative advantage to each phase, show that a shared advantage lets one phase's reward reverse the other's credit, and instantiate it for recommendation and QA (§3, §4.1).
- On four recommendation and QA benchmarks, F-GRPO improves Recall@k and NDCG@k over zero-shot, supervised, decoupled, and GRPO baselines and exceeds zero-shot rerankers on NDCG; a matched ablation isolates the benefit of separate credit assignment (§4.2, §4.3, §5).

2 Problem Formulation

We study a coupled list-to-rank decision in which, for each context $x \in \mathcal{X}$ (a query or user history together with a candidate pool), a model must first construct a slate of candidates and then produce a ranking from that slate. Let $\mathcal{E}(x)$ denote the candidate pool supplied with the context, for example

by an upstream retriever or recommender, so the task is post-retrieval subset selection and ranking: candidate generation means emitting a smaller slate from this pool. Our goal is to improve both (i) the *coverage and quality* of the selected candidates and (ii) their *ordering* in the final list.

Factorized list-to-rank policy. Let $\tau = (e_1, \ldots, e_n)$ denote an ordered *proposal slate* of n candidates with $e_i \in \mathcal{E}(x)$. Let $\sigma = (\sigma_1, \ldots, \sigma_m)$ denote distinct slate indices in ranked order, where $1 \le m \le n$, inducing the ranked list $\tau_\sigma = (e_{\sigma_1}, \ldots, e_{\sigma_m})$. The rank phase may filter and reorder the slate. We model the joint decision (τ, σ) with a factorized policy:

$$\pi_\theta(\tau, \sigma \mid x) = \pi_\theta^{\text{slate}}(\tau \mid x) \cdot \pi_\theta^{\text{rank}}(\sigma \mid x, \tau), \tag{1}$$

where $\pi_\theta^{\text{slate}}$ is the slate generator and π_θ^{rank} is the ranker conditioned on the generated slate. Both $\pi_\theta^{\text{slate}}$ and π_θ^{rank} are realized by a single autoregressive model with shared parameters, which first generates tokens defining the slate τ and then the ranking in the same decoding trajectory.

Feedback signals. Let $U(x, e) \ge 0$ be a task-dependent utility function, and let $\mathcal{G}(x) = \{e \in \mathcal{E}(x) : U(x, e) > 0\}$ denote the set of *gold* (relevant) items for context x. The *slate generator* receives an order-invariant set utility, while the *ranker* receives a position-aware utility over τ_σ: $R_{\text{slate}}(x, \tau) \triangleq U_{\text{slate}}(x, \text{uniq}(\tau))$, $R_{\text{rank}}(x, \tau, \sigma) \triangleq U_{\text{rank}}(x, \tau_\sigma)$. Here, $\text{uniq}(\tau)$ denotes the set of distinct items in τ, and U_{slate} aggregates candidate relevance through a set-level measure such as recall or F1. These signals differ structurally: R_{slate} evaluates the selected set independently of its order, whereas R_{rank} is position-sensitive and depends on the final ranked list, so the two rewards can favor different rollouts (Section 3.3). The specific reward instantiations are given in Section 4.1. Both rewards are computed from the same candidate-level relevance labels, providing phase-specific feedback without separate phase annotations.

Learning objective. Let $\mathcal{D}$ denote the distribution over contexts. The coupled task-level goal is to maximize:

$$\max_\theta \; \mathbb{E}_{x \sim \mathcal{D}} \, \mathbb{E}_{\tau \sim \pi_\theta^{\text{slate}}(\cdot|x)} \, \mathbb{E}_{\sigma \sim \pi_\theta^{\text{rank}}(\cdot|x,\tau)} \Big[R_{\text{slate}}(x, \tau) + \lambda \, R_{\text{rank}}(x, \tau, \sigma) \Big], \tag{2}$$

where $\lambda \ge 0$ controls the trade-off between candidate coverage and ranking quality. Section 3 develops a factorized surrogate for this goal, assigning each reward to its corresponding output span.

3 F-GRPO: Factorized Group-Relative Policy Optimization

F-GRPO keeps the single-rollout structure of GRPO but trains the slate and rank spans with separate losses, each weighted by its own group-relative advantage.

3.1 Two-Span Rollouts and GRPO

For each context x, GRPO samples a group of G rollouts from the current sampling policy $\pi_{\theta_{\text{old}}}$. We parse each rollout $o^{(i)}$ into two segments: the slate content $c_\tau^{(i)}$ and the rank content $c_\sigma^{(i)}$, delimited by tag pairs (<SLATE>…</SLATE> and <RANK>…</RANK>). For a valid rollout, $c_\tau^{(i)}$ decodes to the slate $\tau^{(i)}$, and $c_\sigma^{(i)}$ decodes to the ranked item names or passage IDs in $\tau^{(i)}_{\sigma^{(i)}}$, implicitly specifying the slate indices $\sigma^{(i)}$. Delimiter tokens are included in the forward pass so that content tokens are conditioned on the correct tagged prefix, but are excluded from the loss via position-based masking (details in Appendix C.2). For readability, the conditioning expressions below suppress these fixed delimiter tokens. Standard GRPO optimizes the clipped objective (Shao et al., 2024):

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Big[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o^{(i)}|} \sum_{t=1}^{|o^{(i)}|} \min\Big(\rho_t^{(i)} \hat{A}^{(i)}, \; \text{clip}\big(\rho_t^{(i)}, 1 \pm \epsilon_{\text{clip}}\big) \hat{A}^{(i)} \Big) - \beta_{\text{KL}} \, D_{\text{KL}}\big(\pi_\theta \| \pi_{\text{ref}}\big) \Big], \tag{3}$$

where $\rho_t^{(i)} = \pi_\theta(o_t^{(i)} \mid x, o_{<t}^{(i)}) / \pi_{\theta_{\text{old}}}(o_t^{(i)} \mid x, o_{<t}^{(i)})$ is the per-token importance ratio, $\hat{A}^{(i)}$ is the rollout-level group-relative advantage, ϵ_{clip} is the clipping threshold, β_{KL} is the KL regularization coefficient, and π_{ref} is a frozen reference policy.

3.2 FACTORIZED CREDIT ASSIGNMENT

Each rollout i produces a slate $\tau^{(i)}$ and a ranking $\sigma^{(i)}$, yielding two scalar rewards: $R^{(i)}_{\text{slate}} = R_{\text{slate}}(x, \tau^{(i)})$ and $R^{(i)}_{\text{rank}} = R_{\text{rank}}(x, \tau^{(i)}, \sigma^{(i)})$ as defined in Section 2. Rather than combining these into one scalar, we compute separate advantages following Dr. GRPO (Liu et al., 2025b):

$$\hat{A}^{(i)}_{\text{slate}} = R^{(i)}_{\text{slate}} - \bar{R}_{\text{slate}}, \qquad \hat{A}^{(i)}_{\text{rank}} = R^{(i)}_{\text{rank}} - \bar{R}_{\text{rank}}, \tag{4}$$

where $\bar{R}_{\text{slate}} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}_{\text{slate}}$ and $\bar{R}_{\text{rank}} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}_{\text{rank}}$ are the per-prompt group means over the G rollouts. This phase-specific advantage assignment is the key departure from standard GRPO: standard GRPO applies one rollout-level advantage to all tokens in a completion, whereas F-GRPO applies different rollout-level advantages to the slate and rank token subsequences within the same autoregressive rollout. As a result, the credit assigned to slate tokens depends only on coverage, and the credit assigned to rank tokens depends only on ranking quality. The complete training procedure is summarized in Algorithm 1 in the appendix.

Slate loss. Let C denote the fixed completion-length normalizer used by Dr. GRPO (the maximum completion length in our implementation). The slate loss optimizes slate content conditioned on the prompt x:

$$\mathcal{L}_{\text{slate}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{C}\sum_{t=1}^{|c^{(i)}_\tau|} \min\!\big(\rho^{(i)}_{\tau,t}\,\hat{A}^{(i)}_{\text{slate}},\ \text{clip}(\rho^{(i)}_{\tau,t},\, 1\pm\epsilon_{\text{clip}})\,\hat{A}^{(i)}_{\text{slate}}\big), \tag{5}$$

where $\rho^{(i)}_{\tau,t} = \pi_\theta(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t}) \,/\, \pi_{\theta_{\text{old}}}(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t})$ is the per-token importance ratio and the advantage $\hat{A}^{(i)}_{\text{slate}}$ is uniform across all slate tokens.

Rank loss. The rank loss optimizes ranking content given the prompt *and the generated slate*:

$$\mathcal{L}_{\text{rank}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{C}\sum_{t=1}^{|c^{(i)}_\sigma|} \min\!\big(\rho^{(i)}_{\sigma,t}\,\hat{A}^{(i)}_{\text{rank}},\ \text{clip}(\rho^{(i)}_{\sigma,t},\, 1\pm\epsilon_{\text{clip}})\,\hat{A}^{(i)}_{\text{rank}}\big), \tag{6}$$

where $\rho^{(i)}_{\sigma,t}$ is defined analogously with context $(x, \tau^{(i)})$. Conditioning on the generated slate preserves the autoregressive dependency between phases.

Combined objective. The total loss combines both phases with optional KL regularization:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{slate}} + \lambda\,\mathcal{L}_{\text{rank}} + \beta_{\text{KL}}\,D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}). \tag{7}$$

As shown in Appendix A, this loss admits a first-order decomposition into slate and rank GRPO-style gradients on shared parameters, implemented with a single backward pass through the combined objective. During training, rollouts that fail to produce the required delimiter tags, or that rank items outside the slate, receive a constant format penalty $p_{\text{fmt}} < 0$ in place of the affected phase's reward; Appendix C.3 specifies how this penalty is applied to the generated tokens in malformed cases.

3.3 GRADIENT ANALYSIS

For comparison, consider *Structured Joint GRPO*, which retains the same two-block rollout but defines a single combined reward $R^{(i)}_{\text{joint}} = R^{(i)}_{\text{slate}} + \lambda\,R^{(i)}_{\text{rank}}$ and computes a joint advantage

$$\hat{A}^{(i)}_{\text{joint}} = R^{(i)}_{\text{joint}} - \bar{R}_{\text{joint}}, \tag{8}$$

where $\bar{R}_{\text{joint}} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}_{\text{joint}}$ is the corresponding per-prompt group mean. This joint advantage is applied uniformly to both output spans. At $\theta = \theta_{\text{old}}$, define the fixed-length-normalized score gradients for the two spans as

$$\begin{aligned} g^{(i)}_\tau &\triangleq \frac{1}{C}\sum_{t=1}^{|c^{(i)}_\tau|} \nabla_\theta \log \pi_\theta(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t}), \\ g^{(i)}_\sigma &\triangleq \frac{1}{C}\sum_{t=1}^{|c^{(i)}_\sigma|} \nabla_\theta \log \pi_\theta(c^{(i)}_{\sigma,t} \mid x, \tau^{(i)}, c^{(i)}_{\sigma,<t}). \end{aligned} \tag{9}$$

Ignoring clipping and KL terms for clarity, Structured Joint GRPO yields

$$\nabla_\theta \mathcal{L}_{\text{joint}} = -\frac{1}{G}\sum_{i=1}^{G} \hat{A}^{(i)}_{\text{joint}} \big(g^{(i)}_\tau + g^{(i)}_\sigma\big). \tag{10}$$

Because a single $\hat{A}^{(i)}_{\text{joint}}$ multiplies both span gradients, the credit assigned to slate tokens is influenced by ranking quality and vice versa. This creates a phase-specific credit-assignment ambiguity: consider a rollout with above-average ranking quality ($\hat{A}^{(i)}_{\text{rank}} > 0$) but below-average coverage ($\hat{A}^{(i)}_{\text{slate}} < 0$). Since group centering is linear, $\hat{A}^{(i)}_{\text{joint}} = \hat{A}^{(i)}_{\text{slate}} + \lambda \hat{A}^{(i)}_{\text{rank}}$. If the ranking contribution dominates, the combined advantage is positive, reinforcing slate tokens despite their below-average coverage. Conversely, sufficiently poor ranking can make the combined advantage negative despite above-average coverage, penalizing successful candidate generation. A shared scalar cannot express these opposing phase-level assessments: it conflates *what* was proposed with *how* it was ranked. Such discordance arises naturally from the different reward criteria: coverage is order-invariant, whereas ranking utility depends on which items occupy the leading positions.

