

# R2QRec: Learning Recommendation Queries from Retrieval Feedback

Weixin Chen<sup>1</sup>, Shuchang Liu<sup>1†</sup>, Hailan Yang<sup>1</sup>, Yuhao Zhao<sup>1,2</sup>, Li Kang<sup>2</sup>, Xuefeng Zhang<sup>1</sup>, Chenghao Zhang<sup>1</sup>, Xinyuan Wang<sup>1</sup>, Hanchen Yang<sup>1</sup>, Yinghao Wu<sup>1</sup>, Hua Huang<sup>1</sup>, Yandong Bai<sup>1</sup>, Fan Yang<sup>1</sup>, Xiang Li<sup>1</sup>, Chunjie Chen<sup>1</sup>, Li Chen<sup>2</sup>

<sup>1</sup>Kuaishou Technology <sup>2</sup>Hong Kong Baptist University

## Abstract

Query-based recommendation separates expressing user needs from locating matching items. Its central learning problem is how to turn item interactions into effective retrieval requests when interaction logs do not annotate queries. We present R2QRec, which uses a fixed retrieval system to construct query supervision and evaluate recommendation outcomes. Retrieval verification first selects generated expressions that recover observed items as supervised fine tuning targets, allowing retrieval behavior to determine the target wording. Policy refinement then evaluates generated query sets by the observed item’s rank in their fused recommendation list. Log-rank feedback distinguishes outcomes beyond the recommendation cutoff, while a KL penalty constrains policy change relative to the supervised reference. The stages have distinct roles: verification identifies expressions to imitate, and policy refinement improves the combined ranking produced by newly generated queries. Only the query generator is trained; the retriever and item index remain fixed, and no query annotations or learned reward model are required. Across three public datasets, R2QRec achieves state-of-the-art overall recommendation performance among the evaluated methods. Extensive experiments demonstrate the effectiveness of the proposed design and its generalization across generators and domains.

## Keywords

recommender systems, query generation, retrieval feedback, reinforcement learning, large language models

## 1 Introduction

Recommendation on Web platforms requires finding items that satisfy a user’s needs, even when those needs do not specify a concrete retrieval request. Interaction histories and other available user context provide evidence of preferences, inducing several plausible directions for the next recommendation. Existing approaches express these preferences through discriminative item scores [7, 13] or distributions on discrete item identifiers [4, 20].

Query-based recommendation makes this intermediate preference expression explicit in natural language, then uses a retriever to find matching items across the catalog [6, 16]. A single user can have multiple interests [14]. For example, a reading history containing *Thinking, Fast and Slow*, *The Power of Habit*, and *Outliers* may suggest interests in cognitive science, behavioral change, or human achievement. Large language models (LLMs) can draw on semantic knowledge to turn these plausible interests into separate retrieval requests, with prior work demonstrating the value of

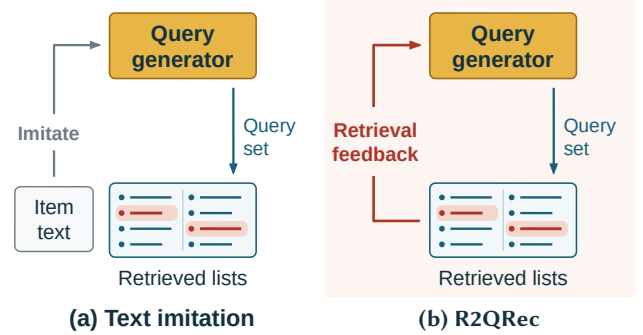


**Figure 1: Learning to generate recommendation queries. (a) Text imitation supervises the generator with an observed item’s text. (b) R2QRec uses the observed item’s retrieval outcomes to verify queries and optimize the generator. Red denotes retrieval feedback and the observed item.**

multiple queries for recommendation [16]. This provides an open-ended language representation of preferences without requiring the generator to select an item code or enumerate a candidate list.

Title-supervised and prompting-based recommenders provide two ways to obtain retrieval text. GPT4Rec trains its generator to predict target item titles before using the generated text for retrieval [16]; BIGRec also uses target item text for supervised adaptation [1]. QueRec [6] generates personalized queries through prompting without LLM fine-tuning. We study a complementary learning formulation: next-item interactions supply query supervision and evaluate the recommendation outcomes of generated queries (Figure 1).

An observed interaction identifies an item to recover, but does not prescribe a unique query to reproduce. Several expressions may retrieve the same item, while the interaction log does not annotate which expression to imitate. We address this supervision problem through *item-verified supervision*: generated expressions that retrieve the observed item successfully become training targets. The item label thereby specifies an admissibility criterion for queries without fixing their wording.

Admissible individual expressions and useful query combinations pose different ranking problems. Even the complete vector of target ranks across queries does not determine the fused recommendation outcome: the positions of competing items across the lists also matter. Our analysis formalizes this distinction, motivating *fusion-aware refinement* of generated sets using the observed item’s rank after reciprocal rank fusion. Log-rank feedback then supplies

<sup>†</sup>Corresponding author: liushuchang@kuaishou.com

relative comparisons among sets for the same context [23]. Its cutoff-risk decomposition explains how it emphasizes improvements near the head while retaining feedback beyond a fixed recommendation cutoff.

These choices define **R2QRec**, a compact framework linking item-level supervision to recommendation-level evaluation. Item-verified supervision constructs the expressions to imitate; fusion-aware refinement evaluates newly generated combinations; the supervised reference constrains policy change through a KL penalty. The same frozen retrieval interface supplies both supervision and outcome evaluation, while only the query generator is updated. This makes recommendation queries a learned interface between user context and item retrieval.

At inference, the generator expresses user preferences as several queries, the retriever executes them, and reciprocal rank fusion combines the resulting item lists [3]. The item index and retrieval model remain fixed, and the learning procedure requires neither annotated queries nor a separate learned reward model.

We evaluate R2QRec on three public recommendation datasets and observe state-of-the-art overall recommendation performance among the evaluated methods. Controlled MovieTV comparisons trace title supervision and retrieval-verified supervision through SFT and subsequent policy refinement. These comparisons connect the design choices to their effect on recommendation quality; further experiments examine multiple queries, model choices, and domain transfer.

Our contributions are summarized as follows:

- **Item-supervised query learning.** We formulate query supervision through the retrieval admissibility of generated expressions, allowing an observed interaction to supervise multiple phrasings without query annotations.
- **Fusion-aware refinement and analysis.** We show that individual target ranks do not determine fused recommendation utility, motivating feedback on query combinations. A cutoff-risk decomposition characterizes the depth weighting of the log-rank objective.
- **Controlled evidence for the learning design.** Experiments distinguish the effect of supervision source from subsequent policy refinement and evaluate recommendation quality across three domains.

## 2 Preliminaries

Let  $\mathcal{U}$  and  $\mathcal{I}$  denote the user and item sets. For user  $u$ , let  $H_u$  denote the chronological interaction history. The user context  $s_u$  represents the user's recommendation needs, primarily through their interaction history. Given  $s_u$ , the system retrieves and ranks items from  $\mathcal{I} \setminus H_u$  as recommendations, where the held-out next item  $i^*$  serves as the target.

Interaction logs provide item targets without annotated queries. For a chronological sequence  $(i_1, \dots, i_n)$ , the final interaction  $i_n$  is reserved for evaluation. An earlier interaction  $i_t$  can serve as a training target with prefix history  $H_u^{(t)} = (i_1, \dots, i_{t-1})$ . Let  $\mathbf{s}_u^{(t)}$  denote the available context formed from this prefix. The training examples are drawn from

$$\mathcal{D} \subseteq \bigcup_{u \in \mathcal{U}} \{(\mathbf{s}_u^{(t)}, i_t) \mid 2 \leq t < n\}. \quad (1)$$

## 3 R2QRec

### 3.1 Query Generation and Retrieval

R2QRec learns a query generator  $Q_\phi(q \mid s_u)$  from item retrieval outcomes while keeping the retriever  $R(i \mid q)$  fixed (Figure 2). The generator expresses the user context as natural-language retrieval queries. We denote a generated query set by  $q = (q_1, \dots, q_K)$ , where each  $q_j$  defines a retrieval request. At inference, we infer  $K$  queries and combine their retrieval results by rank fusion. The query budget  $K$  can be varied at inference without changing the learned parameters.

