

RoutePO: Personalized Routing Policy Optimization for LLM-based Multi-Channel Recall

Zaiyi Zheng^{1*}, Yaochen Zhu², Mingxuan Ju³, Tong Zhao⁴, Neil Shah³, Jundong Li¹

¹University of Virginia, ²Netflix, ³Snap Research, ⁴Scale AI

{sjc4fq, jundong}@virginia.edu, yzhu@netflix.com

{mju, nshah}@snap.com, tong.zhao@scale.com

Abstract

LLM-based rerankers in recommender systems are inherently constrained by limited context windows: they can only reorder a small retrieved candidate set, so any relevant item missed by upstream retrieval cannot be recovered at ranking time. To mitigate this dependency, we propose **RoutePO**, which repositions the LLM from an item ranker to a *recall strategy router* for multi-channel recall. Conditioned on a user’s *profile/context* (interaction history and optional demographics), the LLM outputs personalized fusion weights over heterogeneous retrieval experts, enabling dynamic candidate aggregation without exposing long candidate lists from the entire item pool to the prompt. We train the router via a synergistic two-stage alignment process: **Coarse Routing Alignment** (SFT) warm-starts the policy with a quality-aware oracle prior over retrieval experts, and **Fine-grained Metric Optimization** (RL) directly optimizes *recall-stage* ranking metrics (e.g., NDCG) by sampling simplex-valued routing weights and rewarding fused-list quality. To support continuous exploration over the simplex under GRPO-style optimization, we introduce *Soft-GRPO* based on Gumbel–Softmax reparameterization with nucleus restriction and KL regularization. Finally, we instantiate fusion with both weighted score merging and a weight-aware interleaving strategy (Snake Fusion) that preserves within-channel rankings. Experiments on three real-world datasets demonstrate consistent gains over strong individual recallers and multi-channel fusion baselines, including up to 15.88% relative improvement in NDCG@50 on Food. Our implementation is available on <https://github.com/zhengzaiyi/RoutePO>.

1 Introduction

Large Language Models (LLMs) have been increasingly integrated into recommender systems, typi-

cally serving as *rerankers* or *listwise rankers*. In this setting, an LLM receives a prompt containing a user’s interaction history and a small candidate list (e.g., “Here are 10 movies, rank them for this user”), and outputs a reordered list based on its internal knowledge and reasoning (Sun et al., 2023; Hou et al., 2024). Methods such as RankGPT (Sun et al., 2023) demonstrate that LLMs can achieve strong performance by applying listwise reasoning directly over candidates within the input context. However, ranking operates only on items that have already been retrieved. In real-world recommender systems with millions of items, the recall stage must first generate a manageable candidate pool. Therefore, recommendation quality largely depends on recall coverage: if relevant items are not retrieved, even a powerful ranker cannot recover them.

Although LLMs excel at reasoning over small candidate sets, they are not well suited to operate over the large candidate pools required during recall. Exposing hundreds or thousands of items to an LLM for global reasoning would be computationally prohibitive and incompatible with the autoregressive inference paradigm (Bao et al., 2023). As a result, LLMs are typically confined to the final reranking stage, operating only on a small top- K set produced by upstream retrieval models.

To improve candidate coverage before reranking, industrial recommender systems often employ *multi-channel recall*, where multiple lightweight recall models generate complementary candidate sets. Different channels specialize in different regimes (e.g., sparse-history vs. dense-history users, short-term vs. long-term preference patterns). However, these channels are usually combined using static heuristics or fixed weights (e.g., equal quotas across channels), which fail to adapt to context-dependent user intent. Prior work has shown that exploration, diversity, and novelty preferences vary across users, making one-size-fits-all fusion poli-

*Work done when Zaiyi Zheng was serving as a research intern at Snap Inc.

cies suboptimal (Kapoor et al., 2015; Schedl and Hauger, 2015). For example, dense-history users may benefit more from personalized collaborative signals, while sparse-history or cold-start users may rely more on robust global signals such as item popularity.

We address this limitation by repositioning the LLM from an “item ranker” to a *recall strategy router*. Instead of asking the LLM to reason directly over items, we let it decide *how candidates should be retrieved* from a pool of recall experts conditioned on the user’s profile (interaction history and optional metadata such as demographics). Formally, given a user’s natural-language context, the LLM outputs a personalized routing distribution over recall experts, which is then used to aggregate candidates across channels. In our instantiation, we select representative models from three distinct paradigms to support a controlled evaluation of RoutePO’s routing capabilities: the non-personalized heuristic paradigm (represented by POP), the collaborative filtering paradigm (represented by ITEMKNN), and the graph-based relational paradigm (represented by LIGHTGCN). This shared channel pool lets us isolate routing and fusion behavior over a diverse spectrum of retrieval signals. This design enables personalized candidate aggregation at the user level while avoiding the prohibitive cost of exposing large candidate pools to the LLM or encoding them within its parameters.