F-GRPO instead uses separate span weights:

$$\nabla_\theta \mathcal{L}_{\text{F-GRPO}} = -\frac{1}{G}\sum_{i=1}^{G} \left(\hat{A}^{(i)}_{\text{slate}} g^{(i)}_\tau + \lambda \hat{A}^{(i)}_{\text{rank}} g^{(i)}_\sigma \right), \tag{11}$$

which weights each span with its corresponding advantage. Both terms update the same model parameters; factorization changes the credit weights, not the model architecture. Appendix A provides the full derivation, and Table 3 tests this change in a matched comparison.

Remark 3.1 (Phase-specific gradient weighting). Although both losses update shared parameters θ, at $\theta = \theta_{\text{old}}$ each span's score gradient is weighted solely by its corresponding advantage (Proposition A.1). This removes cross-phase reward mixing from the scalar credit weights: a ranking advantage cannot reverse the sign of the slate advantage applied to slate tokens, and vice versa. Both phases still co-adapt through the shared backbone and the ranker's conditioning on the slate.

4 Experiments

4.1 Experimental Setup

Tasks and datasets. **Sequential recommendation** (MovieLens (Harper & Konstan, 2015), LastFM (Bertin-Mahieux et al., 2011)): select and rank items from 20 candidates containing one gold next item. **Multi-hop QA evidence selection** (HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022)): select and rank supporting evidence from 20 candidates containing 2–4 gold passages. Appendix B gives dataset details.

Models and training. We experiment with *Qwen3-4B-Instruct-2507* (Team, 2025) and *Qwen3.5-2B* (Qwen Team, 2026). RL training is initialized from an SFT warm-start, except for Qwen3.5-2B on QA, which starts from the pretrained model. We use Dr. GRPO (Liu et al., 2025b) with G=8 rollouts per prompt and evaluate greedily (T=0). Slates are capped at 10 items and ranked lists at 5. Full details are in Sections E, F and H.

Reward instantiation. For both tasks, we instantiate the ranking reward with NDCG@k:

$$R^{\text{NDCG}}_{\text{rank}}(x, \tau, \sigma; k) = \frac{\text{DCG@}k(\tau_\sigma)}{\text{IDCG@}k(\tau_\sigma)}, \quad \text{DCG@}k(\tau_\sigma) = \sum_{i=1}^{\min(k,m)} \frac{y_{\sigma_i}}{\log_2(1+i)}, \tag{12}$$

where m is the number of ranked items, $y_i \in \{0, 1\}$ is the binary relevance of item e_i, and $\text{IDCG@}k(\tau_\sigma)$ is the maximum DCG obtained by reordering those same items. The reward is zero when IDCG is zero; training scores the full emitted rank ($k = m$). Evaluation NDCG instead normalizes against all gold items in the candidate pool (Appendix F). For $S = \text{uniq}(\tau)$ and $h = |S \cap \mathcal{G}(x)|$, the slate reward is recall $h/|\mathcal{G}(x)|$ for **recommendation** and F1 $2h/(|S| + |\mathcal{G}(x)|)$ for **QA**. We set $\lambda = 1.0$ in the main experiments; Section 5 examines the slate reward and λ.

Baselines. We compare F-GRPO with task-specific baselines and with zero-shot, supervised, and RL-trained LLMs. Decoupled SFT represents a two-model pipeline, chaining separately trained

Table 1: Recall@k and NDCG@k on LastFM and MovieLens for Qwen3-4B and Qwen3.5-2B. Values are percentages; within each backbone block, **bold** is best and underline second best.

	LastFM						MovieLens					
	Recall@k			NDCG@k			Recall@k			NDCG@k		
Method	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
Random	5.0	15.0	25.1	5.0	10.6	14.7	4.9	15.3	25.1	4.9	10.8	14.8
Popularity	24.7	45.9	56.6	24.7	36.9	41.3	14.4	36.3	55.1	14.4	26.9	34.6
BM25	6.2	17.8	27.7	6.2	12.8	16.8	8.7	18.4	27.7	8.7	14.1	17.9
GRU4Rec	50.8	69.0	75.5	50.8	61.3	64.0	21.8	46.8	60.2	21.8	36.1	41.6
UniSRec (zero-shot)	11.0	25.4	36.9	11.0	19.2	23.9	6.3	16.5	25.6	6.3	12.0	15.8
UniSRec (fine-tuned)	23.9	42.2	54.7	23.9	34.5	39.6	13.6	29.8	43.5	13.6	22.7	28.4
Qwen3-4B												
Zero-shot (two-step)	18.8	44.7	58.4	18.8	34.0	39.6	12.1	29.0	40.0	12.1	21.6	26.1
Zero-shot (single-step)	20.6	43.8	55.8	20.6	34.3	39.2	11.9	26.7	38.8	11.9	20.3	25.2
SFT	40.1	47.1	53.8	40.1	44.0	46.8	21.5	29.3	36.5	21.5	25.8	28.8
Decoupled SFT (sel. only)	14.1	33.3	46.0	14.1	25.3	30.5	10.1	25.3	36.0	10.1	18.9	23.3
Decoupled SFT (rank. only)	28.8	42.9	51.9	28.8	37.0	40.7	16.5	28.0	37.0	16.5	23.0	26.8
Decoupled SFT (full)	33.4	40.4	46.3	33.4	37.4	39.8	19.7	27.8	35.5	19.7	24.3	27.5
GRPO	44.7	55.1	73.9	44.7	50.8	58.6	23.9	44.4	57.7	23.9	35.8	41.2
F-GRPO (ours)	**46.8**	**72.4**	**81.7**	**46.8**	**61.7**	**65.5**	**26.9**	**53.6**	**66.6**	**26.9**	**42.3**	**47.6**
Qwen3.5-2B												
Zero-shot (two-step)	0.6	1.6	2.3	0.6	1.1	1.5	1.2	1.8	2.2	1.2	1.5	1.7
Zero-shot (single-step)	3.1	6.5	8.1	3.1	5.1	5.7	3.3	4.7	5.5	3.3	4.0	4.4
SFT	37.9	45.4	52.6	37.9	42.1	45.1	**27.7**	34.7	41.0	**27.7**	31.6	34.2
Decoupled SFT (sel. only)	0.5	1.3	2.0	0.5	0.9	1.2	0.7	1.8	2.7	0.7	1.3	1.7
Decoupled SFT (rank. only)	23.1	31.6	38.3	23.1	28.0	30.8	12.3	20.7	27.1	12.3	17.1	19.7
Decoupled SFT (full)	32.4	40.1	46.1	32.4	36.9	39.3	23.6	31.6	40.6	23.6	28.2	31.9
GRPO	38.7	60.4	71.3	38.7	51.3	55.8	24.4	**50.7**	56.4	24.4	**39.6**	**42.1**
F-GRPO (ours)	**40.0**	**63.2**	**74.3**	**40.0**	**53.5**	**58.1**	21.5	45.8	**61.1**	21.5	35.5	41.8

selector and ranker modules. The GRPO baseline instead produces a <RANK> block directly and optimizes a single ranking reward. To separate output structure from credit assignment, we also evaluate Structured Joint GRPO, which keeps F-GRPO's two-block output and training setup but applies one joint advantage to both spans. Full details are in Sections D, E.1 and I.

4.2 Sequential Recommendation

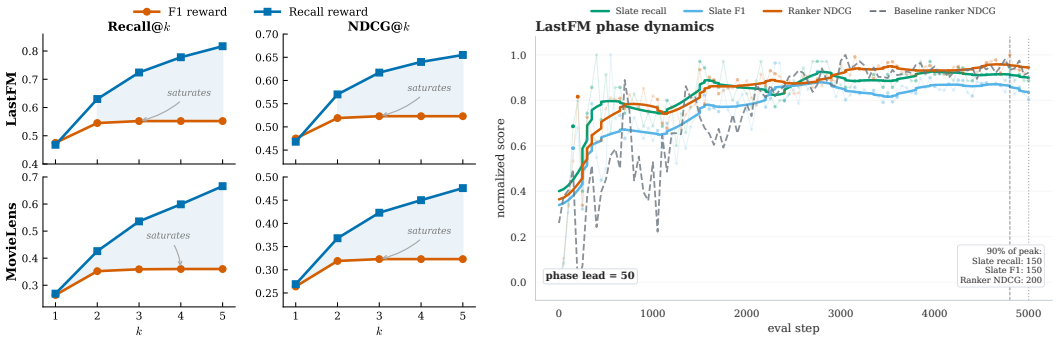
We report Recall@k and NDCG@k for $k \in \{1, 3, 5\}$ on LastFM and MovieLens in Table 1 (Precision@k and Hit@k are deferred to Appendix F).

(i) RL fine-tuning yields large gains over supervised methods. F-GRPO improves over SFT on both datasets (Qwen3-4B Recall@3: +53.7% relative on LastFM and +82.9% on MovieLens), confirming that sequence-level reward optimization provides signal beyond token-level imitation.

(ii) F-GRPO improves over GRPO. The gains are clearest at higher k, where broader slate coverage compounds with ranking quality. With Qwen3-4B, F-GRPO raises Recall@3 by 31.4% relative on LastFM and 20.7% on MovieLens, and it outperforms GRPO on every metric in three of the four dataset–backbone settings. The exception is MovieLens with Qwen3.5-2B, where F-GRPO trails GRPO at shallower cutoffs but achieves the best Recall@5 (61.1 vs. 56.4).

(iii) F-GRPO is competitive with specialized sequential models. Traditional baselines (GRU4Rec, UniSRec) score learned item embeddings over a closed item set, whereas our LLM generates item names as free-form text. Despite this harder setting, Qwen3-4B with F-GRPO surpasses all traditional methods at every cutoff on MovieLens and at @3 and @5 on LastFM.

(iv) Decoupled pipelines suffer from distribution mismatch. The full Decoupled SFT variant underperforms single-model SFT on every metric despite using two separately fine-tuned checkpoints. Its ranker is trained on gold slates but deployed on generated ones, a shift that may contribute to this gap. During F-GRPO training, the ranker instead conditions on the slate generator's own rollouts.



(a) Slate reward ablation (Qwen3-4B).

(b) Phase separation on LastFM (Qwen3.5-2B).

Figure 2: Reward ablation and training dynamics. (a) Slate reward ablation on LastFM and MovieLens. (b) On LastFM, the slate generator matures before the ranker.

Table 2: Recall@k and NDCG@k on MuSiQue and HotpotQA for Qwen3-4B and Qwen3.5-2B. Values are percentages; within each backbone block, **bold** is best and underline second best.

	MuSiQue						HotpotQA					
	Recall@k			NDCG@k			Recall@k			NDCG@k		
Method	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
Random	5.0	15.0	24.7	13.3	14.9	19.6	5.0	15.0	25.1	9.9	13.1	18.1
BM25	22.9	42.9	53.4	57.0	46.7	51.4	41.8	72.2	82.7	83.6	73.2	78.5
RankZephyr	33.5	61.3	71.5	82.3	67.1	71.1	47.4	84.9	90.3	94.8	85.9	88.6
MonoT5	34.9	64.5	74.9	86.1	70.7	74.6	46.3	86.2	92.8	92.5	86.2	89.5
DuoT5	34.3	62.8	73.5	84.8	69.2	73.3	43.2	79.8	86.2	86.4	80.0	83.3
LiT5	33.6	54.7	64.4	82.7	61.9	65.6	46.3	78.4	85.2	92.5	80.3	83.7
Qwen3-4B												
Zero-shot (two-step)	4.1	8.5	9.0	10.2	9.1	9.2	3.0	5.8	6.1	6.0	5.8	5.9
Zero-shot (single-step)	23.4	48.1	56.3	58.4	51.6	54.7	30.7	61.0	65.4	61.4	60.2	62.4
SFT	27.9	54.4	60.9	69.5	59.5	61.5	38.3	68.7	73.0	76.6	69.6	71.8
Decoupled SFT (sel. only)	8.6	16.3	18.3	21.7	18.0	18.5	9.8	19.1	20.6	19.6	18.9	19.7
Decoupled SFT (rank. only)	32.3	63.0	68.3	79.8	68.6	69.7	43.8	84.3	89.2	87.6	84.0	86.5
Decoupled SFT (full)	24.0	39.0	44.2	60.6	44.9	46.5	32.6	49.0	52.9	65.1	52.0	54.0
GRPO	36.4	63.0	63.0	90.8	71.3	69.2	**48.0**	**93.0**	**93.1**	**96.1**	93.0	93.1
F-GRPO (ours)	**36.8**	**71.3**	**71.8**	**91.6**	**78.5**	**76.5**	**48.0**	92.6	92.6	96.0	**93.3**	**93.3**
Qwen3.5-2B												
Zero-shot (two-step)	2.2	4.1	4.8	5.4	4.5	4.7	1.0	1.9	2.1	2.0	1.9	2.0
Zero-shot (single-step)	5.2	10.6	12.4	13.2	11.5	12.1	4.0	7.4	8.4	8.0	7.4	7.9
SFT	13.5	29.2	38.0	34.4	31.2	35.0	5.2	15.1	25.3	10.4	13.2	18.2
Decoupled SFT (sel. only)	0.3	0.6	0.7	0.7	0.6	0.7	0.3	0.6	0.7	0.5	0.5	0.6
Decoupled SFT (rank. only)	17.8	37.4	49.8	43.7	39.0	44.8	21.4	51.7	68.5	42.8	47.7	56.1
Decoupled SFT (full)	5.2	14.8	24.5	13.8	14.7	19.5	6.0	16.6	25.7	11.9	14.8	19.3
GRPO	36.1	63.1	65.2	**90.0**	71.4	70.6	**48.1**	86.3	86.5	96.1	88.4	88.5
F-GRPO (ours)	**36.2**	**72.7**	**73.2**	89.9	**79.2**	**77.2**	**48.1**	**91.4**	**91.4**	**96.3**	**92.5**	**92.5**

4.3 Multi-Hop Question Answering

We report Recall@k and NDCG@k for $k \in \{1, 3, 5\}$ on MuSiQue and HotpotQA in Table 2 (Precision@k and Hit@k are deferred to Appendix F).