The frozen retriever  $R$  maps each query to a ranked list over  $\mathcal{I} \setminus H_u$ . Let  $\text{rank}_R(i \mid q_j)$  denote the item's rank in this list. Queries and item documents are encoded in the same semantic embedding space, and item embeddings can be cached. While the generator learns to express preferences in free-form text, it retrieves relevant items in this embedding space. Finally, Reciprocal Rank Fusion (RRF) [3] combines the  $K$  retrieval lists. We return the top- $N$  items in the fused ranking as recommendations. Let  $\rho_F(i \mid q)$  denote the rank of item  $i$  after applying this fusion to the query set  $q$ . We use the same retrieval pool, history masking, fusion settings, and tie handling when evaluating training query sets.

### 3.2 Training Objective

The two stages connect different levels of item feedback. *Item-verified supervision* selects admissible retrieval expressions for initial training; *fusion-aware refinement* evaluates the recommendation list produced by newly generated query sets. We evaluate the latter through the observed next item's rank and regularize the generator toward a reference policy trained on verified queries. Let  $Q_{\phi^{(0)}}$  denote the fixed reference obtained by supervised fine tuning (SFT) on retrieval-verified queries.

For a training pair  $(\mathbf{s}^{(t)}, i_t)$ , the regularized objective is

$$\begin{aligned} \mathcal{L}(\phi \mid \mathbf{s}^{(t)}, i_t) = & \underbrace{\mathbb{E}_{q \sim Q_\phi(\cdot \mid \mathbf{s}^{(t)})} [\ell_R(i_t, q)]}_{\text{retrieval rank loss}} \\ & + \underbrace{\beta D_{\text{KL}}(Q_\phi(\cdot \mid \mathbf{s}^{(t)}) \parallel Q_{\phi^{(0)}}(\cdot \mid \mathbf{s}^{(t)}))}_{\text{reference regularization}}, \end{aligned} \quad (2)$$

Here,  $\ell_R(i_t, q) = \log(1 + \rho_F(i_t \mid q))$ . This rank loss evaluates the target position in the combined recommendation list produced by the query set. It requires retrieval ranks and no probability normalization over items. The KL term penalizes deviation from the SFT reference.

The objective measures recovery of an observed interaction through retrieval, allowing different query phrasings to receive feedback. We learn the generator in two stages: SFT constructs the initialization and reference policy, followed by group-relative policy updates using retrieval ranks. The latter use a policy surrogate for discrete generation and ranking, with a log-probability-based estimate of the regularization term.

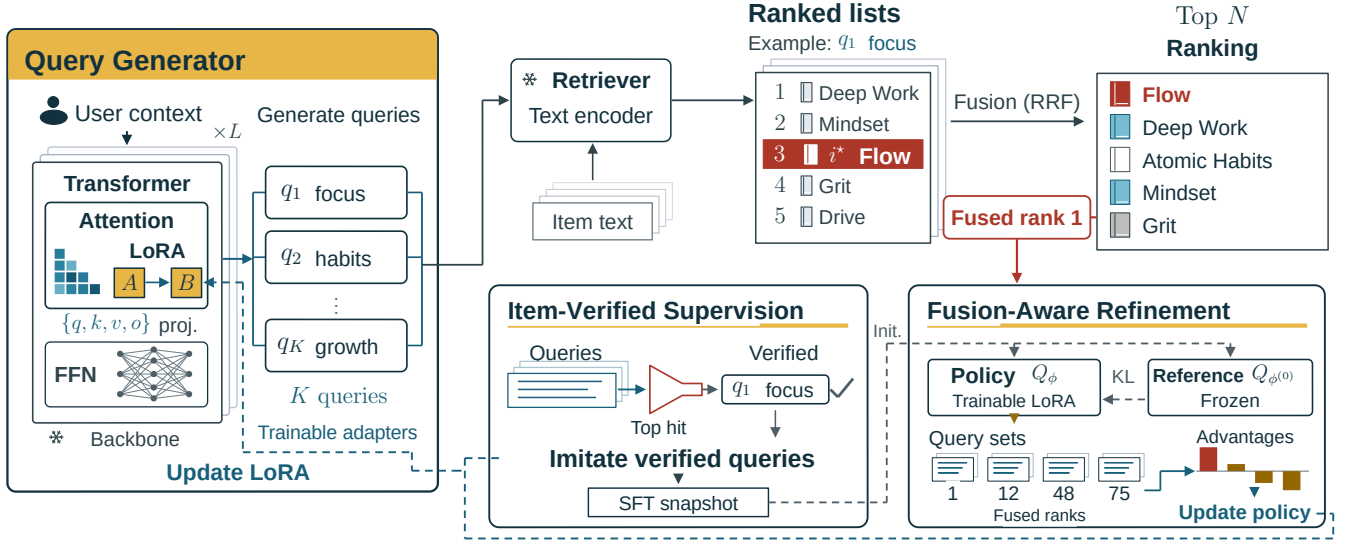


Figure 2: R2QRec overview. Item-verified supervision constructs SFT targets from queries that recover observed items. Fusion-aware refinement applies GRPO to target ranks in the RRF-fused lists. The policy and frozen reference start from the SFT checkpoint; the policy is the query generator shown on the left. KL regularization constrains the policy update. Only LoRA adapters are trained; RRF combines retrieval lists at inference.

### 3.3 Item-Verified Supervision

Interaction logs specify which item was selected but do not provide the query that would retrieve it. We construct supervision by generating candidate queries and verifying them with  $R$ . The observed item provides a criterion for evaluating candidate expressions without fixing their wording. For each training pair  $(s_u^{(t)}, i_t)$ , the base LLM generates a candidate set  $Q_u^{(t)}$  of  $M$  query strings. We retain queries that retrieve  $i_t$  within the acceptance threshold  $\tau$ :

$$\hat{Q}_u^{(t)} = \{q_j \in Q_u^{(t)} \mid \text{rank}_R(i_t \mid q_j) \leq \tau\}. \quad (3)$$

Only pairs with at least one accepted query enter the SFT set. We use the retained queries to construct training responses  $v$ , yielding a dataset  $\hat{\mathcal{D}} = \{(s, v)\}$ . The acceptance threshold  $\tau$  selects initial supervision independently of the recommendation cutoff  $N$ . The target item is used by the retrieval verifier and is not appended to the generator’s input. Consequently, a query can become a training target through its retrieval behavior even when it differs from the observed item’s title. For a training response  $v$  with tokens  $v_\ell$ , we minimize

$$\mathcal{L}_{\text{SFT}}(\phi) = - \sum_{(s, v) \in \hat{\mathcal{D}}} \sum_{\ell=1}^{|v|} \log Q_\phi(v_\ell \mid s, v_{<\ell}). \quad (4)$$

We update LoRA adapters [10] and keep the LLM backbone and retriever frozen. The resulting checkpoint  $Q_{\phi(0)}$  initializes reinforcement learning and serves as its reference policy.

### 3.4 Retrieval Rank Feedback

SFT imitates accepted queries, while reinforcement learning compares the retrieval outcomes of newly generated query sets. For each training context  $s$  and target  $i_t$ , we generate  $G$  candidate query

sets  $q^{(1)}, \dots, q^{(G)}$  from  $Q_\phi(\cdot \mid s)$ . Each set contains  $K$  queries. We execute the queries, fuse their retrieval lists with RRF, and compute the reward from the resulting rank of  $i_t$ :

$$r^{(g)} = -\log(1 + \rho_F(i_t \mid q^{(g)})). \quad (5)$$

The reward evaluates the combined recommendation list through the same fusion operator used to produce recommendations. Log-rank feedback distinguishes fused target ranks beyond the recommendation cutoff, where cutoff-based hit rewards are tied.

On the same rollouts, strictly monotonic rank feedback preserves at least as many groups with nonzero centered advantages as binary hit feedback; Appendix C.4 establishes this comparison.