We instantiate this idea with a synergistic two-stage alignment process. **Coarse Routing Alignment (SFT)**. Stage 1 warm-starts the router with a coarse quality-aware prior by imitating an oracle selector. **Fine-grained Metric Optimization (RL)**. Building on this initialization, Stage 2 introduces a GRPO-inspired reinforcement learning phase to bridge the gap between hard selection and continuous weight allocation. While SFT provides a robust quality-aware prior, the RL stage directly optimizes non-differentiable ranking metrics (e.g., NDCG) by sampling multiple simplex-valued routing distributions, performing multi-channel aggregation, and using fused-list quality as the reward. This trains the LLM to allocate *personalized fusion weights* to recallers that maximize utility, rather than simply selecting one model. Our contributions include:

- We cast multi-channel recall as a decision process and propose an LLM-based routing framework that adapts retrieval strategies to individual user contexts.

- We develop a two-stage training paradigm with **Coarse Routing Alignment (SFT)** for robust routing priors and **Fine-grained Metric Optimization (RL)** for simplex-valued personalized weight allocation aligned with *recall-stage* ranking metrics (e.g., NDCG).
- Extensive experiments on real-world datasets demonstrate that our method consistently outperforms strong individual recallers and competitive multi-channel fusion baselines.

2 Preliminaries

In this section, we formalize the problem setting and introduce the core definitions and formulations used throughout the paper.

2.1 Base Recall Models (Channels)

Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets. Each user $u \in \mathcal{U}$ is associated with an interaction history (sequence) $\mathbf{h}_u = (i_1, \dots, i_{T_u})$, where $i_t \in \mathcal{I}$. For evaluation, we consider a truncated history $\mathbf{h}_u^{\text{eval}}$ and a ground-truth set $\mathbf{y}_u \subseteq \mathcal{I}$ (e.g., the next item(s) to be retrieved). We assume access to a collection of M base recall models (a.k.a. *channels*) $\{\mathcal{R}_m\}_{m=1}^M$. Given $(u, \mathbf{h}_u^{\text{eval}})$, each channel returns a scored & ranked candidate list

$$\mathcal{C}_m(u, K) = \{(i, s_m(u, i))\}_{i \in \text{TopK}_m(u)}, \quad (1)$$

where $s_m(u, i) \in \mathbb{R}$ is the channel-specific relevance score and $\text{TopK}_m(u)$ denotes the set of K highest-scoring items under $\mathcal{R}_m$. Following common retrieval practice, we exclude items that the user has already interacted with. In particular, we define an exclusion set $\mathcal{E}_u$ (e.g., all historical items), and only allow candidate items in $\mathcal{I} \setminus \mathcal{E}_u$. Although some domains permit repeat engagement (e.g., re-watching or replaying), we follow standard recall evaluation protocols by focusing on the discovery of new relevant items. When the full interaction set is available, if a ground-truth item appears in the user’s historical interactions, we temporarily remove it from the exclusion set during evaluation, i.e., $\mathcal{E}_u^{\text{eval}} = \mathcal{E}_u \setminus \mathbf{y}_u$, while masking all other previously interacted items.

2.2 Problem Formulation

Given user context x_u (constructed from $\mathbf{h}_u^{\text{eval}}$ and optional descriptive information of items), we have M recall channels $\{\mathcal{R}_m\}_{m=1}^M$, where each channel returns a scored top- K list $\mathcal{C}_m(u; K)$ as in Eq. (1).

Let $\mathcal{C}_u = \bigcup_{m=1}^M \text{TopK}_m(u)$ denote the union candidate pool (after masking the exclusion set $\mathcal{E}_u$). We decompose multi-channel recall into two functional components: (1) a *weight generator* (controller) $\mathcal{G}_\theta$ that produces per-user weights

$$\mathbf{w}_u = \mathcal{G}_\theta(x_u), \mathbf{w}_u \in \mathbb{R}_{\geq 0}^M, \sum_{m=1}^M w_{u,m} = 1, \quad (2)$$

and (2) a *weight-conditioned fusion function* $\mathcal{F}$ that merges the channel outputs under $\mathbf{w}_u$ and returns the final ranked list

$$\mathcal{C}(u, K) = \mathcal{F}(\{\mathcal{C}_m(u, K)\}_{m=1}^M, \mathbf{w}_u, x_u, K), \quad (3)$$

where $\mathcal{C}(u, K)$ contains K items sorted by the fused relevance. This formulation is intentionally generic: $\mathcal{G}_\theta$ can be instantiated by an LLM-based router, and $\mathcal{F}$ can be any weighted merging strategy (e.g., score fusion or quota-based selection).