(i) F-GRPO helps most when coverage is the bottleneck. On MuSiQue, where each question has 2–4 gold passages, F-GRPO outperforms GRPO at both model scales, with gains concentrated at higher cutoffs (Recall@3: +13.2% relative for Qwen3-4B and +15.2% for Qwen3.5-2B). On HotpotQA, the 4B models are essentially tied near saturation, while the 2B model shows a clear gap at @3 and above (+5.9% relative at Recall@3), suggesting that the factorized formulation is most beneficial when the backbone finds coverage harder. The advantage also holds on a larger pool: when MuSiQue

is augmented to 50 candidates, F-GRPO improves the six-metric average by 7.9% relative to GRPO with Qwen3.5-2B (Appendix F.2).

(ii) F-GRPO exceeds dedicated rerankers on NDCG. The rerankers are trained on MS MARCO and applied zero-shot, whereas GRPO and F-GRPO are trained in-domain, so we treat them as reference points rather than matched baselines. With either general-purpose backbone, F-GRPO outperforms all dedicated rerankers on NDCG at every cutoff on both MuSiQue and HotpotQA. On MuSiQue, Qwen3-4B also exceeds every reranker at Recall@1 and Recall@3 and is within 3.1 points of the best at Recall@5.

(iii) Decoupled pipelines remain less reliable than factorized RL. The selector-only and full decoupled variants are consistently weak, and even the strongest variant (rank only) remains below F-GRPO on every reported metric, consistent with the gold-slate training mismatch discussed for recommendation. Representative HotpotQA outputs are provided in Appendix G.

5 Analysis

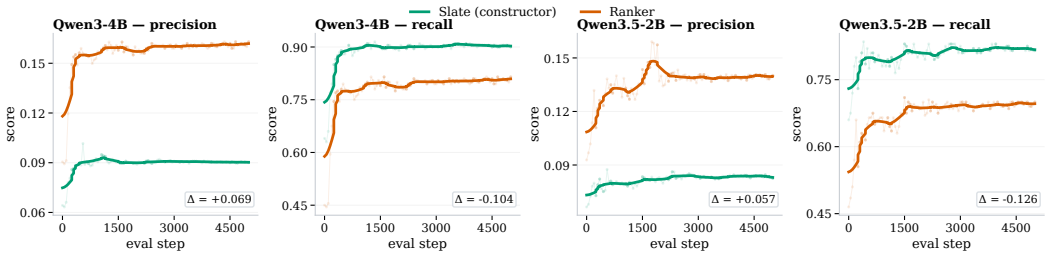


Figure 3: Precision–recall redistribution between the slate and ranker on LastFM across two model sizes. The ranker trades recall for precision by filtering the slate. Appendix F.1 compares ranking quality at a fixed cutoff.

Slate reward formulation. Figure 2a compares F1- and recall-based slate rewards. At k=1, the two perform comparably, but performance with the F1 reward saturates by k=3 on LastFM and k=4 on MovieLens, as its precision penalty discourages proposing candidates beyond the gold set. Performance with the recall reward improves monotonically with k, reaching a +48% relative gain in Recall@5 over the F1 reward on LastFM and +85% on MovieLens.

Phase separation. Figure 2b tracks when each component of the factorized policy matures during training. The slate generator reaches 90% of its peak slate recall by step 150, while the ranker does not reach the same fraction of its peak NDCG until step 200. This ordering is consistent with the conditional structure of the factorized policy (Eq. (1)): the ranker conditions on the slate, so effective ranking depends on the slate generator surfacing relevant candidates.

Precision–recall redistribution. Figure 3 shows a consistent division of labor at both model scales: for Qwen3-4B and Qwen3.5-2B, respectively, the slate maintains recall 0.90/0.82 with precision 0.090/0.082, while ranking raises precision to 0.163/0.140 by filtering the slate. Because a shorter list mechanically raises precision, Appendix F.1 compares both at a fixed cutoff: conditioned on the gold item appearing in the slate, the learned rank improves NDCG@5 over the unreranked slate prefix in all four recommendation settings (see also Appendices F.3 and F.4).

5.1 Ablation Studies

Credit-assignment isolation. Table 3 separates the effect of the two-block output from that of the credit-assignment rule. On LastFM, the two-block formulation (with its structured initialization and two rewards) raises the average from 54.6 to 61.1, and replacing only the shared advantage with separate slate and rank advantages raises it to 62.5 (+2.3% relative). On MuSiQue, the averages are 65.6, 67.2, and 71.1, a larger +5.8% gain from separate credit alone. F-GRPO improves every reported metric over Structured Joint GRPO on both datasets: the two components are complementary, and separate credit matters most when each question requires covering 2–4 gold passages.

Table 3: Credit-assignment isolation on Qwen3-4B. Direct-Rank GRPO emits only a ranked list. Structured Joint GRPO and F-GRPO share the same two-block output, initialization, rewards, and training configuration; they differ only in whether one joint advantage or separate advantages weight the two spans. R/N denote Recall/NDCG; Avg. is the unweighted mean of the six metrics (%).

Method	Output	Credit signal	R@1	R@3	R@5	N@1	N@3	N@5	Avg.
LastFM									
Direct-Rank GRPO	`<RANK>`	one rank reward	44.7	55.1	73.9	44.7	50.8	58.6	54.6
Structured Joint GRPO	`<SLATE>+<RANK>`	one joint advantage	44.7	71.7	80.8	44.7	60.5	64.3	61.1
F-GRPO	`<SLATE>+<RANK>`	separate span advantages	**46.8**	**72.4**	**81.7**	**46.8**	**61.7**	**65.5**	**62.5**
MuSiQue									
Direct-Rank GRPO	`<RANK>`	one rank reward	36.4	63.0	63.0	90.8	71.3	69.2	65.6
Structured Joint GRPO	`<SLATE>+<RANK>`	one joint advantage	36.4	65.3	65.4	90.9	73.6	71.5	67.2
F-GRPO	`<SLATE>+<RANK>`	separate span advantages	**36.8**	**71.3**	**71.8**	**91.6**	**78.5**	**76.5**	**71.1**

Hyperparameter sensitivity. We conduct sensitivity studies on LastFM with Qwen3.5-2B; full results are in Appendix E.3. Underweighting the ranking loss with λ=0.5 consistently degrades top-position quality, while n=5 limits coverage and n=15 dilutes the pool. Across Recall@k and NDCG@k, λ=1.0 and slate size n=10 provide robust defaults (Figure 4).

6 Related Work

LLM-based retrieval and ranking. Generative retrieval methods emit document identifiers directly (Bevilacqua et al., 2022; Wang et al., 2022; Tang et al., 2024). When a candidate pool is available, LLMs serve as rerankers: pointwise (Nogueira et al., 2020), pairwise (Pradeep et al., 2021), or listwise (Sun et al., 2023; Pradeep et al., 2023; Tamber et al., 2023). More recent work optimizes ranking with RL: Neural PG-RANK (Gao et al., 2024) trains a Plackett-Luce ranking policy with REINFORCE, Rank-R1 (Zhuang et al., 2025) applies GRPO to setwise reranking, Rank-GRPO (Zhu et al., 2026) elevates credit assignment to the rank level, and IRPO (Wu et al., 2025a) and Huang et al. (2026a) extend DPO to listwise preferences. Closest to our work are methods that couple generation and ranking: two-stage counterfactual learning (Gupta et al., 2025), GeMS (Deffayet et al., 2023), and HiGR (Pang et al., 2026). However, these approaches either separate the generator and ranker into distinct models or train both stages under a single shared objective, leaving no mechanism to attribute outcome quality to individual phases. F-GRPO instead generates and ranks within a single autoregressive rollout and applies phase-specific credit assignment, so the slate generator and ranker each receive a gradient signal from their own reward.

Reinforcement learning for LLMs. PPO (Schulman et al., 2017) was scaled to LLM alignment in RLHF (Ouyang et al., 2022); subsequent work favors critic-free variants. GRPO (Shao et al., 2024) replaces the critic with group-relative normalization and has been widely adopted for large-scale reasoning-oriented RL training. Refinements include Dr. GRPO (Liu et al., 2025b) (unbiased normalization), DAPO (Yu et al., 2025) (decoupled clipping and token-level loss), and GVPO (Zhang et al., 2025) (provably optimal group weights). Offline alternatives such as DPO (Rafailov et al., 2023; Wang et al., 2026; Huang et al., 2025d; Li et al., 2025b; Huang et al., 2025c; Kveton et al., 2025) optimize directly from pairwise preferences. Concurrent work develops structure-aware credit for interactive trajectories: DC-GRPO (Park et al., 2026) assigns turn-level credit in multi-turn jailbreak learning, while HiPER (Peng et al., 2026) assigns hierarchical credit across planning and execution. F-GRPO instead targets two semantic spans within a single list-to-rank completion. An extended discussion is in Appendix J.

7 Conclusion

We introduced F-GRPO, which trains a single LLM to jointly generate and rank candidates using separate group-relative advantages for each phase. This gives slate construction and ranking distinct credit signals within a shared model while avoiding the distribution mismatch of decoupled pipelines. F-GRPO improves over GRPO and decoupled baselines on recommendation and multi-hop QA, most when candidate coverage is the bottleneck, and a matched ablation shows that separate advantages outperform a shared joint advantage on every reported metric across LastFM and MuSiQue.

AI USE STATEMENT

AI writing tools were used to assist in drafting and verifying the theoretical proofs in this paper and for language editing of the manuscript.

REFERENCES

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683, 2022.

Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten de Rijke. Generative slate recommendation with reinforcement learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, pp. 580–588, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394079. doi: 10.1145/3539597.3570412. URL https://doi.org/10.1145/3539597.3570412.

Ge Gao, Jonathan D. Chang, Claire Cardie, Kianté Brantley, and Thorsten Joachim. Policy-gradient training of language models for ranking, 2024. URL https://arxiv.org/abs/2310.04407.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. Re2G: Retrieve, rerank, generate. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2701–2715, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.194. URL https://aclanthology.org/2022.naacl-main.194/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Shashank Gupta, Yiming Liao, and Maarten de Rijke. Towards two-stage counterfactual learning to rank. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, ICTIR '25, pp. 177–182, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400718618. doi: 10.1145/3731120.3744583. URL https://doi.org/10.1145/3731120.3744583.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06939.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 585–593, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539381. URL https://doi.org/10.1145/3534678.3539381.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part II*, pp. 364–381, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-56059-0. doi: 10.1007/978-3-031-56060-6_24. URL https://doi.org/10.1007/978-3-031-56060-6_24.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=NFM8F5cV0V.

Songwen Hu, Ryan A Rossi, Tong Yu, Junda Wu, Handong Zhao, Sungchul Kim, and Shuai Li. Interactive visualization recommendation with hier-sucb. In *Proceedings of the ACM on Web Conference 2025*, pp. 313–321, 2025b.

Chengkai Huang, Hongtao Huang, Tong Yu, Kaige Xie, Junda Wu, Shuai Zhang, Julian Mcauley, Dietmar Jannach, and Lina Yao. A survey of foundation model-powered recommender systems: From feature-based, generative to agentic paradigms. *arXiv preprint arXiv:2504.16420*, 2025a.