### 3.5 Fusion-Aware Refinement

The counts  $K$  and  $G$  have distinct roles.  $K$  controls how many retrieval requests a single response contains;  $G$  controls how many alternative responses are compared for one context. With the default  $K = 5$  and  $G = 4$ , we request four responses of five queries and assign one reward to each response.

We use GRPO’s within-group reward normalization [23] to compute the group relative advantage:

$$\hat{A}^{(g)} = \frac{r^{(g)} - \bar{r}}{\sigma_r + \epsilon}, \quad (6)$$

where  $\bar{r}$  and  $\sigma_r$  are the mean and standard deviation of the  $G$  rewards for the same context, and  $\epsilon > 0$  stabilizes the denominator. The group mean provides a context-specific baseline: a positive  $\hat{A}^{(g)}$  identifies a response with an above-average retrieval reward, while a negative value identifies one with a below-average reward.

An earlier Books study recorded approximately 88% zero reward-gradient steps with binary hit feedback, compared with 6.4% with log-rank feedback.

Dividing by the group standard deviation normalizes the scale of these relative comparisons. Every query in a response shares the same set-level advantage. If all rewards in a group coincide, the centered advantages are zero. In particular,  $G = 1$  removes the relative reward signal. The policy loss combines the advantage signal with a KL penalty:

$$\mathcal{L}_{\text{RL}}(\phi) = -\frac{1}{G} \sum_{g=1}^G \hat{A}^{(g)} \frac{\log Q_{\phi}(q^{(g)} | \mathbf{s})}{T_g} + \beta D_{\text{KL}}(Q_{\phi}(\cdot | \mathbf{s}) \| Q_{\phi^{(0)}}(\cdot | \mathbf{s})). \quad (7)$$

The first term is the policy learning term, averaged over the  $G$  generated responses. Here  $\log Q_{\phi}(q^{(g)} | \mathbf{s})$  is the sum of token log probabilities for the generated response, and  $T_g$  is its token count. Their ratio uses the mean token log probability so that the term does not grow simply by summing over a longer response. With advantages held fixed during the update, minimizing the negative advantage-weighted term encourages higher probability for above-average responses and lower probability for below-average ones. The resulting advantage updates the complete response and is determined by its fused target rank.

The second term regularizes the current policy toward the frozen SFT policy  $Q_{\phi^{(0)}}$ . The direction  $D_{\text{KL}}(Q_{\phi} \| Q_{\phi^{(0)}})$  penalizes assigning probability to outputs that the reference considers unlikely. The coefficient  $\beta$  controls how strongly this reference constrains rank-driven updates: a larger value favors staying closer to the retrieval-verified initialization. We compute the regularization term from response log probabilities under the current policy and the frozen reference. We use one gradient update per rollout group with gradient norm clipping, without reusing rollouts across multiple optimization epochs.

### 3.6 From Item Feedback to Recommendation Utility

The design separates three questions: what an observed item can supervise, how query combinations affect item competition, and which recommendation depths the loss emphasizes. An item label specifies a retrieval target without prescribing a unique expression. Item-verified supervision uses it to define admissible queries; Appendix C.1 formalizes this selection by conditioning on acceptance.

Admissibility alone does not characterize the outcome of combining queries. Lemma C.1 gives two sets of retrieval lists with identical target-rank vectors but different fused target ranks. The difference arises from competing items across the lists: even the full vector of individual target ranks, and therefore its best or mean, cannot determine the fused outcome in general. Fusion-aware refinement observes this combined ranking when comparing generated sets.

Having specified the feedback level, the log-rank objective determines its weighting across recommendation depths.

**PROPOSITION 3.1 (WEIGHTED CUTOFF-RISK DECOMPOSITION).** *Fix an event with  $L \geq 1$  eligible items and one target. Let  $Z \in \{1, \dots, L\}$  be its fused rank under any distribution over generated query sets, and let  $h_n = \Pr(Z \leq n)$ . Then*

$$\mathbb{E}[\log(1 + Z)] = \log 2 + \sum_{n=1}^{L-1} w_n (1 - h_n), \quad w_n = \log \frac{n+2}{n+1}. \quad (8)$$

**PROOF SKETCH.** Expand  $\log(1 + Z) - \log 2$  into adjacent log increments and take expectations. Appendix C.3 gives the full derivation.  $\square$

With one target per event,  $h_n$  is the expected Recall@ $n$ . Thus, the retrieval loss is a depth-weighted recommendation risk: it weights missed-target events across cutoffs, with positive weights that decrease toward the tail. An equal one-position improvement receives more reward near the head, while improvements beyond a particular cutoff still provide feedback. The identity explains the objective's weighting across depths; it does not imply improvement at every individual cutoff.

## 4 Empirical Study

Our experiments address the following research questions:

- **RQ1:** How does R2QRec compare with existing recommenders on public datasets?
- **RQ2:** How do its training stages and components contribute to recommendation quality?
- **RQ3:** How sensitive is R2QRec to model choices, hyperparameters, and domain changes?
- **RQ4:** How does query retrieval compare with agentic candidate ranking?

### 4.1 Setup

We evaluate on three public datasets Yelp, MovieTV, and Amazon Books [31]. We use the benchmark's evaluation splits and train a separate generator for each domain. Table 1 summarizes the data.

**Table 1: Dataset statistics. Avg. length is the mean interaction-history length.**

| Dataset | Users | Items  | Interactions | Avg. length |
|---------|-------|--------|--------------|-------------|
| Yelp    | 3.0K  | 31.6K  | 63.1K        | 21.4        |
| MovieTV | 5.6K  | 29.0K  | 79.7K        | 14.1        |
| Books   | 7.4K  | 120.9K | 207.8K       | 28.2        |

The main retrieval evaluation ranks the full item set without negative sampling and includes all 2,950, 5,649, and 7,377 held out users on Yelp, MovieTV, and Books, respectively. Unless stated otherwise, experiments use these evaluation sets; the group-size, KL-weight, and inference query-count scans use the same fixed subset of 1,000 users per domain. History items are excluded from retrieval. We select the SFT acceptance threshold  $\tau$ ,  $G$ ,  $\beta$ , and inference query count  $K$  on validation data; the remaining R2QRec hyperparameters use fixed defaults. No configuration is selected or adjusted using test results. All text retrieval methods index the same documents, formed from item titles and cleaned descriptions. For Yelp, descriptions include categories, city, and star rating. For MovieTV, they include the synopsis after removing physical media and condition notes.

We use Qwen2.5-7B-Instruct [19] as the autoregressive query generator. History items enter the prompt as titles, with the most recent last. Descriptions and metadata are available to the retriever but are excluded from the generator's history input. A shared