2.3 Vanilla GRPO

We briefly review Group Relative Policy Optimization (GRPO), which estimates advantages via within-group reward normalization and avoids a learned value model. For each context x_u , GRPO samples a group of G routing outputs $\{\mathbf{w}_u^{(g)}\}_{g=1}^G$ from the old policy $\pi_{\theta_{\text{old}}}(\mathbf{w} \mid x_u)$. Each sample induces a fused ranked list

$$\hat{\pi}_u^{(g)} = F(\{\mathcal{C}_m(u, K)\}_{m=1}^M, \mathbf{w}_u^{(g)}, x_u, K), \quad (4)$$

and we define the scalar reward as

$$r(x_u, \mathbf{w}_u^{(g)}) = \text{NDCG}@K(\hat{\pi}_u^{(g)}, \mathbf{y}_u). \quad (5)$$

In Eq. (5), $\hat{\pi}_u^{(g)}$ denotes the ordered sequence of item IDs in the final aggregated list. Thus, $r(\cdot)$ evaluates retrieval outcomes induced by $\mathbf{w}_u^{(g)}$, not the weight vector in isolation.

Advantage estimation. We compute the normalized advantage within the group:

$$\hat{A}_g = \frac{r(x_u, \mathbf{w}_u^{(g)}) - \text{mean}_{g'} r(x_u, \mathbf{w}_u^{(g')})}{\text{std}_{g'} r(x_u, \mathbf{w}_u^{(g')})}. \quad (6)$$

Policy update. The vanilla GRPO objective follows the clipped policy-ratio form:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{x_u \sim \mathcal{D}_{\text{RL}}, \\ \{\mathbf{w}_u^{(g)}\}_{g=1}^G \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{G} \sum_{g=1}^G \min \left(\rho_g(\theta) \hat{A}_g, \right. \right. \\ \left. \left. \text{clip}(\rho_g(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_g \right) \right], \quad (7)$$

where $\rho_g(\theta) = \pi_\theta(\mathbf{w}_u^{(g)} \mid x_u) / \pi_{\theta_{\text{old}}}(\mathbf{w}_u^{(g)} \mid x_u)$ and ϵ is the clipping threshold.

3 Methodology

In this section, we introduce our LLM-based multi-channel recall framework with two stages: (1) aligning the LLM’s understanding of user signals with the distinct capabilities of different recallers, and (2) aligning the LLM’s simplex-valued routing weights with the objective of maximizing fused-list quality directly.

3.1 Coarse Routing Alignment (SFT)

We first perform supervised fine-tuning (SFT) to warm-start the router G_θ , instantiated as an LLM. This stage is intentionally coarse: it is not intended to learn the final fusion policy, but to move the router away from random or purely linguistic channel selection and initialize it with a simple oracle signal derived from recall quality. In this stage, G_θ learns personalized *hard routing* by selecting the single most suitable recaller for each user. Given a user context x_u (constructed from the truncated interaction history and optional user metadata such as demographics when available), the router predicts an expert index $m_u \in \{1, \dots, M\}$. We then use the candidate list returned by the selected recaller, i.e., $\mathcal{C}(u, K) = \mathcal{C}_{m_u}(u, K)$.

Dataset construction. We construct an SFT dataset $\mathcal{D}_{\text{SFT}} = \{(x_u, m_u)\}$, where $m_u \in \{1, \dots, M\}$ is an oracle channel label. Concretely, for each user u , we evaluate all M base recallers and select the channel whose top- K list achieves the best $\text{NDCG}@K$ against the ground-truth item set $\mathbf{y}_u$:

$$m_u = \arg \max_{m \in \{1, \dots, M\}} \text{NDCG}@K \left(\mathcal{C}_m(u, K), \mathbf{y}_u \right). \quad (8)$$

Model architecture. We instantiate G_θ as a pre-trained language model with a lightweight MLP-based classification head. Although the final routing decision is an M -way classification, we use an LLM backbone because x_u is a natural-language user profile containing heterogeneous fields and sequence-structured interaction history. The LLM provides strong semantic and contextual encoding of these textual signals, while the MLP head maps the representation to channel logits with minimal task-specific overhead. Given x_u , the model produces channel logits $\mathbf{z}_u \in \mathbb{R}^M$, and the routing