Chengkai Huang, Junda Wu, Yu Xia, Zixu Yu, Ruhan Wang, Tong Yu, Ruiyi Zhang, Ryan A Rossi, Branislav Kveton, Dongruo Zhou, et al. Towards agentic recommender systems in the era of multimodal large language models. *arXiv preprint arXiv:2503.16734*, 2025b.

Chengkai Huang, Junda Wu, Zhouhang Xie, Yu Xia, Rui Wang, Tong Yu, Subrata Mitra, Julian McAuley, and Lina Yao. Pluralistic off-policy evaluation and alignment. *arXiv preprint arXiv:2509.19333*, 2025c.

Hongtao Huang, Chengkai Huang, Junda Wu, Tong Yu, Julian McAuley, and Lina Yao. Listwise preference diffusion optimization for user behavior trajectories prediction. *Advances in Neural Information Processing Systems*, 38:159383–159408, 2026a.

Zihan Huang, Junda Wu, Rohan Surana, Tong Yu, David Arbour, Ritwik Sinha, and Julian McAuley. Image difference captioning via adversarial preference optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33746–33758, 2025d.

Zihan Huang, Rohan Surana, Zhouhang Xie, Junda Wu, Yu Xia, and Julian McAuley. Evaluation on entity matching in recommender systems. *arXiv preprint arXiv:2601.17218*, 2026b.

Branislav Kveton, Xintong Li, Julian McAuley, Ryan Rossi, Jingbo Shang, Junda Wu, and Tong Yu. Active learning for direct preference optimization. *arXiv preprint arXiv:2503.01076*, 2025.

Li Li, Peilin Cai, Ryan A Rossi, Franck Dernoncourt, Branislav Kveton, Junda Wu, Tong Yu, Linxin Song, Tiankai Yang, Yuehan Qin, et al. A personalized conversational benchmark: Towards simulating personalized conversations. *arXiv preprint arXiv:2505.14106*, 2025a.

Xintong Li, Chuhan Wang, Junda Wu, Rohan Surana, Tong Yu, Julian McAuley, and Jingbo Shang. Importance sampling for multi-negative multimodal direct preference optimization. *arXiv preprint arXiv:2509.25717*, 2025b.

Haowei Liu, Xuyang Wu, Guohao Sun, Hsin-Tai Wu, Zhiqiang Tao, and Yi Fang. Ract: Ranking-aware chain-of-thought optimization for llms. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP 2025, pp. 178–188, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400722189. doi: 10.1145/3767695.3769487. URL https://doi.org/10.1145/3767695.3769487.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Sichun Luo, Bowei He, Haohan Zhao, Wei Shao, Yanlin Qi, Yinya Huang, Aojun Zhou, Yuxuan Yao, Zongpeng Li, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. Recranker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Trans. Inf. Syst.*, 43(5), July 2025. ISSN 1046-8188. doi: 10.1145/3705728. URL https://doi.org/10.1145/3705728.

MiniMax. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

Gagan Mundada, Zihan Huang, Rohan Surana, Sheldon Yu, Jennifer Yuntong Zhang, Xintong Li, Tong Yu, Lina Yao, Jingbo Shang, Julian McAuley, et al. Ws-grpo: Weakly-supervised group-relative policy optimization for rollout-efficient reasoning. *arXiv preprint arXiv:2602.17025*, 2026.

Bo Ni, Yu Wang, Leyao Wang, Branislav Kveton, Franck Dernoncourt, Yu Xia, Hongjie Chen, Reuben Luera, Samyadeep Basu, Subhojyoti Mukherjee, Puneet Mathur, Nesreen K. Ahmed, Junda Wu, Li Li, Huixin Zhang, Ruiyi Zhang, Tong Yu, Sungchul Kim, Jiuxiang Gu, Zhengzhong Tu, Alexa Siu, Zichao Wang, Seunghyun Yoon, Nedim Lipka, Namyong Park, Zihao Lin, Trung Bui, Yue Zhao, Tyler Derr, and Ryan A. Rossi. A survey on LLM-based conversational user simulation. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4266–4301, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.200. URL https://aclanthology.org/2026.eacl-long.200/.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL https://aclanthology.org/2020.findings-emnlp.63/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Yunsheng Pang, Zijian Liu, Yudong Li, Shaojie Zhu, Zijian Luo, Chenyun Yu, Sikai Wu, Shichen Shen, Cong Xu, Bin Wang, Kai Jiang, Hongyong Yu, Chengxiang Zhuo, and Zang Li. Higr: Efficient generative slate recommendation via hierarchical planning and multi-objective preference alignment, 2026. URL https://arxiv.org/abs/2512.24787.

Junyoung Park, Namgyu Park, Sechan Lee, Yoon-Chan Jhi, Jihoon Cho, and Sangdon Park. MJ: Multi-turn LLM jailbreaking via decomposed credit assignment. *arXiv preprint arXiv:2607.11070*, 2026. URL https://arxiv.org/abs/2607.11070.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *arXiv preprint arXiv:2602.16165*, 2026. URL https://arxiv.org/abs/2602.16165.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models, 2021. URL https://arxiv.org/abs/2101.05667.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!, 2023. URL https://arxiv.org/abs/2312.02724.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

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

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.

Sahel Sharifymoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. Rankllm: A python package for reranking with llms. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, pp. 3681–3690, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715921. doi: 10.1145/3726302.3730331. URL https://doi.org/10.1145/3726302.3730331.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

Rohan Surana, Junda Wu, Zhouhang Xie, Yu Xia, Harald Steck, Dawen Liang, Nathan Kallus, and Julian McAuley. From reviews to dialogues: Active synthesis for zero-shot llm-based conversational recommender system. *arXiv preprint arXiv:2504.15476*, 2025.

Rohan Surana, Gagan Mundada, Xunyi Jiang, Chuhan Wang, Zhenwei Tang, Difan Jiao, Zihan Huang, Yuxin Xiong, Junda Wu, Sheldon Yu, Xintong Li, Raghav Jain, Nikki Kuang, Sizhe Zhou, Bowen Jin, Zhendong Chu, Tong Yu, Ryan Rossi, Kuan-Hao Huang, Jingbo Shang, Jiawei Han, and Julian McAuley. Generate, filter, control, replay: A comprehensive survey of rollout strategies for llm reinforcement learning, 2026. URL https://arxiv.org/abs/2605.02913.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. URL https://arxiv.org/abs/2602.02710.

Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. Scaling down, litting up: Efficient zero-shot listwise reranking with seq2seq encoder-decoder models, 2023. URL https://arxiv.org/abs/2312.16098.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. Listwise generative retrieval models via a sequential learning process. *ACM Trans. Inf. Syst.*, 42(5), April 2024. ISSN 1046-8188. doi: 10.1145/3653712. URL https://doi.org/10.1145/3653712.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Chuhan Wang, Xintong Li, Jennifer Yuntong Zhang, Junda Wu, Chengkai Huang, Lina Yao, Julian McAuley, and Jingbo Shang. Scenealign: Aligning multimodal reasoning to scene graphs in complex visual scenes. *arXiv preprint arXiv:2601.05600*, 2026.

Lei Wang and Ee-Peng Lim. Zero-shot next-item recommendation using large pretrained language models, 2023. URL https://arxiv.org/abs/2304.03153.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614, 2022.

Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. Coral: Collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 3391–3401, 2024a.

Junda Wu, Hanjia Lyu, Yu Xia, Zhehao Zhang, Joe Barrow, Ishita Kumar, Mehrnoosh Mirtaheri, Hongjie Chen, Ryan A Rossi, Franck Dernoncourt, et al. Personalized multimodal large language models: A survey. *arXiv preprint arXiv:2412.02142*, 2024b.

Junda Wu, Rohan Surana, Zhouhang Xie, Yiran Shen, Yu Xia, Tong Yu, Ryan A. Rossi, Prithviraj Ammanabrolu, and Julian McAuley. In-context ranking preference optimization. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=L2NPhLAKEd.

Junda Wu, Yu Xia, Tong Yu, Xiang Chen, Sai Sree Harsha, Akash V Maharaj, Ruiyi Zhang, Victor Bursztyn, Sungchul Kim, Ryan A Rossi, et al. Doc-react: Multi-page heterogeneous document question-answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 67–78, 2025b.

Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A Rossi, Lina Yao, and Julian McAuley. Sand: Boosting llm agents with self-taught action deliberation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3062–3077, 2025a.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4275–4286, 2025b.

Zhouhang Xie, Junda Wu, Hyunsik Jeon, Zhankui He, Harald Steck, Rahul Jha, Dawen Liang, Nathan Kallus, and Julian McAuley. Neighborhood-based collaborative filtering for conversational recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pp. 1045–1050, 2024.

Zhouhang Xie, Junda Wu, Yiran Shen, Raghav Jain, Yu Xia, Xintong Li, Aaron Chang, Ryan A. Rossi, Tong Yu, Sachin Kumar, Bodhisattwa Prasad Majumder, Jingbo Shang, Prithviraj Ammanabrolu, and Julian McAuley. A survey on personalized and pluralistic preference alignment in large language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=lSWOMjonL7.

An Yan, Zhengyuan Yang, Junda Wu, Wanrong Zhu, Jianwei Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Julian McAuley, Jianfeng Gao, and Lijuan Wang. List items one by one: A new data source and learning paradigm for multimodal LLMs. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=UfWwBaLuXV.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual*

Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. Llamarec: Two-stage recommendation using large language models for ranking, 2023. URL https://arxiv.org/abs/2311.02089.

Kaichen Zhang, Yuzhong Hong, Junwei Bao, Hongfei Jiang, Yang Song, Dingqian Hong, and Hui Xiong. Gvpo: Group variance policy optimization for large language model post-training, 2025. URL https://arxiv.org/abs/2504.19599.

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, 2026. URL https://arxiv.org/abs/2510.20150.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.06034.

A Theoretical Analysis

A.1 Gradient Derivation for Two-Phase Sequence-Level Loss

We derive the gradient of the two-phase loss (Eq. (7)) and show that, at the expansion point $\theta = \theta_{\text{old}}$, it decomposes into two GRPO updates with separate reward weights.

Proposition A.1 (Gradient decomposition). *At $\theta = \theta_{\text{old}}$, the gradient of the combined loss decomposes as $\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}_{\text{slate}} + \lambda \nabla_\theta \mathcal{L}_{\text{rank}} + \beta_{\text{KL}} \nabla_\theta D_{\text{KL}}$, where $\nabla_\theta \mathcal{L}_{\text{slate}}$ is weighted by the slate advantages $\hat{A}^{(i)}_{\text{slate}}$ and $\nabla_\theta \mathcal{L}_{\text{rank}}$ is weighted by the rank advantages $\hat{A}^{(i)}_{\text{rank}}$.*

Proof. For each rollout i, recall that $c^{(i)}_\tau$ and $c^{(i)}_\sigma$ denote the slate and rank content tokens, respectively. The two-phase loss from Eq. (7) is:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{slate}} + \lambda\, \mathcal{L}_{\text{rank}} + \beta_{\text{KL}}\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}). \tag{13}$$

Slate Phase Gradient. The slate loss (Eq. 5) applies the clipped surrogate with per-token ratios $\rho^{(i)}_{\tau,t} = \pi_\theta(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t}) / \pi_{\theta_{\text{old}}}(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t})$ and a uniform advantage $\hat{A}^{(i)}_{\text{slate}}$ across all slate tokens:

$$\mathcal{L}_{\text{slate}} = -\frac{1}{G} \sum_{i=1}^{G} \frac{1}{C} \sum_{t=1}^{|c^{(i)}_\tau|} \min\big(\rho^{(i)}_{\tau,t}\, \hat{A}^{(i)}_{\text{slate}},\ \text{clip}(\rho^{(i)}_{\tau,t},\, 1{-}\epsilon_{\text{clip}},\, 1{+}\epsilon_{\text{clip}})\, \hat{A}^{(i)}_{\text{slate}}\big). \tag{14}$$

At $\theta = \theta_{\text{old}}$, all ratios $\rho^{(i)}_{\tau,t} = 1$, so the clipping is inactive and the gradient simplifies to:

$$\nabla_\theta \mathcal{L}_{\text{slate}}\big|_{\theta=\theta_{\text{old}}} = -\frac{1}{G} \sum_{i=1}^{G} \frac{\hat{A}^{(i)}_{\text{slate}}}{C} \sum_{t=1}^{|c^{(i)}_\tau|} \nabla_\theta \log \pi_\theta(c^{(i)}_{\tau,t} \mid x, c^{(i)}_{\tau,<t}). \tag{15}$$

This is the standard REINFORCE gradient with phase-specific group-relative advantages applied uniformly to all slate tokens and the fixed Dr. GRPO normalizer C defined in Section 3.