**Table 2: Main recommendation results (%). Bold/underline: best/second best.**

| Method         | Yelp        |             |             |             |             |             |             |             |      |             | MovieTV     |              |             |             |             |             |             |             |             |              | Books       |      |      |             |  |        |  |  |  |  |
|----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|------|------|-------------|--|--------|--|--|--|--|
|                | Recall@N    |             |             |             |             | NDCG@N      |             |             |      |             | Recall@N    |              |             |             |             | NDCG@N      |             |             |             |              | Recall@N    |      |      |             |  | NDCG@N |  |  |  |  |
|                | @5          | @10         | @20         | @50         | @5          | @10         | @20         | @50         | @5   | @10         | @20         | @50          | @5          | @10         | @20         | @50         | @5          | @10         | @20         | @50          | @5          | @10  | @20  | @50         |  |        |  |  |  |  |
| SASRec         | 0.17        | 0.27        | 0.68        | 1.46        | 0.10        | 0.13        | 0.23        | 0.39        | 4.87 | 6.76        | 9.17        | 13.47        | 3.70        | 4.30        | 4.91        | <u>5.76</u> | 1.74        | 1.91        | 2.10        | 2.60         | 1.52        | 1.57 | 1.62 | 1.72        |  |        |  |  |  |  |
| BERT4Rec       | 0.44        | 0.64        | 1.08        | 2.14        | 0.31        | 0.37        | 0.48        | 0.69        | 4.71 | 6.57        | <u>9.22</u> | <b>13.95</b> | 3.43        | 4.03        | 4.70        | 5.63        | 1.34        | 1.60        | 1.88        | 2.55         | 1.15        | 1.23 | 1.30 | 1.44        |  |        |  |  |  |  |
| LightGCN       | 0.61        | 1.29        | 2.20        | 4.17        | 0.33        | 0.55        | 0.78        | 1.16        | 2.69 | 4.32        | 6.36        | 10.78        | 1.84        | 2.35        | 2.87        | 3.74        | 1.87        | 2.48        | 3.10        | 4.15         | 1.40        | 1.60 | 1.75 | 1.96        |  |        |  |  |  |  |
| UniSRec        | 0.98        | 1.46        | 2.17        | 5.08        | 0.58        | 0.73        | 0.91        | 1.47        | 4.73 | 6.37        | 8.66        | 13.05        | 3.46        | 4.00        | 4.57        | 5.43        | 2.72        | 3.52        | 4.36        | 5.95         | 2.15        | 2.42 | 2.63 | 2.94        |  |        |  |  |  |  |
| TIGER          | 0.51        | 0.78        | 1.25        | 1.25        | 0.35        | 0.44        | 0.56        | 0.56        | 2.76 | 3.88        | 5.51        | 5.51         | 2.05        | 2.41        | 2.82        | 2.82        | 0.84        | 0.87        | 0.95        | 0.95         | 0.77        | 0.77 | 0.79 | 0.79        |  |        |  |  |  |  |
| LETTER         | 0.27        | 0.68        | 1.08        | 1.08        | 0.19        | 0.32        | 0.42        | 0.42        | 2.83 | 3.88        | 5.63        | 5.63         | 2.21        | 2.54        | 2.99        | 2.99        | 0.87        | 0.95        | 1.07        | 1.07         | 0.76        | 0.79 | 0.82 | 0.82        |  |        |  |  |  |  |
| BM25           | 0.85        | 1.25        | 1.66        | 2.64        | 0.56        | 0.68        | 0.79        | 0.98        | 4.39 | 6.18        | 8.13        | 10.83        | 3.11        | 3.69        | 4.18        | 4.73        | 3.82        | 4.87        | 5.99        | 7.75         | 2.80        | 3.14 | 3.43 | 3.78        |  |        |  |  |  |  |
| InputRetrieval | 0.54        | 0.81        | 1.32        | 3.39        | 0.43        | 0.51        | 0.63        | 1.04        | 2.21 | 3.38        | 4.69        | 7.70         | 1.44        | 1.81        | 2.14        | 2.74        | 2.97        | 4.27        | 6.14        | 10.63        | 1.98        | 2.40 | 2.87 | 3.75        |  |        |  |  |  |  |
| BIGRec         | 1.59        | 2.17        | 3.22        | 5.15        | 1.20        | 1.38        | 1.65        | 2.03        | 5.81 | <u>6.82</u> | 7.61        | 9.10         | <u>4.58</u> | <u>4.90</u> | <u>5.10</u> | <u>5.40</u> | <u>6.40</u> | 7.62        | 8.76        | 10.34        | <u>4.99</u> | 5.39 | 5.67 | 5.99        |  |        |  |  |  |  |
| GPT4Rec        | <u>2.10</u> | <u>2.92</u> | <u>3.93</u> | <u>6.58</u> | <u>1.37</u> | <u>1.63</u> | <u>1.88</u> | <u>2.41</u> | 5.08 | 6.43        | 7.79        | 10.02        | 3.57        | 4.00        | 4.35        | 4.78        | 6.14        | 7.39        | 8.74        | 10.74        | 4.76        | 5.16 | 5.50 | 5.89        |  |        |  |  |  |  |
| QueRec         | 1.36        | 2.58        | 3.86        | 5.97        | 0.91        | 1.30        | 1.62        | 2.04        | 3.98 | 4.94        | 6.21        | 8.20         | 3.09        | 3.40        | 3.72        | 4.12        | 3.32        | 4.45        | 6.03        | 7.94         | 2.37        | 2.74 | 3.13 | 3.51        |  |        |  |  |  |  |
| QueryAgent-R1  | 1.08        | 1.90        | 3.05        | 4.88        | 0.70        | 0.95        | 1.25        | 1.61        | 2.94 | 3.81        | 5.01        | 6.71         | 2.10        | 2.38        | 2.68        | 3.02        | 6.07        | <u>7.66</u> | <u>9.37</u> | <u>11.71</u> | 4.63        | 5.15 | 5.58 | <u>6.05</u> |  |        |  |  |  |  |
| R2QRec         | 3.22        | 4.75        | 6.34        | 10.24       | 2.26        | 2.75        | 3.15        | 3.91        | 6.50 | 8.00        | 10.25       | <u>13.67</u> | 4.59        | 5.07        | 5.64        | 6.32        | 9.00        | 11.32       | 13.56       | 16.38        | 6.91        | 7.66 | 8.23 | 8.79        |  |        |  |  |  |  |

prompt template fills domain specific slots for item type, vocabulary hints, and query examples. The prompt templates remain fixed within each comparison. Dense retrieval methods use the frozen all-MiniLM-L6-v2 encoder [21]. We use dot products of normalized query and item embeddings.

For open-pool recommendation, we report Recall@N and NDCG@N for  $N \in \{5, 10, 20, 50\}$ , with NDCG@10 as the primary metric. Each user has one relevant target, so the ideal DCG is 1. Unless stated otherwise, R2QRec generates  $K = 5$  queries.

## 4.2 Implementation Details

We implement R2QRec in PyTorch and train on a single NVIDIA A800 GPU with 80 GB memory. We search the SFT acceptance threshold  $\tau$  from 50 to 1,000, using the same training-only candidate pool and SFT settings for each threshold. Each domain's threshold is selected by validation NDCG@10 after SFT and fixed before policy refinement.

Both training stages use LoRA [10] on the attention query, key, value, and output projections ( $W_Q, W_K, W_V, W_O$ ). We set the adapter rank to  $r = 16$  and the scaling parameter to  $\alpha = 32$ . The backbone and retrieval encoder remain frozen. SFT runs for three epochs with learning rate  $2 \times 10^{-4}$ , per-device batch size 2, gradient accumulation over eight batches, and a 3% warmup ratio. The SFT checkpoint remains fixed as the KL reference.

We search  $G$  from 1 to 32,  $\beta$  from 0 to 1, and inference query count  $K$  from 1 to 20. For all baseline models, we carefully tune hyperparameters on the validation sets and report test performance using the best validation-selected configuration. Results are averaged over five random runs. Reported improvements over the best-performing baselines are statistically significant ( $p < 0.05$ ), assessed using a two-sided  $t$ -test.

## 4.3 Baselines

We compare two groups of baselines. **Discriminative and generative recommendation.** SASRec [13] and BERT4Rec [25] model interaction sequences; LightGCN [7] learns from the interaction

graph; UniSRec [9] incorporates item text for transferable sequential recommendation. TIGER [20] and LETTER [28] generate item identifiers. **Query-based recommendation.** BM25 [22] and InputRetrieval perform lexical and dense retrieval, respectively, from the same user context and item documents as R2QRec, without an LLM. BIGRec [1] generates target-item text and grounds it to the catalog. GPT4Rec [16] trains a query generator using item text; the concurrent QueryAgent-R1 [24] uses an agentic search procedure to refine e-commerce query recommendations. QueRec [6] combines personalized-query retrieval with collaborative scores through adaptive reranking, without LLM fine tuning. Trainable text-generation baselines use their respective training objectives. GPT4Rec and BIGRec use the same Qwen2.5-7B backbone, frozen MiniLM retriever,  $K = 5$  query budget, and rank-fusion procedure as R2QRec. The zero-shot generator uses the same backbone, prompt, and retrieval pipeline as R2QRec, without task training.

## 4.4 Results on Public Datasets (RQ1)

Table 2 reports recommendation results over the full item set. We make four observations.

- **Leading results across domains.** R2QRec achieves the best results among the evaluated methods in 23 of the 24 dataset-metric comparisons. Its relative NDCG@10 gains over the strongest baseline are 68.7%, 3.5%, and 42.1% on Yelp, MovieTV, and Books, respectively.
- **Strong performance on short recommendation lists.** On MovieTV, R2QRec improves Recall@5 and Recall@10 by 11.9% and 17.3% over the strongest baseline. BERT4Rec leads at Recall@50 (13.95% versus 13.67%), where R2QRec ranks second, while R2QRec leads at every NDCG cutoff. This pattern highlights its effectiveness at placing relevant items near the top of the recommendation list.
- **Effectiveness of generated retrieval queries.** Interestingly, LLM-based query recommenders provide the strongest baseline in 21 of the 24 comparisons. GPT4Rec leads the baselines on Yelp, while BIGRec and QueryAgent-R1 provide the strongest results

on Books. These results support generated queries as an effective interface between user context and item retrieval.

- **Gains under a shared inference pipeline.** With the same backbone, retriever, query budget, and rank-fusion procedure as GPT4Rec and BIGRec, R2QRec outperforms both on every reported metric across all three domains. The advantage therefore persists when these inference components are held fixed. The following ablations examine the contributions of retrieval verification and policy refinement.

#### 4.5 Ablation Study (RQ2)

For the component ablations, we report mean performance over five random runs and assess statistical significance at the  $p < 0.05$  level. Table 3 examines individual components on MovieTV using the same generator backbone and retriever. Full achieves statistically significant NDCG@10 gains over the component ablations. SFT only achieves 4.33% NDCG@10, compared with 5.07% for Full. Policy refinement adds 0.74 percentage points over SFT only.

- **Training and retrieval verification.** Removing retrieval filtering from SFT data construction, while retaining subsequent GRPO, reduces MovieTV NDCG@10 from 5.07% to 4.03%, a drop of 1.04 percentage points. This controlled comparison supports filtering the generated supervision before training.
- **Group feedback and KL.** On MovieTV,  $G = 1$  yields 4.37% NDCG@10, compared with 5.07% for Full. At  $G = 1$ , Eq. 6 gives zero reward advantage, leaving only regularization. The w/o KL variant reaches 4.67% NDCG@10, compared with 5.07% for Full.
- **Multiple retrieval queries.** Using  $K = 1$  at inference with the Full checkpoint lowers NDCG@10 from 5.07% to 4.75% and Recall@10 from 8.00% to 7.45%. The checkpoint is held fixed, so this comparison measures the effect of the inference query count.

Table 5 traces both supervision sources through training with the same backbone and inference pipeline. Title SFT improves from 3.14% to 3.71% after GRPO, while retrieval-verified SFT improves from 4.33% to 5.07%. Policy refinement benefits both initializations, and retrieval-verified supervision provides the stronger starting point and final result. The 1.36-percentage-point final gap shows the value of the supervision source even when both pipelines include policy refinement.

Additional verification results appear in Table 4.

**Table 3: MovieTV component ablations (%). Bold: best value per metric. The w/o multi-query variant uses  $K = 1$  at inference with the Full checkpoint.**

| Method               | Recall@N    |             |              |              | NDCG@N      |             |             |             |
|----------------------|-------------|-------------|--------------|--------------|-------------|-------------|-------------|-------------|
|                      | @5          | @10         | @20          | @50          | @5          | @10         | @20         | @50         |
| Zero-shot            | 4.07        | 5.45        | 7.13         | 9.42         | 2.93        | 3.37        | 3.79        | 4.24        |
| SFT only             | 5.08        | 7.06        | 8.92         | 11.44        | 3.69        | 4.33        | 4.80        | 5.30        |
| <b>Full</b>          | <b>6.50</b> | <b>8.00</b> | <b>10.25</b> | <b>13.67</b> | <b>4.59</b> | <b>5.07</b> | <b>5.64</b> | <b>6.32</b> |
| w/o verification     | 4.76        | 6.36        | 7.98         | 11.13        | 3.52        | 4.03        | 4.44        | 5.07        |
| w/o KL ( $\beta=0$ ) | 5.58        | 7.43        | 9.10         | 12.13        | 4.06        | 4.67        | 5.09        | 5.69        |
| $G=1$                | 5.26        | 7.10        | 8.87         | 11.52        | 3.77        | 4.37        | 4.82        | 5.34        |
| w/o multi-query      | 5.93        | 7.45        | 9.17         | 11.63        | 4.26        | 4.75        | 5.19        | 5.68        |

**Table 4: Retrieval verification across domains (%). The w/o verification variant retains all generated pseudo-labels; subsequent training stages are unchanged.**

| Domain  | Method           | Recall@N |       |       |       | NDCG@N |      |      |      |
|---------|------------------|----------|-------|-------|-------|--------|------|------|------|
|         |                  | @5       | @10   | @20   | @50   | @5     | @10  | @20  | @50  |
| Yelp    | Full             | 3.22     | 4.75  | 6.34  | 10.24 | 2.26   | 2.75 | 3.15 | 3.91 |
|         | w/o verification | 2.88     | 4.03  | 6.14  | 9.76  | 1.86   | 2.22 | 2.75 | 3.47 |
| MovieTV | Full             | 6.50     | 8.00  | 10.25 | 13.67 | 4.59   | 5.07 | 5.64 | 6.32 |
|         | w/o verification | 4.76     | 6.36  | 7.98  | 11.13 | 3.52   | 4.03 | 4.44 | 5.07 |
| Books   | Full             | 9.00     | 11.32 | 13.56 | 16.38 | 6.91   | 7.66 | 8.23 | 8.79 |
|         | w/o verification | 8.70     | 10.90 | 13.24 | 16.08 | 6.64   | 7.35 | 7.95 | 8.51 |

**Table 5: Supervision source on MovieTV: NDCG@10 (%).  $\Delta$  is the gain from subsequent GRPO over the corresponding SFT initialization.**

| Supervision                | SFT only | + GRPO      | $\Delta$ |
|----------------------------|----------|-------------|----------|
| Item titles                | 3.14     | 3.71        | +0.57    |
| Retrieval-verified queries | 4.33     | <b>5.07</b> | +0.74    |

#### 4.6 Hyperparameter Sensitivity (RQ3)

Figure 3 examines group size  $G$ , KL weight  $\beta$ , and query count  $K$  on a fixed 1,000-user subset of each domain, using NDCG@10 and Recall@10. These scans describe sensitivity; configuration selection uses the validation protocol in Section 4.1. Group size has no common optimum: MovieTV attains its highest NDCG@10 at  $G = 32$ , whereas Books peaks at  $G = 8$ . Performance varies non-monotonically across the displayed KL weights. Query-count effects differ across domains: MovieTV increases from 4.05% at  $K = 1$  to 5.11% at  $K = 20$ , with intermediate fluctuations, while Books NDCG@10 varies within 6.75–7.01% without a consistent gain as  $K$  grows.

#### 4.7 Generator and Retriever Analysis (RQ3)

**4.7.1 Query generator families.** Figure 4 compares zero-shot and Full results for three query generators across the three domains. On Yelp, Llama-3-8B [18] achieves 3.01% NDCG@10 and outperforms Qwen (2.75%) and Mistral-7B [11] (2.51%) on this metric. Qwen achieves the highest NDCG@10 on MovieTV and Books, at 5.07% and 7.66%. Llama reaches 3.63% and 6.83%, while Mistral reaches 2.68% and 6.75% on these domains. The pipeline is applicable to all three backbones, but the best generator depends on the domain.

**4.7.2 Generator size and retrieval encoder.** Figure 5 separately examines these choices on Yelp and MovieTV *without task training*. On MovieTV, increasing Qwen from 1.5B to 7B improves NDCG@10 from 2.04% to 3.37%, while 14B does not improve further under this fixed prompt. On MovieTV, bge reaches 3.53%, compared with 3.37% for MiniLM and 3.24% for mpnet. MiniLM is close to bge on Yelp. We retain MiniLM as the common retriever for the main comparisons.