Rank Phase Gradient. Analogously, the rank loss (Eq. 6) uses per-token ratios $\rho^{(i)}_{\sigma,t}$ conditioned on the prompt augmented with the generated slate $(x, \tau^{(i)})$:

$$\nabla_\theta \mathcal{L}_{\text{rank}}\big|_{\theta=\theta_{\text{old}}} = -\frac{1}{G} \sum_{i=1}^{G} \frac{\hat{A}^{(i)}_{\text{rank}}}{C} \sum_{t=1}^{|c^{(i)}_\sigma|} \nabla_\theta \log \pi_\theta(c^{(i)}_{\sigma,t} \mid x, \tau^{(i)}, c^{(i)}_{\sigma,<t}). \tag{16}$$

Algorithm 1 Factorized List-to-Rank Training with Phase-Specific Group-Relative Advantages

Require: Training dataset $\mathcal{D}_{\text{train}}$, group size G, clip ϵ_{clip}, KL coef. β_{KL}, trade-off λ, lr α, temperature T, steps M, format penalty p_{fmt}

Initialize policy parameters θ from the designated pretrained or SFT checkpoint; optionally fix reference $\pi_{\text{ref}} \leftarrow \pi_\theta$ if $\beta_{\text{KL}} > 0$
for step $= 1, \ldots, M$ **do**
  Sample minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$
  $\theta_{\text{old}} \leftarrow \theta$ ▷ Snapshot current policy
  for all $x \in \mathcal{B}$ **do** ▷ Rollout collection
    for $i = 1$ to G **do**
      Sample rollout $o^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ with temperature T
      Parse $o^{(i)}$ into slate content $c_\tau^{(i)}$ and rank content $c_\sigma^{(i)}$ via delimiter tags
      if both <SLATE> and <RANK> present **then** ▷ Case 1
        Compute $R_{\text{slate}}^{(i)}$, $R_{\text{rank}}^{(i)}$ from task-specific rewards
      else if only <SLATE> present **then** ▷ Case 2
        Compute $R_{\text{slate}}^{(i)}$; set $R_{\text{rank}}^{(i)} \leftarrow p_{\text{fmt}}$
      else ▷ Case 3: malformed
        $R_{\text{slate}}^{(i)} \leftarrow p_{\text{fmt}}$, $R_{\text{rank}}^{(i)} \leftarrow p_{\text{fmt}}$
      end if
      Apply rank-subset and phase-size reward penalties as specified in Appendices C and E.2
    end for
    Compute group means $\bar{R}_{\text{slate}}$ and $\bar{R}_{\text{rank}}$ over G rollouts
    for $i = 1$ to G **do** ▷ Factorized advantage estimation
      $\hat{A}_{\text{slate}}^{(i)} \leftarrow R_{\text{slate}}^{(i)} - \bar{R}_{\text{slate}}$
      $\hat{A}_{\text{rank}}^{(i)} \leftarrow R_{\text{rank}}^{(i)} - \bar{R}_{\text{rank}}$
    end for
  end for
  Phase 1 (Slate): Compute per-token log-probs $\log \pi_\theta(c_{\tau,t}^{(i)} \mid x, c_{\tau,<t}^{(i)})$ for all collected slate segments
    Apply delimiter masking to tagged outputs (exclude <SLATE>, </SLATE>); use the full raw completion when <SLATE> is missing
    Compute $\mathcal{L}_{\text{slate}}$ via per-token clipped loss (Eq. (5)) with $\hat{A}_{\text{slate}}^{(i)}$
  Phase 2 (Rank): Compute per-token log-probs $\log \pi_\theta(c_{\sigma,t}^{(i)} \mid x, \tau^{(i)}, c_{\sigma,<t}^{(i)})$ for Case 1 rank segments and Case 2 post-slate continuations
    Apply delimiter masking for Case 1; use the raw continuation without delimiter masking for Case 2
    Compute $\mathcal{L}_{\text{rank}}$ via per-token clipped loss (Eq. (6)) with $\hat{A}_{\text{rank}}^{(i)}$
  $\mathcal{L}(\theta) \leftarrow \mathcal{L}_{\text{slate}} + \lambda\, \mathcal{L}_{\text{rank}} + \beta_{\text{KL}}\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$ ▷ Single gradient step per batch
end for
return Trained policy π_θ

Separate reward weights. The slate gradient is weighted by $\hat{A}_{\text{slate}}^{(i)}$, computed from R_{slate} (Section 2), and the rank gradient is weighted by $\hat{A}_{\text{rank}}^{(i)}$, computed from R_{rank} (Section 2). Both gradients update shared parameters θ, so the proposition concerns scalar credit assignment rather than parameter independence. The slate span is trained against coverage quality and the rank span against ranking quality. This completes the proof. □

Relation to standard GRPO. Each phase instantiates the Dr. GRPO variant of the objective in Eq. (3), using mean-subtracted advantages and the fixed normalizer C. The combined loss $\mathcal{L} = \mathcal{L}_{\text{slate}} + \lambda \mathcal{L}_{\text{rank}}$ can therefore be viewed as two GRPO optimizations on shared model parameters, where λ controls the relative importance of ranking quality. The key difference from standard GRPO is that each optimization operates on a *different token subsequence* of the same rollout and uses a *different reward function*, whereas standard GRPO applies a single reward uniformly to all tokens.

Relation to the task-level objective. The full policy gradient of Eq. (2) credits slate actions for both coverage and downstream ranking reward. F-GRPO uses a phase-local surrogate: coverage advantages weight slate tokens and ranking advantages weight rank tokens. It therefore differs from an unbiased full-return gradient, while retaining joint training through the shared parameters.

Table 4: Dataset statistics. Counts reflect the fixed training, validation, and final-evaluation samples used in experiments.

Dataset	Train	Val	Test	Gold Count
Recommendation				
MovieLens	10,000	200	2,000	1 item / 20 candidates
LastFM	10,000	200	2,000	1 item / 20 candidates
Multi-Hop Question Answering				
HotpotQA	10,000	200	2,000	2 passages / 20 candidates
MuSiQue	10,000	200	2,000	2–4 passages / 20 candidates

B Dataset Details

B.1 Dataset Statistics

B.2 Dataset Formulation

We describe how each dataset is adapted for factorized list-to-rank training. All datasets follow the same structure: a context x, a candidate pool of 20 items, binary relevance labels, and a task-specific prompt. Candidates are shuffled to prevent positional shortcuts.

MovieLens. Each sample provides a user history of 10 previously watched movies and a candidate pool of 20 movies (1 true next movie + 19 distractors). Movie titles are normalized to handle year suffixes (e.g., "Toy Story (1995)" $\rightarrow$ "Toy Story").

LastFM. Each sample provides a variable-length user history (3–10 artists) and 20 candidates (1 gold + 19 distractors). The prompt template is identical to MovieLens, with "items" referring to artist names instead of movie titles.

HotpotQA. Each sample provides a multi-hop question and 20 candidate passages (exactly 2 gold supporting passages + 18 distractors), constructed from the HotpotQA distractor split (Yang et al., 2018). Passages are identified by anonymized IDs (e.g., P5, P15) that serve the same role as item names in recommendation. The task requires selecting and ranking the two bridge passages needed to answer the question. The processed release provides train and test splits; we evaluate on a fixed 2,000-example sample of its test split.

MuSiQue. Each sample provides a compositional multi-hop question (Trivedi et al., 2022) and 20 candidate passages (2–4 gold supporting passages + 16–18 distractors). Passages are shuffled and assigned anonymized IDs (P1–P20). MuSiQue provides official train (19,938) and validation (2,417) splits but no separate test split. We subsample 10,000 training examples from the train split and evaluate on a fixed 2,000-example sample of the official validation split.

C Implementation Details

This section describes the key implementation choices that bridge the formal objective (Section 3) to a working training pipeline.

C.1 Structured Output Format

All tasks use the same structured output format with XML-style delimiter tags. The model generates two segments in a single autoregressive pass:

Recommendation. Items are full names (movie/artist titles); the rank orders items by predicted preference:

```
<SLATE>Item A, Item B, Item C, Item D</SLATE>
<RANK>Item C, Item A, Item D, Item B</RANK>
```

Question answering. Items are passage IDs; the rank orders passages by relevance to the query:

```
<SLATE>P2, P5, P8</SLATE>
<RANK>P5, P2, P8</RANK>
```

During parsing, we extract content within tags and apply task-specific normalization for reward computation.

C.2 Delimiter Masking

The two-phase loss (Eq. (7)) requires that delimiter tokens participate in the forward pass (so that content tokens are conditioned on the correct prefix) but are excluded from the loss. We implement this via *position-based masking*: for each delimiter pair (e.g., <SLATE>, </SLATE>), we cache the opening token count N_{open} and closing token count N_{close} at initialization and zero out the loss for the first N_{open} and last N_{close} positions of each segment. Letting T_{seg} denote the total number of tokens in a segment (including delimiters), the effective segment loss is $L_{\text{seg}} = \sum_{t=N_{\text{open}}+1}^{T_{\text{seg}}-N_{\text{close}}} \ell_t$.

C.3 Malformed Output Handling

During training, the model may produce outputs missing one or both delimiter tags. We classify each rollout into one of three cases and assign rewards accordingly:

1. **Case 1** (both <SLATE> and <RANK> present): Normal two-phase processing with computed rewards for both phases.
2. **Case 2** (only <SLATE> present): Slate reward is computed normally; the rank phase receives a format penalty $p_{\text{fmt}} = -1.0$, applied to the raw continuation after </SLATE> (or to the full completion if generation stops immediately).
3. **Case 3** (<SLATE> missing): Both phases receive p_{fmt}, and the slate loss is applied to the full raw completion without delimiter masking.

This classification determines the masking configuration for each rollout. Rollouts within the same case share the same delimiter configuration and are batched together. We quantify the observed formatting behavior across the complete F-GRPO training runs in Table 5.

C.4 Rank Subset Validation

Every ranked item must come from the model's own slate. We enforce this by checking whether every ranked item appears in the unique slate set $\text{uniq}(\tau)$. If not, the rank reward is replaced with the format penalty p_{fmt}, providing a negative learning signal against out-of-slate hallucination.

D Baseline Details

We provide full descriptions of all baselines to facilitate reproduction. All baselines operate on the same candidate pools, data splits, and evaluation metrics as F-GRPO.

D.1 Recommendation Baselines

All recommendation baselines operate on the same 20-candidate ranking task (1 gold + 19 distractors per sample) with identical evaluation (10,000 train / 2,000 test; 200 validation samples held out for model selection).

Random. Candidates are uniformly permuted and the top-5 returned. We average over 5 seeds for stable estimates.

Table 5: Formatting statistics over the 80,000 training completions from each final F-GRPO run. "Both blocks" requires complete, nonempty <SLATE> and <RANK> blocks; "Missing tag" indicates that either block is absent. "Out-of-slate" is the percentage of all completions for which both blocks are present and a ranked item is absent from the slate. Average parsed lengths are computed over completions with both blocks, before oversize penalties. "SFT init." records the initialization used in each main experiment rather than a matched SFT ablation.

Dataset	Backbone	SFT init.	Both blocks (%)	Missing tag (%)	Out-of-slate (%)	Avg. $\|\tau\|$	Avg. $\|\sigma\|$
LastFM	Qwen3-4B	Yes	99.999	0.001	0.170	9.98	5.02
MovieLens	Qwen3-4B	Yes	99.995	0.005	0.169	10.19	5.06
MuSiQue	Qwen3-4B	Yes	100.000	0.000	0.039	2.64	2.40
HotpotQA	Qwen3-4B	Yes	100.000	0.000	0.094	2.30	2.13
LastFM	Qwen3.5-2B	Yes	99.989	0.011	0.450	9.89	4.99
MovieLens	Qwen3.5-2B	Yes	99.886	0.114	0.310	10.02	5.04
MuSiQue	Qwen3.5-2B	No	99.986	0.014	0.155	2.31	2.29
HotpotQA	Qwen3.5-2B	No	99.965	0.035	0.224	2.08	2.08

Popularity. Item frequencies are counted from training answers with Laplace smoothing: $\text{score}(i) = (\text{count}(i) + 1)/(\text{total} + |\mathcal{V}|)$, where $|\mathcal{V}|$ is the number of unique items in the training set. Candidates are ranked by score. This non-personalized baseline measures signal in item popularity alone.

BM25 (Robertson & Zaragoza, 2009). The user's interaction history is concatenated into a query string. Each candidate item name is treated as a document and scored with BM25Okapi. This tests whether surface-level lexical overlap between history and candidate names is informative.

GRU4Rec (Hidasi et al., 2016). GRU-based sequential model following the official implementation. Architecture: constrained embedding (shared input/output weights), 1-layer GRU (100 hidden). Trained for 100 epochs (Adagrad, lr 0.05, batch 64) with cross-entropy loss and popularity-weighted negative sampling (α=0.5). Best checkpoint selected by validation NDCG@5 every 10 epochs. Adapted from session-parallel batching to standard batching (our samples are independent histories).

UniSRec (Hou et al., 2022). Uses BERT-base CLS token embeddings (768D) as item representations, processed through a Mixture-of-Experts adaptor (8 experts, 768→300) and a 2-layer SASRec-style encoder. *Zero-shot*: The pretrained checkpoint (`UniSRec-FHCKM-300`, trained on 5 Amazon domains) is applied directly without fine-tuning. *Fine-tuned*: The MoE adaptor is updated via inductive fine-tuning on the target domain (encoder frozen), trained for 300 epochs with early stopping (patience 10) on validation NDCG@10. Data is converted to RecBole format for fine-tuning.

D.2 QA Reranking Baselines

For the QA tasks (HotpotQA, MuSiQue), we include four reranking-specialized baselines, all applied *zero-shot* (no task-specific training). These baselines are rerankers only: they reorder all given passages without an explicit selection phase. In contrast, our factorized approach first selects a subset via <SLATE>, then ranks within it via <RANK>. We report only ranker metrics for all rerankers (no slate metrics). All models are accessed via the `rank_llm` library (Sharifymoghaddam et al., 2025) (v0.21.0).

RankZephyr (Pradeep et al., 2023). A 7B model built on Zephyr-7B-β (Mistral-based), fine-tuned for listwise reranking via knowledge distillation from GPT-3.5 and GPT-4 on 100K MS MARCO queries. All candidate passages are formatted into a listwise prompt template, and the model outputs a reordered list of identifiers (e.g., `[2] > [5] > [1] > ...`). Since the 20-candidate pools fit within the model's reranking window, each query uses one listwise generation call. Checkpoint: `castorini/rank_zephyr_7b_v1_full`.

MonoT5 (Nogueira et al., 2020). A 3B T5-based model fine-tuned on MS MARCO with a pointwise relevance objective: given a query-passage pair, the model generates "true" or "false," and the softmax probability of "true" serves as the relevance score. Each candidate is scored independently

(N query-passage evaluations, processed in batches), then passages are ranked by descending score. Checkpoint: `castorini/monot5-3b-msmarco-10k`; context size 512, batch size 32.

DuoT5 (Pradeep et al., 2021). A 3B T5-based model from the Expando-Mono-Duo framework, fine-tuned on MS MARCO with a pairwise relevance objective. For each query, all $\binom{N}{2} = 190$ candidate pairs are compared (N=20 for both datasets), and the final ranking is by descending total wins. Checkpoint: `castorini/duot5-3b-msmarco-10k`; context size 512.

LiT5 (Tamber et al., 2023). A T5-based model ($\sim$3B parameters) that uses Fusion-in-Decoder (FiD) for listwise reranking: all candidates are encoded independently and fused during decoding, enabling joint comparison when producing the output permutation. The Distill variant is distilled from GPT-3.5/4 on MS MARCO. Checkpoint: `castorini/LiT5-Distill-xl`; context size 300, window size 20.

Summary of reranking paradigms. Table 6 summarizes the key differences among the four rerankers.

Table 6: Comparison of QA reranking methods. N is the number of candidate passages. Counts refer to scoring units or generation calls per query; runtime also depends on batching, input length, and generated tokens. Dedicated rerankers are trained on MS MARCO and applied zero-shot; our methods are trained in-domain with RL.

Method	Paradigm	Size	Scoring/generation	Training
MonoT5	Pointwise	3B	N pairs	MS MARCO
DuoT5	Pairwise	3B	$\binom{N}{2}$ comparisons	MS MARCO
RankZephyr	Listwise	7B	1 generation	MS MARCO (distilled)
LiT5	Listwise (FiD)	3B	1 generation	MS MARCO (distilled)
GRPO	Generative	4B	1 generation	In-domain (RL)
F-GRPO (ours)	Generative (factorized)	4B	1 generation	In-domain (RL)

D.3 LLM Baselines

Recommendation-side baselines. For recommendation, the compared LLM baselines correspond to prior paradigms: Zero-shot (two-step) follows staged prompting that separates candidate generation from ranking (Wang & Lim, 2023); Zero-shot (single-step) treats the model as a direct black-box generator/ranker (Hou et al., 2024); the baseline SFT follows supervised generative recommendation setups (Luo et al., 2025); Decoupled SFT mirrors two-stage pipelines with separately trained selector and ranker modules (Yue et al., 2023); and all RL variants use Dr. GRPO normalization (Liu et al., 2025b).

Direct-Rank GRPO. This is the GRPO row reported in the main recommendation and QA tables. It uses the rank-only prompt and emits a single `<RANK>` block, with no explicit slate construction. One ranking reward is group-centered into one advantage and applied to the ranked-list tokens. This provides the main rank-only RL comparison; Structured Joint GRPO below supplies the matched credit-assignment control.

Structured Joint GRPO. This controlled ablation matches F-GRPO's initialization, `<SLATE>+<RANK>` output, two reward functions, group size, and all other training hyperparameters. It replaces the separate phase advantages with one group-relative advantage computed from $R^{(i)}_{\text{joint}} = R^{(i)}_{\text{slate}} + \lambda R^{(i)}_{\text{rank}}$ and applies that advantage to both spans. Consequently, the Structured Joint GRPO–F-GRPO comparison changes only the credit-assignment rule (Table 3).

F-GRPO. F-GRPO uses the same structured rollout and rewards as Structured Joint GRPO but applies the slate and rank advantages only to their corresponding token spans, as defined in Section 3. Zero-shot and SFT baselines are described in Appendix I and E.1, respectively.

E TRAINING DETAILS

E.1 SFT WARM-START

RL methods (Direct-Rank GRPO, Structured Joint GRPO, and F-GRPO) are initialized with a supervised fine-tuning (SFT) warm-start, except for Qwen3.5-2B on QA tasks, which starts from the pretrained model directly. The warm-start familiarizes the model with fixed-pool selection and the structured <SLATE>/<RANK> format, reducing sparse feedback from missing tags or hallucinated item names. Across tasks, supervision is used only to construct SFT targets and compute rewards during training; inference receives only the prompt and candidate pool.

E.1.1 FACTORIZED SFT (OURS)

The factorized SFT trains on gold completions with both <SLATE> and <RANK> tags. Given a sample with gold item(s) $\mathcal{G}$ and distractor pool $\mathcal{N}$:

1. Set slate size $n = 10$ (fixed, matching the output size cap `max_slate_items`).
2. Construct slate: all gold items $\mathcal{G}$ plus $\min(n - |\mathcal{G}|, |\mathcal{N}|)$ randomly sampled distractors, then shuffle.
3. Construct rank: gold items first (shuffled), then distractors (shuffled), truncated to `max_rank_items` (default 5).

Example Gold Completion (Recommendation, Factorized)

<SLATE>Dist. F, Dist. B, **Gold**, Dist. A, Dist. C, Dist. H, Dist. D, Dist. G, Dist. E, Dist. I</SLATE>
<RANK>**Gold**, Dist. F, Dist. B, Dist. A, Dist. C</RANK>

E.1.2 BASELINE SFT

The baseline SFT trains on gold completions with only the <RANK> tag:

Example Gold Completion (Recommendation, Baseline)

<RANK>**Gold**, Distractor A, Distractor B, Distractor C, Distractor D</RANK>

Gold items are listed first, followed by randomly sampled distractors, truncated to 5 candidates.

E.1.3 DECOUPLED SFT

The decoupled SFT baseline trains *separate specialist models* for selection and ranking, then chains them at inference via two sequential LLM calls. This compares joint training within a shared backbone (as in F-GRPO) with independently optimized specialists.

Selector training. The selector is trained on <SLATE>-only gold completions using the Step-1 prompts from the two-step zero-shot baseline (Section I). Gold slates contain all gold items plus randomly sampled distractors (shuffled), matching the factorized SFT slate construction.

Ranker training. The ranker is trained on <RANK>-only gold completions using Step-2 prompts conditioned on gold slates. Gold ranks follow the factorized SFT rank construction. Note that the ranker always sees *gold slates* during training.

Inference. At inference, the selector generates a <SLATE>, which is parsed and fed as context to the ranker's Step-2 prompt. The ranker then generates a <RANK>.

Table 7: Shared SFT training hyperparameters for baseline, factorized, and decoupled models.

Training Setup	
Learning rate	2×10^{-5} (3% warmup)
Epochs	1
Optimizer	AdamW
Precision	BF16
Batch size	2×8 accumulation (16)
Max sequence length	2048
Max completion length	256 (Rec/QA)
Training samples	9,800 (+200 held out for validation)
Checkpoint selection	Best eval loss

Table 8: Shared GRPO hyperparameters for Direct-Rank GRPO, Structured Joint GRPO, and F-GRPO.

Training Setup	
Learning rate	5×10^{-6} (3% warmup); 2×10^{-6} for Qwen3.5-2B QA
Batch size	2×8 accumulation (16)
Epochs	1
RL algorithm	Dr. GRPO
Group size	8 rollouts per prompt
Temperature	1.0
Top-p	0.9
Max completion length	256 (Rec/QA)
Clipping	$\epsilon_{\text{clip}} = 0.2$
KL weight	$\beta_{\text{KL}} = 0.0$
Ranking trade-off	$\lambda = 1.0$
Training samples	10k
Output size caps	rec/QA: `max_slate_items` = 10, `max_rank_items` = 5
Oversize penalty	-0.5 per phase (two-item parsing buffer)
Evaluation	Final checkpoint; 2,000 evaluation samples

Three variants.

- **Selector only**: Trained selector + base model ranker (zero-shot).
- **Ranker only**: Base model selector (zero-shot) + trained ranker.
- **Full**: Both selector and ranker trained independently.

This baseline applies to recommendation and QA tasks. Both selector and ranker use *best eval loss* checkpoint selection, matching the other SFT baselines.

E.1.4 SFT HYPERPARAMETERS

All SFT variants share the same training recipe, summarized in Table 7.

The SFT prompt templates match the corresponding rank-only or structured prompts exactly (see Section H). After SFT, the checkpoint is loaded as the starting policy for GRPO training with learning rate 5×10^{-6} (2×10^{-6} for Qwen3.5-2B on QA tasks).

E.2 GRPO HYPERPARAMETERS

Table 8 summarizes the GRPO training configuration.

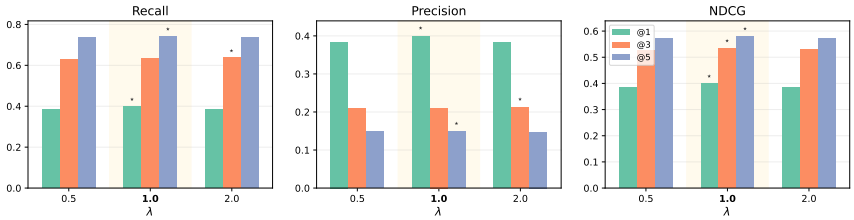
Following recent works (Yu et al., 2025; Liu et al., 2025b; Hu et al., 2025a), we set $\beta_{\text{KL}} = 0$.

The nominal size caps guide generation; the trainer allows a two-item parsing buffer before replacing a phase's reward with the oversize penalty. This accommodates item names containing commas, which the parser can split into extra entries. For the default caps, the penalty applies above 12 parsed slate items or 7 parsed rank items.

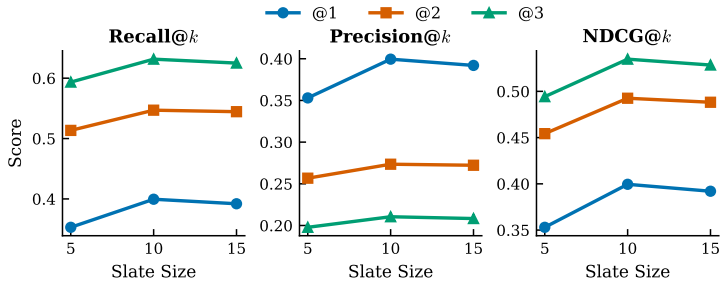
Evaluation uses greedy decoding (T=0, no sampling) with G_{eval}=1 (deterministic single-prediction evaluation) for all tasks.