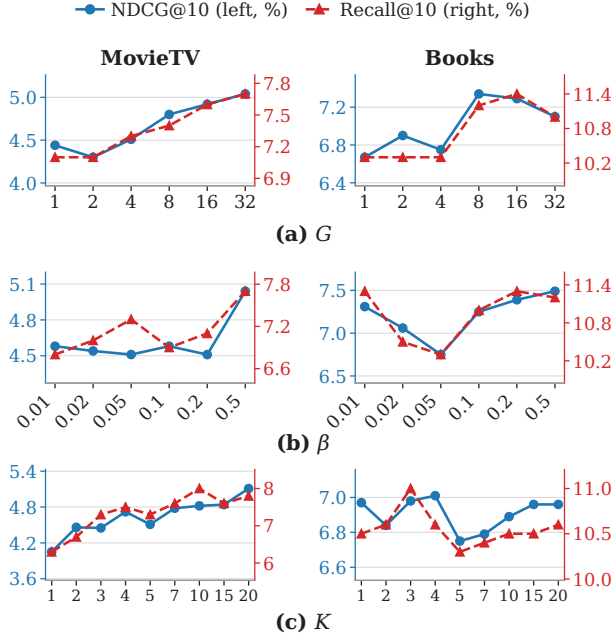


Figure 3: Hyperparameter sensitivity on 1,000 sampled users per domain. Blue/left: NDCG@10; red/right: Recall@10. (a)  $G$ . (b)  $\beta$ . (c)  $K$ .

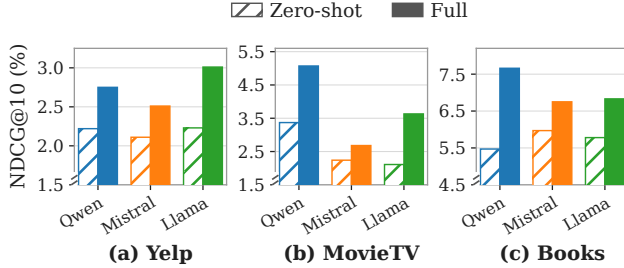


Figure 4: Query generator families: adjacent hatched/solid bars compare zero-shot/Full for Qwen2.5-7B, Mistral-7B, and Llama-3-8B.

#### 4.8 Cross-Domain Transfer (RQ3)

We train the generator on one domain and evaluate it on another without updating its parameters, retrieving from the target domain’s item index. Figure 6 organizes the source–target comparisons. Transfer is asymmetric. In-domain training remains strongest on Yelp and MovieTV, at 2.75% and 5.07%. On Books, generators trained on Yelp and MovieTV achieve NDCG@10 of 8.01% and 8.09%, respectively, exceeding the Books-trained generator (7.66%). These results indicate transferable query-generation behavior, with performance that depends on the source–target pair.

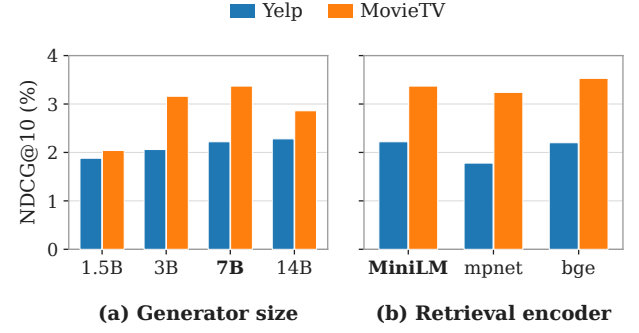


Figure 5: Generator and encoder sensitivity without task training. (a) Generator size, with MiniLM. (b) Encoder, with the 7B generator. Bold labels mark defaults.

|         | Target domain |         |       | NDCG@10 (%) |
|---------|---------------|---------|-------|-------------|
|         | Yelp          | MovieTV | Books |             |
| Yelp    | 2.75          | 4.54    | 8.01  | 8           |
| MovieTV | 2.41          | 5.07    | 8.09  |             |
| Books   | 2.43          | 3.93    | 7.66  |             |

Figure 6: Cross-domain transfer: NDCG@10 (%). Rows identify training domains and columns target domains.

#### 4.9 Case Study: From History to Queries

Figure 7 illustrates how a reading history can be expressed as several retrieval queries. The books suggest related interests in cognitive science, behavioral change, and human achievement. The example queries express these interests from different angles, with *Flow: The Psychology of Optimal Experience* illustrating a relevant item. The intermediate language representation makes the connection between historical interests and retrieval requests explicit.

#### 4.10 Agentic Recommendation (RQ4)

We compare R2QRec with  $i^2$ Agent and MemRec [2] in a candidate-ranking setting. All methods use Qwen2.5-7B and the same candidate set for each user: the target and nine randomly sampled negatives, excluding history items.

The agentic baselines reason directly over candidate content. R2QRec generates queries from user context, then scores the candidates by retrieval; candidate items are not included in its generator prompt. We summarize this setting with NDCG@5.

Figure 8 shows that R2QRec reaches NDCG@5 of 60.5%, 53.9%, and 60.8% on Yelp, MovieTV, and Books.  $i^2$ Agent achieves 52.6%, 56.5%, and 54.3%, respectively, while MemRec achieves 67.6%, 67.7%, and 62.9%. MemRec leads on all three datasets. R2QRec outperforms  $i^2$ Agent on Yelp and Books, while scoring lower on MovieTV. These results distinguish open-catalog query retrieval from reasoning directly over a supplied candidate list.

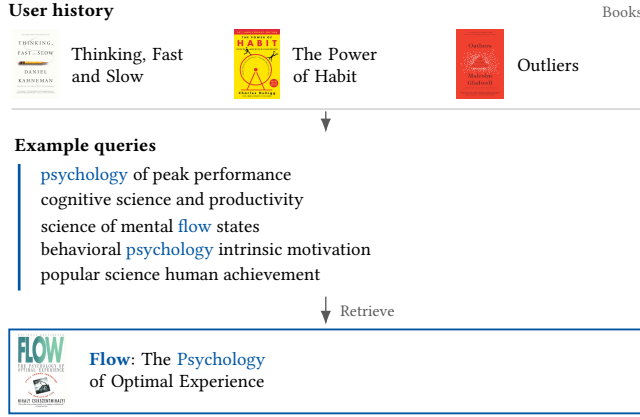


**Figure 7: Books case study. Five queries express related interests in the history, with *Flow* as an example relevant item. Cover thumbnails identify book titles.**

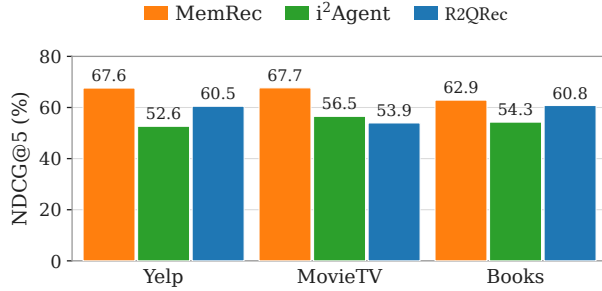


**Figure 8: Candidate-ranking NDCG@5 with the same backbone and 10 candidates per user.**

## 5 Related Work

### 5.1 Query Learning for Recommendation

Query-based recommendation connects language generation with item retrieval, with the closest approaches differing in how they train or construct the generated text. GPT4Rec fine-tunes a language model to predict target item titles, then uses the generated text with BM25 to retrieve recommendations [16]. Its generator learns target-title likelihood; retrieval ranks do not supervise its parameter updates. BIGRec similarly uses target item text for supervised adaptation before grounding generated text to actual items [1].

Rather than adapting the generator to target text, QueRec [6] generates enriched item descriptions and personalized queries through prompting without LLM fine tuning, and integrates semantic retrieval with collaborative signals. This approach uses an LLM to express preferences, but does not learn its query generation policy from the retrieval outcomes. Rec-R1 couples LLM-generated text with fixed recommendation models [17]. Concurrent QueryAgent-R1 combines memory-based user profiling with a search agent for proactive e-commerce query recommendation [24]. The focus of R2QRec is how item-level observations supervise an intermediate

query interface. Retrieval verification converts those observations into query targets without prescribing their wording, while policy refinement evaluates generated sets at the fused ranking used for recommendation. This connects supervision construction and recommendation-level outcome optimization through a fixed retrieval system.