E.3 Ablation Results

Hyperparameter sensitivity. Figure 4a shows that λ=1.0 yields the best or near-best performance across all metrics; underweighting the ranking loss (λ=0.5) consistently degrades top-position quality, indicating that sufficient rank-loss weight is important. Figure 4b varies the maximum slate size $n \in \{5, 10, 15\}$. All metrics peak at n=10: n=5 limits candidate coverage, while n=15 dilutes the pool with low-relevance items, slightly degrading Precision@1 and NDCG@1. We use λ=1.0 and n=10 as defaults.



(a) Effect of λ on LastFM (Qwen3.5-2B). Stars mark the best setting per cutoff.



(b) Effect of slate size on LastFM (Qwen3.5-2B).

Figure 4: Hyperparameter sensitivity for F-GRPO. (a) λ=1.0 yields the best or near-best performance across all metrics; underweighting the ranking loss (λ=0.5) degrades top-position quality. (b) Performance peaks at slate size n=10; smaller slates limit coverage while larger slates dilute the candidate pool.

F Additional Results

The main text reports Recall@k and NDCG@k. For a ranked list L, let h_k be the number of gold items in its first k positions. Recall@k is $h_k/|\mathcal{G}(x)|$, Precision@k is h_k/k (including when fewer than k items are returned), and Hit@k is $\mathbb{1}[h_k > 0]$. Evaluation NDCG@k divides DCG@k by $\sum_{j=1}^{\min(k,|\mathcal{G}(x)|)} 1/\log_2(1+j)$, using all gold items in the candidate pool rather than only those emitted. Metrics are averaged over examples. For completeness, we include the remaining metrics here. In recommendation, Hit@k equals Recall@k because each example has one gold item. Table 9 reports Precision@k for recommendation (LastFM and MovieLens).

Table 10 reports Precision@k and Hit@k for multi-hop QA (MuSiQue and HotpotQA). The trends are consistent with those reported in the main text: F-GRPO achieves the best or second-best performance among LLM-based methods across all cutoffs.

F.1 Conditional Rank-Phase Analysis

Because recommendation has one gold item, comparing precision of a ten-item slate with that of a five-item rank partly reflects the change in denominator. We therefore condition on the gold item

Table 9: Precision@k on LastFM and MovieLens for Qwen3-4B and Qwen3.5-2B. All values are percentages. Within each backbone block, **bold** marks the best value and underlining the second best.

	LastFM			MovieLens		
Method	@1	@3	@5	@1	@3	@5
Random	5.0	5.0	5.0	4.9	5.1	5.0
Popularity	24.7	15.3	11.3	14.4	12.1	11.0
BM25	6.2	5.9	5.5	8.7	6.1	5.5
GRU4Rec	50.8	23.0	15.1	21.8	15.6	12.0
UniSRec (zero-shot)	11.0	8.5	7.4	6.3	5.5	5.1
UniSRec (fine-tuned)	23.9	14.1	10.9	13.6	9.9	8.7
Qwen3-4B						
Zero-shot (two-step)	18.8	14.9	11.7	12.1	9.7	8.0
Zero-shot (single-step)	20.6	14.6	11.2	11.9	8.9	7.8
SFT	40.1	15.7	10.8	21.5	9.8	7.3
Decoupled SFT (sel. only)	14.1	11.1	9.2	10.1	8.4	7.2
Decoupled SFT (rank. only)	28.8	14.3	10.4	16.5	9.3	7.4
Decoupled SFT (full)	33.4	13.5	9.2	19.7	9.3	7.1
GRPO	44.7	18.4	14.8	23.9	14.8	11.5
F-GRPO (ours)	**46.8**	**24.1**	**16.3**	**26.9**	**17.9**	**13.3**
Qwen3.5-2B						
Zero-shot (two-step)	0.6	0.5	0.5	1.2	0.6	0.4
Zero-shot (single-step)	3.1	2.2	1.6	3.3	1.5	1.1
SFT	37.9	15.1	10.5	**27.7**	11.6	8.2
Decoupled SFT (sel. only)	0.5	0.4	0.4	0.7	0.6	0.5
Decoupled SFT (rank. only)	23.1	10.5	7.7	12.3	6.9	5.4
Decoupled SFT (full)	32.4	13.4	9.2	23.6	10.5	8.1
GRPO	38.7	20.1	14.3	24.4	**16.9**	11.3
F-GRPO (ours)	**40.0**	**21.1**	**14.9**	21.5	15.3	**12.2**

Table 10: Precision@k and Hit@k on MuSiQue and HotpotQA for Qwen3-4B and Qwen3.5-2B. All values are percentages. Within each backbone block, **bold** marks the best value and underlining the second best.

	MuSiQue						HotpotQA					
	Precision@k			Hit@k			Precision@k			Hit@k		
Method	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
Random	13.3	13.3	13.2	13.3	36.0	53.8	9.9	10.0	10.0	9.9	28.5	44.7
BM25	57.0	36.5	27.7	57.0	81.5	89.6	83.6	48.1	33.1	83.6	96.6	98.6
RankZephyr	82.3	51.2	36.6	82.3	95.0	97.6	94.8	56.6	36.1	94.8	98.9	99.6
MonoT5	86.1	54.8	38.6	86.1	97.9	99.4	92.5	57.5	37.1	92.5	99.0	99.8
DuoT5	84.8	53.2	37.9	84.8	96.7	99.1	86.4	53.2	34.5	86.4	92.6	94.9
LiT5	82.7	45.8	32.6	82.7	94.3	96.8	92.5	52.2	34.1	92.5	98.3	99.2
Qwen3-4B												
Zero-shot (two-step)	10.2	7.3	4.6	10.2	11.8	11.9	6.0	3.9	2.4	6.0	6.3	6.4
Zero-shot (single-step)	58.4	41.0	29.2	58.4	70.4	72.6	61.4	40.6	26.1	61.4	68.2	69.6
SFT	69.5	46.3	31.4	69.5	91.3	94.5	76.6	45.8	29.2	76.6	90.1	92.5
Decoupled SFT (sel. only)	21.7	13.9	9.6	21.7	26.1	26.9	19.6	12.8	8.2	19.6	22.0	22.4
Decoupled SFT (rank. only)	79.8	53.6	35.3	79.8	95.1	96.9	87.6	56.2	35.7	87.6	97.6	99.0
Decoupled SFT (full)	60.6	33.5	23.0	60.6	75.4	80.4	65.1	32.6	21.1	65.1	76.4	79.4
GRPO	90.8	54.0	32.4	90.8	96.9	96.9	**96.1**	**62.0**	**37.2**	**96.1**	**99.4**	**99.5**
F-GRPO (ours)	**91.6**	**60.8**	**36.8**	**91.6**	**97.8**	**97.8**	96.0	61.7	37.0	96.0	**99.4**	99.4
Qwen3.5-2B												
Zero-shot (two-step)	5.4	3.4	2.4	5.4	7.0	7.5	2.0	1.3	0.8	2.0	2.4	2.5
Zero-shot (single-step)	13.2	9.2	6.5	13.2	17.8	18.9	8.0	5.0	3.4	8.0	9.6	9.9
SFT	34.4	25.1	19.8	34.4	61.8	72.9	10.4	10.0	10.1	10.4	28.6	45.6
Decoupled SFT (sel. only)	0.7	0.4	0.4	0.7	1.0	1.3	0.5	0.4	0.3	0.5	0.9	0.9
Decoupled SFT (rank. only)	43.7	31.4	25.4	43.7	71.5	84.2	42.8	34.5	27.4	42.8	75.4	87.6
Decoupled SFT (full)	13.8	13.1	13.0	13.8	35.5	52.2	11.9	11.1	10.3	11.9	31.3	45.4
GRPO	**90.0**	53.1	32.9	**90.0**	97.3	**98.3**	96.1	57.6	34.6	96.1	**99.1**	**99.2**
F-GRPO (ours)	89.9	**62.1**	**37.7**	89.9	**97.8**	97.8	**96.3**	**61.0**	**36.6**	**96.3**	**99.1**	99.1

appearing in the generated slate and compare the model's ranked top five with three controls: the first five slate items without reranking, a nominal uniform random five-of-ten reference, and an oracle that places the gold first. The learned rank improves conditional NDCG@5 over the unreranked slate

Table 11: Rank-phase isolation on held-out validation examples at the final checkpoint (200 examples per setting). NDCG@5 conditions on the gold item appearing in the generated slate. "Slate first 5" truncates the slate without reranking; "Random 5/10" is the analytic expectation for a uniformly ordered five-item sample from the nominal ten-item slate; the oracle places the gold item first. All values are percentages.

Backbone	Dataset	Random 5/10	Slate first 5	Actual rank	Δ Rank–Slate	Oracle
Qwen3-4B	LastFM	29.5	66.5	**70.3**	+3.8	100.0
Qwen3-4B	MovieLens	29.5	59.6	**61.4**	+1.8	100.0
Qwen3.5-2B	LastFM	29.5	64.8	**65.7**	+0.9	100.0
Qwen3.5-2B	MovieLens	29.5	51.0	**53.7**	+2.8	100.0

Table 12: MuSiQue results for Qwen3.5-2B with 50 candidates and a maximum slate size of 10. The 30 additional candidates are random cross-question distractors. R@k and N@k denote Recall@k and NDCG@k, respectively; Avg. is the unweighted average of the six metrics. All values are percentages.

Method	R@1	R@3	R@5	N@1	N@3	N@5	Avg.
GRPO	35.8	59.8	61.4	89.3	68.5	67.5	63.7
F-GRPO	**36.1**	**68.5**	**68.9**	**89.4**	**75.8**	**73.9**	**68.8**

prefix in all four dataset–backbone settings, showing that the rank phase contributes beyond slate construction. Both lists are evaluated at the same cutoff with the same gold-item normalization, so the comparison does not benefit mechanically from shortening the output list.

F.2 Larger Candidate Pool

To test sensitivity to candidate-pool size, we augment each MuSiQue instance's original 20 candidates with 30 distractor passages sampled at random from other questions in the same split, yielding 50 candidates in total. We keep the maximum slate size fixed at 10 and otherwise use the same evaluation protocol as in the main experiments. As shown in Table 12, F-GRPO improves the average of the six ranking metrics by 7.9% relative to GRPO. Because the additional passages are random cross-question distractors rather than retrieval-based hard negatives, this experiment establishes robustness to a larger random-distractor pool, not to harder retrieval negatives.

F.3 Optimization Dynamics

Figure 5 compares F-GRPO and GRPO evaluation metrics over training on Qwen3-4B. F-GRPO converges to higher values on most metrics on both datasets, with the clearest gains at higher cutoffs. The advantage is most pronounced on Recall@5 and NDCG@5, where the factorized slate's broader coverage compounds with ranking quality: on MovieLens, F-GRPO reaches Recall@5$\approx$0.70 while GRPO plateaus at$\approx$0.60. F-GRPO also converges faster than rank-only GRPO, achieving strong performance within the first 1,000 steps on LastFM, while GRPO requires$\approx$2,000 steps to reach comparable Recall@1. This faster convergence is consistent with the credit-assignment analysis in Section 3: separate span weights provide more targeted learning signals to slate and rank tokens. The matched Structured Joint GRPO ablation in Table 3 further shows that the benefits extend beyond the two-block output structure.

F.4 Error Attribution

We decompose F-GRPO's test errors into two failure modes: *slate miss* (the gold item never appears in the slate) and *rank drop* (the gold item appears in the slate but the ranker fails to surface it).

On both datasets, errors arise from both slate misses and rank drops, confirming that final performance depends on both candidate coverage and within-slate ranking. The matched credit-assignment ablation in Table 3 directly establishes the benefit of optimizing the two phases with separate advantages.

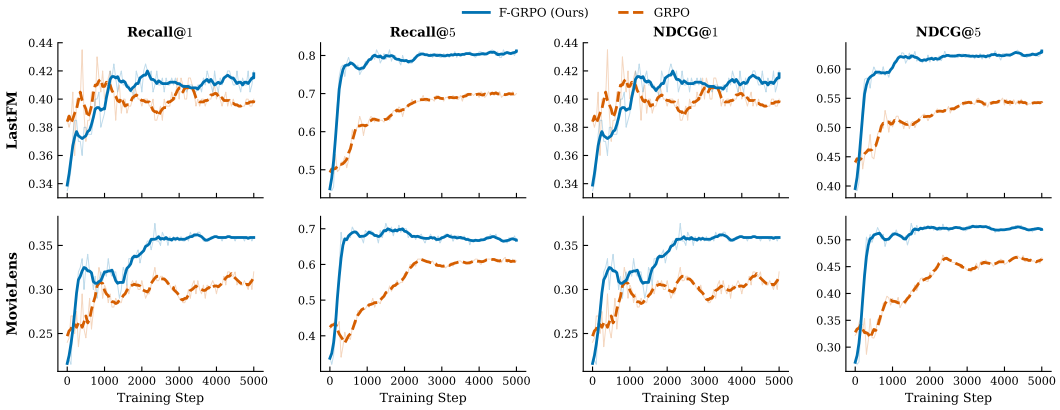


Figure 5: Evaluation metrics during training for F-GRPO and GRPO on Qwen3-4B. F-GRPO converges faster and reaches higher values on most metrics, with the clearest gains at higher cutoffs. Faint lines show raw values, bold lines are smoothed.