### 5.2 Other Recommendation Approaches

Discriminative recommenders score items through learned representations, with sequential models deriving user representations from interaction histories. GRU4Rec uses recurrent networks [8], SASRec applies causal self attention [13], and BERT4Rec uses masked item prediction [25]. LightGCN captures collaborative structure through graph propagation [7], while UniSRec uses item text to support transfer across domains [9]. RecFormer extends this approach to textual item sequences [15]. Generative recommenders instead predict item identifiers or text. P5 casts recommendation tasks as text generation [5]; TIGER generates semantic IDs obtained through residual quantization [20]; and OneRec develops generative recommendation at industrial scale [4]. LETTER learns item tokenization with semantic and collaborative regularization [28]. R2QRec generates retrieval requests that can each match multiple items.

Tags provide another semantic connection between users and items [29, 30, 32]. SemRec combines structural and semantic relationships in social tagging data for item recommendation [30]. R2QRec instead learns to compose language requests, with observed-item retrieval outcomes supervising their wording.

Natural-language interfaces broaden how users interact with recommenders [34]. Agentic approaches further introduce memory, tool use, and interaction between agents [2, 12, 27, 31, 33]. The candidate-ranking protocols considered here provide a small item set to the LLM, allowing direct reasoning about candidate content. R2QRec instead generates retrieval queries without candidate enumeration in its prompt, and ranks the supplied candidates by retrieval when evaluated in this setting.

## 6 Conclusion

We presented R2QRec, a framework for learning recommendation queries from item interactions through a fixed retrieval system. Its design separates two learning decisions: retrieval verification identifies useful expressions to imitate, and policy refinement evaluates newly generated query sets through their combined item ranking. The same retrieval interface supports both stages, while only the query generator is updated. Controlled comparisons show the benefit of retrieval-verified supervision over title imitation and the further gain from policy refinement. The resulting query interface supports recommendation over the full item set without requiring query annotations or a learned reward model.

Future work can adapt the query budget to user context and examine alternative fusion functions within query learning. Another direction is to study query refresh policies for changing user interests and item collections. The separation between query generation and retrieval also supports exploring additional retrieval backends. Broader questions of verifier reliability when LLMs arbitrate between prior knowledge and retrieved evidence are also relevant to retrieval-augmented systems more generally [26].

## References

- [1] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems* (2025).
- [2] Weixin Chen, Yuhao Zhao, Jingyuan Huang, Zihe Ye, Mingxuan Ju, Tong Zhao, Neil Shah, Li Chen, and Yongfeng Zhang. 2026. MemRec: Collaborative Memory-Augmented Agentic Recommender System. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- [3] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [4] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965* (2025).
- [5] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*.
- [6] Donghee Han, Hwanjun Song, and Mun Yong Yi. 2025. Rethinking LLM-Based Recommendations: A Personalized Query-Driven Parallel Integration.. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.
- [7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-Based Recommendations with Recurrent Neural Networks. In *International Conference on Learning Representations*.
- [9] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- [11] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).
- [12] Li Kang, Yuhao Zhao, and Li Chen. 2026. Personalized News Headline Generation via Multi-Agent User Simulation. In *Findings of the Association for Computational Linguistics: EMNLP 2026*.
- [13] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*.
- [14] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
- [15] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [16] Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation. In *Proceedings of the 2023 SIGIR Workshop on eCommerce (CEUR Workshop Proceedings)*.
- [17] Jiacheng Lin, Tian Wang, and Kun Qian. 2025. Rec-R1: Bridging Generative Large Language Models and User-Centric Recommendation Systems via Reinforcement Learning. *Transactions on Machine Learning Research* (2025).
- [18] Llama Team. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).
- [19] Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).
- [20] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, Maciej Kula, Ed Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. In *Advances in Neural Information Processing Systems*.
- [21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
- [22] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* (2009).
- [23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).
- [24] Dike Sun, Zheng Zou, Jingtong Zang, Qi Sun, Huaipeng Zhao, Tao Luo, and Xiaoyi Zeng. 2026. QueryAgent-R1: Bridging Query Generation and Product Retrieval for E-Commerce Query Recommendation. *arXiv preprint arXiv:2606.05671* (2026).
- [25] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
- [26] Yuxi Sun, Wenbo Shang, Wei Gao, Xin Huang, and Jing Ma. 2026. Diagnosing LLM Arbitration Behavior over Pre-evidence Epistemic States in RAG-based Fact-Checking. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*.
- [27] Zhen Tao, Riwei Lai, Chenyun Yu, Weixin Chen, Li Chen, Beibei Kong, Lei Cheng, Chengxiang Zhuo, Zang Li, and Qingqiang Sun. 2026. Personalizing How Recommendation Agents Reason: Self-Evolving Per-User Skills for Recommendation Agents. In *Findings of the Association for Computational Linguistics: EMNLP 2026*.
- [28] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable Item Tokenization for Generative Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*.
- [29] Jinke Wu, Yinghao Wu, Shuchang Liu, Yandong Bai, Qi Zhu, Huihui Xiao, and Fan Yang. 2026. GUIDER: Generative User Interest Discovery via Explicit Reasoning with Large Language Models. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*.
- [30] Guandong Xu, Yanhui Gu, Peter Dolog, Yanchun Zhang, and Masaru Kitsuregawa. 2011. SemRec: A Semantic Enhancement Framework for Tag Based Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [31] Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025. iAgent: LLM Agent as a Shield between User and Recommender Systems. In *Findings of the Association for Computational Linguistics: ACL 2025*.
- [32] Qing Yu, Xiaobei Wang, Shuchang Liu, Yandong Bai, Xiaoyu Yang, Xueliang Wang, Chang Meng, Shanshan Wu, Hailan Yang, Bin Wen, Huihui Xiao, Xiang Li, Fan Yang, Xiaoqiang Feng, Lantao Hu, Han Li, Kun Gai, and Lixin Zou. 2025. Who You Are Matters: Bridging Interests and Social Roles via LLM-Enhanced Logic Recommendation. In *Advances in Neural Information Processing Systems*.
- [33] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024. AgentCF: Collaborative Learning with Autonomous Language Agents for Recommender Systems. In *Proceedings of the ACM Web Conference 2024*.
- [34] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2025. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *ACM Transactions on Information Systems* (2025).

## A Inference Latency and Cost

### A.1 Generation Cost

We report generation time separately from retrieval and fusion. The recorded generation wall times below were measured on a single A800-80GB, with  $K = 5$  and batch size 1.

| Setting              | #users | Wall time | s/user |
|----------------------|--------|-----------|--------|
| Qwen2.5-7B, MovieTV  | 5,649  | 3,963 s   | 0.702  |
| Qwen2.5-7B, Books    | 7,377  | 5,606 s   | 0.760  |
| Qwen2.5-14B, MovieTV | 5,649  | 3,677 s   | 0.651  |
| Qwen2.5-14B, Books   | 7,377  | 6,283 s   | 0.852  |

The generator does not enumerate the item set in its prompt. Its runtime depends on the model, input and output lengths, and generation settings.

### A.2 Retrieval and Indexing Cost

We measure retrieval and fusion on CPU with NumPy against precomputed item embeddings ( $d = 384$ ), as summarized below.

| Domain  | Catalog size | Retrieval + fusion (ms/user) |
|---------|--------------|------------------------------|
| Yelp    | 32K          | 5.0                          |
| MovieTV | 29K          | 5.0                          |
| Books   | 121K         | 26.6                         |

These measurements use exact retrieval. GPU execution and approximate indexing offer additional implementation choices as the item set grows.

Exact dense retrieval scores all eligible items, so its work grows with the item set. Item embeddings are precomputed and reused across requests. A new domain requires constructing its item index, but our transfer experiments do not update the generator parameters. We distinguish generation, retrieval, and offline indexing costs.

## B Pseudocode

Algorithms 1 and 2 summarize training and inference, respectively. Training generates query sets as responses; inference constructs a set from individually generated queries.