Table 13: Error attribution for F-GRPO (Qwen3-4B). Success means the gold item appears in the emitted ranked list. Slate misses and rank drops both contribute to unsuccessful predictions.

Dataset	Slate miss	Rank drop	Success
LastFM	8.9%	9.4%	81.8%
MovieLens	17.9%	15.4%	66.6%

G Representative Examples

We provide two representative HotpotQA examples from Qwen3-4B. For readability, we show passage titles rather than anonymized IDs (P1–P20).

HotpotQA Example 1 (Qwen3-4B)

Question: Shahnoza Nazirova traveled to Icheon and Ansan, South Korea to compete in what sport and event?

Gold evidence: Shahnoza Nazirova; Volleyball at the 2014 Asian Games – Women

F-GRPO <SLATE>: `Shahnoza Nazirova; Volleyball at the 2014 Asian Games -- Women; Icheon`

F-GRPO <RANK>: `Shahnoza Nazirova; Volleyball at the 2014 Asian Games -- Women`

Baseline <RANK>: `Icheon; Shahnoza Nazirova; Ansan`

HotpotQA Example 2 (Qwen3-4B)

Question: What Russian city are both Mariinsky Ballet and Arthur Saint-Léon located?

Gold evidence: Mariinsky Ballet; Arthur Saint-Léon

F-GRPO <SLATE>: `Mariinsky Ballet; Mariinsky Theatre; Arthur Saint-Léon`

F-GRPO <RANK>: `Mariinsky Ballet; Arthur Saint-Léon`

Baseline <RANK>: `Mariinsky Ballet; Larissa Lezhnina; Mariinsky Theatre`

H PROMPT TEMPLATES

We reproduce the full prompt templates used for all tasks. The factorized prompt instructs the model to output both a slate and a ranking in a single pass, wrapped in `<SLATE>...</SLATE>` and `<RANK>...</RANK>` tags; the baseline prompt uses only `<RANK>` tags. Both include explicit size guidance ("up to 10" for the slate, "top 5" for the rank), matching the output size caps enforced during training (Appendix E.2).

H.1 RECOMMENDATION (MOVIELENS & LASTFM)

FACTORIZED MODE

Recommendation Factorized Prompt Template

System: You are a recommendation system. Given a user's consumption history, predict which items they are most likely to select next from the provided candidates.

User:

```
User history:
{history}
Candidate Items:
{candidates}
```

Based on the user's history, predict which items they are likely to select next.
You MUST follow this exact format. Output ONLY the tagged blocks.
Rules:

- SLATE: Select up to {max_slate} items the user would most likely choose next.
- RANK: Rank your top {max_rank} items from SLATE, from most to least likely to be their next choice.
- Use ONLY item names from the candidates list (no paraphrases).

Output format:

```
<SLATE>Item A, Item B, Item C, Item D</SLATE>
<RANK>Item B, Item A, Item D, Item C</RANK>
```

BASELINE MODE

Recommendation Baseline Prompt Template

System: You are a recommendation system. Given a user's consumption history, predict which items they are most likely to select next from the provided candidates.

User:

```
User history:
{history}
Candidate Items:
{candidates}
```

Based on the user's history, predict which items they are likely to select next.
You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- RANK: Select and rank the top {max_rank} items from most to least likely to be their next choice.
- Use ONLY item names from the candidates list (no paraphrases).

Output format:

```
<RANK>Item B, Item A, Item D, Item C</RANK>
```

H.2 Question Answering (HotpotQA & MuSiQue)

Factorized Mode

QA Factorized Prompt Template

System: You are an evidence selection system. Given a question and candidate passages (each with an ID and content), identify which passage IDs contain supporting evidence.

User:
`Query: {question}`
`Candidate Passages:`
`{candidates}`
You MUST follow this exact format. Output ONLY the tagged blocks.
Rules:

- SLATE: Select up to {max_slate} passage IDs that contain relevant evidence.
- RANK: Rank your top {max_rank} passage IDs from SLATE, from most to least relevant.
- Use ONLY the passage IDs listed above (do not invent IDs).

Output format:
`<SLATE>P1, P2, P3</SLATE>`
`<RANK>P2, P1, P3</RANK>`

Baseline Mode

QA Baseline Prompt Template

System: You are an evidence selection system. Given a question and candidate passages (each with an ID and content), identify which passage IDs contain supporting evidence.

User:
`Query: {question}`
`Candidate Passages:`
`{candidates}`
You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- RANK: Select and rank the top {max_rank} passage IDs from most to least relevant.
- Use ONLY the passage IDs listed above (do not invent IDs).

Output format:
`<RANK>P2, P1, P3</RANK>`

Note. Candidate passages are formatted as `**P1**` followed by title and text content. For both HotpotQA and MuSiQue, passages are numbered P1–P20, and each passage's Wikipedia title appears in its text.

I Two-Step Zero-Shot Prompt Templates

The two-step zero-shot baseline decomposes the list-to-rank task into two sequential LLM calls without any training: Step 1 generates a `<SLATE>` (candidate selection), and Step 2 feeds the generated slate back as context to produce a `<RANK>` (ranking). Both steps use greedy decoding (T=0) with a single rollout per prompt. System prompts are shared with the factorized templates above; we show only the user messages.

I.1 RECOMMENDATION

Recommendation: Step 1 (Slate Generation)

`User history:`
`{history}`
`Candidate Items:`
`{candidates}`
Based on the user's history, select items they are most likely to choose next.
You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- SLATE: Select up to {max_slate} items the user would most likely choose next.
- Use ONLY item names from the candidates list (no paraphrases).

Output format:
`<SLATE>Item A, Item B, Item C, Item D</SLATE>`

Recommendation: Step 2 (Ranking)

`User history:`
`{history}`
`Candidate Items:`
`{candidates}`
Based on the user's history, you previously selected these items as likely next choices: {slate_items}
You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- RANK: Rank the top {max_rank} items from most to least likely.
- Use ONLY item names from your selection (no paraphrases).

Output format:
`<RANK>Item B, Item A, Item D, Item C</RANK>`

I.2 QUESTION ANSWERING

QA: Step 1 (Slate Generation)

`Query: {question}`
`Candidate Passages:`
`{candidates}`
You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- SLATE: Select up to {max_slate} passage IDs that contain relevant evidence.
- Use ONLY the passage IDs listed above (do not invent IDs).

Output format:
`<SLATE>P1, P2, P3</SLATE>`

QA: Step 2 (Ranking)

```
Query: {question}
Candidate Passages:
{candidates}
```

Based on the query, you previously selected these passage IDs as containing relevant evidence: {slate_items}

You MUST follow this exact format. Output ONLY the tagged block.
Rules:

- RANK: Rank the top {max_rank} passage IDs from most to least relevant.
- Use ONLY passage IDs from your selection (do not invent IDs).

Output format:
`<RANK>P2, P1, P3</RANK>`

J Extended Related Work

This section expands on the discussion in Section 6, providing additional context for each line of work.

J.1 LLM-based Retrieval and Reranking

Recent work on LLM-based retrieval broadly splits into generative retrieval and reranking over a fixed candidate pool. In generative retrieval, language models directly emit document identifiers rather than independently scoring a corpus: Autoregressive Search Engines (Bevilacqua et al., 2022) generates corpus-constrained n-gram substrings as identifiers using an FM-Index, Neural Corpus Indexer (Wang et al., 2022) learns hierarchical semantic document identifiers via clustering, and ListGR (Tang et al., 2024) extends the formulation with sequential, position-aware listwise training so the generation objective better matches ranked evaluation metrics.

When a candidate set is already available, LLMs have been applied as increasingly strong rerankers. MonoT5 (Nogueira et al., 2020) and the Expando-Mono-Duo framework (Pradeep et al., 2021) showed that seq2seq models serve as effective pointwise and pairwise rerankers within classical multi-stage pipelines. Subsequent work moved toward listwise formulations. RankGPT (Sun et al., 2023) showed that proprietary LLMs can perform zero-shot listwise reranking competitive with supervised methods, and subsequent open-weight efforts distill this capability into smaller models: RankZephyr (Pradeep et al., 2023) into a 7B model and LiT5 (Tamber et al., 2023) into 220M–3B parameters via a Fusion-in-Decoder architecture.

More recent methods optimize ranking objectives directly with RL or preference learning. Neural PG-RANK (Gao et al., 2024) casts retrieval as a Plackett-Luce ranking policy trained with REINFORCE. RaCT (Liu et al., 2025a) combines chain-of-thought supervised fine-tuning with ranking preference optimization. Rank-R1 (Zhuang et al., 2025) applies GRPO to a setwise reranker that selects the single most relevant document per call, eliciting reasoning without explicit supervision. Rank-GRPO (Zhu et al., 2026) adapts GRPO for conversational recommendation by elevating credit assignment from the token level to individual rank positions within a single list. IRPO (Wu et al., 2025a) extends DPO to listwise ranking with a positional preference model inspired by Plackett-Luce.

Closest to our setting are methods that treat candidate generation and final ordering as coupled decisions rather than isolated stages. Two-stage counterfactual learning to rank (Gupta et al., 2025) jointly optimizes a candidate generator and ranker under logged feedback via alternating updates, while GeMS (Deffayet et al., 2023) and HiGR (Pang et al., 2026) optimize whole recommendation slates rather than independent items, using latent-space RL and hierarchical planning with listwise preference alignment, respectively. Our work follows this direction but realizes both phases within a single autoregressive LLM rollout: the same model first constructs a slate and then ranks it, and the two output spans receive separate group-relative signals.

J.2 Reinforcement Learning for LLMs

Reinforcement learning for LLM post-training largely builds on policy-gradient methods. PPO (Schulman et al., 2017), with its clipped surrogate objective, was scaled to LLM alignment in RLHF (Ouyang et al., 2022), where a learned reward model guides policy updates. Recent reasoning-focused work has increasingly favored critic-free variants that avoid learning a separate value model. Shao et al. (2024) introduced GRPO, which replaces the critic with group-relative reward normalization over multiple samples from the same prompt, making large-scale RL more practical for verifiable reasoning tasks; Guo et al. (2025) subsequently scaled this approach to frontier reasoning.

Subsequent work mainly refines how credit is assigned within this critic-free framework. Dr. GRPO (Liu et al., 2025b) identifies question-difficulty and response-length biases from reward standardization and per-response length normalization, replacing them with mean-centered rewards and a fixed length normalizer. GVPO (Zhang et al., 2025) revisits the weighting rule more fundamentally, deriving group weights from the KL-constrained reward-maximization problem so that the intractable partition function cancels, yielding provably optimal updates. DAPO (Yu et al., 2025) shows that large-scale reasoning RL is highly sensitive to training recipe choices, improving long-chain-of-thought training with decoupled clipping, dynamic sampling, token-level policy loss, and overlong reward shaping. MiniMax-M1 (MiniMax, 2025) introduces CISPO, which clips importance-sampling weights rather than zeroing out per-token gradient contributions, preserving entropy and stabilizing training for long generations. DC-GRPO (Park et al., 2026) combines immediate and future deviations for turn-level credit in interactive trajectories, while HiPER (Peng et al., 2026) assigns hierarchical advantages across planning and execution segments and analyzes estimator bias and variance. F-GRPO focuses on a complementary within-completion structure: a slate span followed by a rank span, with each span trained by a separate group-relative signal.

Alongside these online RL methods, work on offline and alternative objectives has broadened the design space. DPO (Rafailov et al., 2023) reparameterizes the reward under a Bradley-Terry preference model so that the partition function cancels, enabling direct optimization from pairwise preferences without an explicit reward model or RL loop. MaxRL (Tajwar et al., 2026) argues that expected-reward RL optimizes only a first-order approximation to the underlying maximum-likelihood objective over correct rollouts, and proposes a compute-indexed family of objectives that interpolate toward exact likelihood as sampling budget grows. Our method is complementary: it focuses on a list-to-rank rollout containing two coupled decisions and assigns separate group-relative signals to candidate generation and ranking.