---

### Algorithm 1 R2QRec training (one domain)

---

**Require:** training pairs  $\mathcal{D} = \{(s, i_t)\}$ , frozen retriever  $R$ , base LLM, validation-selected acceptance threshold  $\tau$

#### Stage 1: SFT with retrieval verification

- 1: Initialize  $Q_\phi$  from the base LLM
- 2: Generate  $M = 5$  candidate queries for each training context
- 3: Retain each query with  $\text{rank}_R(i_t | q_j) \leq \tau$  ▷ Eq. 3
- 4: Train LoRA adapters on accepted query targets for 3 epochs

#### Stage 2: GRPO

- 5:  $Q_{\phi^{(0)}} \leftarrow Q_\phi$  ▷ freeze SFT reference
  - 6: **for** each policy optimization step **do**
  - 7:   Select a training pair  $(s, i_t)$
  - 8:   Generate  $G$  candidate query sets from  $Q_\phi(\cdot | s)$
  - 9:   **for**  $g = 1$  to  $G$  **do**
  - 10:     Retrieve with each query in  $q^{(g)}$
  - 11:     Fuse the retrieval lists by RRF, as at inference
  - 12:     Compute  $\rho_F(i_t | q^{(g)})$  and  $r^{(g)}$  using Eq. 5
  - 13:   **end for**
  - 14:   Compute group advantages using Eq. 6
  - 15:   Update  $\phi$  once using Eq. 7
  - 16: **end for**
- 

---

### Algorithm 2 R2QRec inference (one user)

---

**Require:** user context  $s_u$ , frozen retriever  $R$ , item index, trained  $Q_\phi$

- 1: Infer  $K$  queries with the trained generator  $Q_\phi$
  - 2: Form the query set  $q = (q_1, \dots, q_K)$
  - 3: Retrieve a ranked list with each  $q_j$ , masking history items
  - 4: Fuse the  $K$  lists by RRF
  - 5: **return** the  $N$  highest ranked items
- 

## C Analysis of the Retrieval Objective

We detail the analysis in Section 3.6 for a fixed training event and frozen retrieval system.

### C.1 Item-Supervised Query Verification

Let  $P_0(x | s)$  be the base proposal distribution over individual query strings, and define

$$\mathcal{A}_{s, i_t} = \{x : \text{rank}_R(i_t | x) \leq \tau\}. \quad (9)$$

Write  $a = P_0(\mathcal{A}_{s, i_t} | s) > 0$ . Ideal rejection sampling yields the conditional distribution

$$P_{\text{acc}}(x | s, i_t) = \frac{P_0(x | s) \mathbf{1}[x \in \mathcal{A}_{s, i_t}]}{a}. \quad (10)$$

For any distribution  $P$  supported on accepted queries with finite  $D_{\text{KL}}(P || P_0)$ ,

$$D_{\text{KL}}(P || P_0) = D_{\text{KL}}(P || P_{\text{acc}}) - \log a. \quad (11)$$

On the accepted support,  $P_0(x | s) = aP_{\text{acc}}(x | s, i_t)$ . Substitution into  $\mathbb{E}_{x \sim P} \log(P(x)/P_0(x))$  gives Eq. 11. The nonnegativity of KL then identifies  $P_{\text{acc}}$  as the closest such distribution to  $P_0$  in  $D_{\text{KL}}(P||P_0)$ .

This standard conditional-distribution identity makes the role of the item label explicit: it defines which expressions are admissible, while their relative proposal probabilities are retained within the accepted set. Our SFT uses finite accepted samples assembled into responses, so the identity describes query selection rather than an exact solution for the trained response distribution. The identity assumes a nonzero acceptance probability.

## C.2 Query-Level Ranks and Fused Utility

Consider RRF [3] over complete ranked lists, with a fixed constant  $c > 0$  and score

$$S_q(i) = \sum_{j=1}^K \frac{1}{c + \text{rank}_R(i | q_j)}. \quad (12)$$

LEMMA C.1 (INDIVIDUAL RANKS DO NOT DETERMINE FUSED RANK). *For any  $c > 0$ , the target-rank vector alone does not determine the target's RRF rank over general collections of complete ranked lists.*

PROOF. For target  $t$  and competing items  $a, b$ , consider two lists per set:

| Set | List 1      | List 2      | Target ranks | Fused rank |
|-----|-------------|-------------|--------------|------------|
| A   | $(a, t, b)$ | $(a, t, b)$ | $(2, 2)$     | 2          |
| B   | $(a, t, b)$ | $(b, t, a)$ | $(2, 2)$     | 3          |

In A, item  $a$  remains above  $t$  and item  $b$  remains below it after fusion. In B, each competitor obtains one first and one third position, giving the score difference

$$\begin{aligned} S_B(a) - S_B(t) &= S_B(b) - S_B(t) \\ &= \frac{1}{c+1} + \frac{1}{c+3} - \frac{2}{c+2} \\ &= \frac{2}{(c+1)(c+2)(c+3)} > 0. \end{aligned} \quad (13)$$

Both competitors therefore outrank  $t$  despite unchanged target ranks in the individual lists. Thus, even the full target-rank vector, including its best and average ranks, is insufficient to determine fused utility.  $\square$

The construction does not imply that greater query diversity necessarily improves recommendations: useful combinations depend on their effects on both the target and competing items.

## C.3 Weighted Top- $N$ Miss Risk

PROOF OF PROPOSITION 3.1. For any realized rank  $z \in \{1, \dots, L\}$ ,

$$\begin{aligned} \log(1+z) - \log 2 &= \sum_{n=1}^{z-1} [\log(n+2) - \log(n+1)] \\ &= \sum_{n=1}^{L-1} \log \frac{n+2}{n+1} \mathbf{1}[z > n]. \end{aligned} \quad (14)$$

Taking expectations proves Eq. 8, since  $\Pr(Z > n) = 1 - h_n$ ; the sum is empty when  $L = 1$ .  $\square$

For events with different eligible-pool sizes  $L_e$ , use the common upper bound  $L_{\text{max}} = \max_e L_e$  and set  $h_{e,n} = 1$  for  $n \geq L_e$ . The added terms are zero, so averaging the identity across events makes the averaged  $h_{e,n}$  the expected Recall@ $n$  in the single-target setting. Each weight  $w_n$  is positive and decreases with  $n$ . Equivalently, improving a rank from  $r+1$  to  $r$  increases reward by  $\log((r+2)/(r+1))$ , giving larger per-position gains near the head.

The expansion characterizes the loss as a weighted combination of missed-target risks, rather than as the loss at any single cutoff. A smaller expected log-rank therefore need not improve every individual Recall@ $N$  or NDCG@ $N$ . For truncated rank measurements with a finite missing-target sentinel, the same algebra applies to that measured rank variable; its interpretation is correspondingly limited to the recorded depths.

## C.4 Resolution of Group-Relative Feedback

LEMMA C.2 (RESOLUTION OF CENTERED ADVANTAGES). *For a fixed group of  $G \geq 1$  target ranks  $z_1, \dots, z_G$  and a cutoff  $N$ , let  $b_g = \mathbf{1}[z_g \leq N]$  and  $r_g = -\log(1+z_g)$ . If the centered binary-reward vector is nonzero, then the centered rank-reward vector is nonzero.*

PROOF. A centered vector is nonzero exactly when the underlying rewards are nonconstant. Nonconstant binary rewards imply distinct ranks, which yield distinct log-rank rewards by strict monotonicity.  $\square$

The converse need not hold: different ranks on the same side of  $N$  have equal binary rewards. Thus, on any fixed collection of rollout groups, rank feedback preserves at least as many groups with nonzero centered advantages as binary hit feedback, without an independence assumption.

For a probabilistic expression, additionally suppose that the  $G$  ranks are conditionally independent draws from  $p_z = \Pr(Z = z)$  for the fixed event. Let  $p_{\text{hit}} = \sum_{z \leq N} p_z$ . The probabilities of a constant-reward group are

$$\begin{aligned} P_{\text{rank}} &= \sum_{z=1}^L p_z^G, \\ P_{\text{hit}} &= p_{\text{hit}}^G + (1 - p_{\text{hit}})^G \geq P_{\text{rank}}. \end{aligned} \quad (15)$$

The inequality follows by grouping the rank probabilities into the hit and miss sets and expanding their  $G$ th powers. It holds with equality for  $G = 1$ ; for  $G > 1$ , it is strict if either set contains at least two ranks of positive probability.

Nonzero centered advantages need not yield a nonzero parameter gradient: score-function terms may cancel, and KL contributes separately. Any strictly monotonic rank reward shares this resolution property, while Appendix C.3 specifies log-rank's weighting.