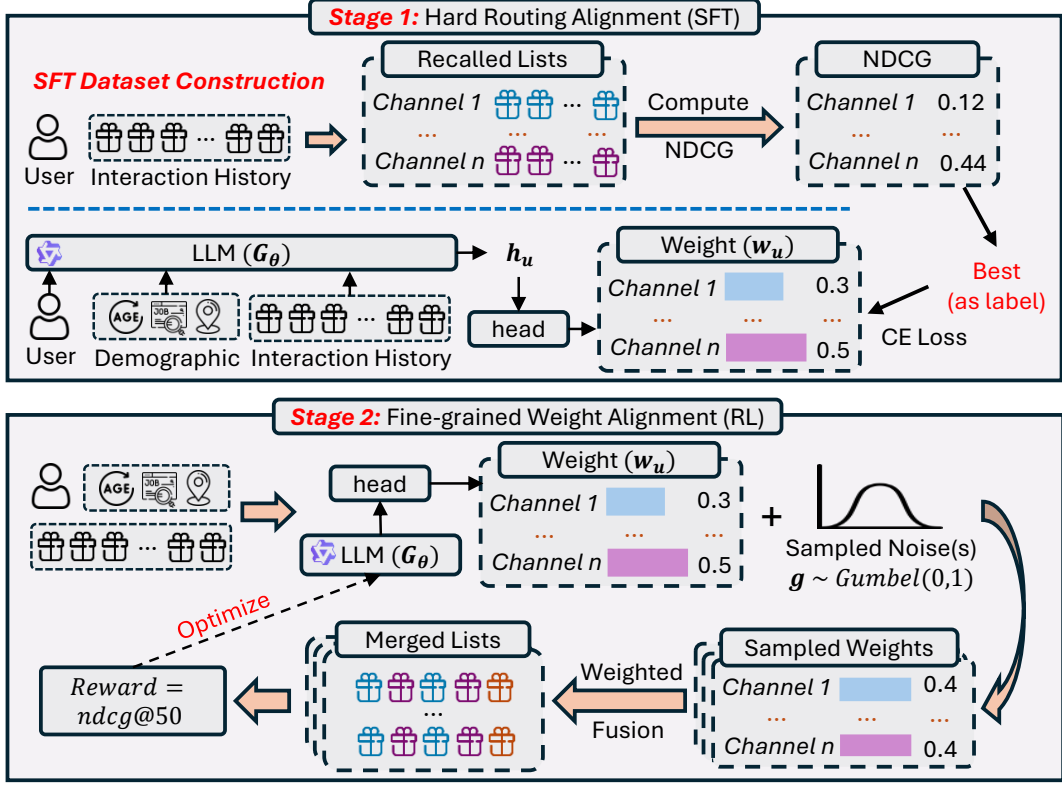


Figure 1: Overview of RoutePO. Given user demographics and interaction history, an LLM router outputs personalized fusion weights over multiple recall channels. The framework is trained in two stages: Coarse Routing Alignment (SFT) for quality-aware initialization and Fine-grained Metric Optimization (RL) using NDCG as reward. The learned weights are then used to aggregate channel-specific candidate lists via weighted fusion.

distribution is obtained by softmax:

$$\begin{aligned} \mathbf{z}_u &= G_\theta(x_u), \\ \mathbf{w}_u &= \text{softmax}(\mathbf{z}_u) \in \Delta^{M-1}. \end{aligned} \quad (9)$$

Training objective. With the hard oracle label in Eq. (8), we minimize the cross-entropy loss:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x_u, m_u) \sim \mathcal{D}_{\text{SFT}}} [\log \mathbf{w}_{u, m_u}]. \quad (10)$$

This SFT stage equips the router with an initial understanding of how different recallers perform under various user contexts, enabling it to learn meaningful associations between user signals and recaller strengths. However, hard routing only selects a single recaller and does not directly optimize the quality of the fused candidate list. To learn continuous routing weights that better support multi-channel fusion, we further introduce a reinforcement learning stage to refine the router.

3.2 Fine-grained Metric Optimization (RL)

Building on Stage 1 (Coarse Routing Alignment), this stage bridges the gap between hard expert selection and continuous weight allocation over

channels. While SFT provides a robust quality-aware prior, the RL stage directly optimizes non-differentiable ranking metrics (e.g., NDCG) on $\mathcal{D}_{\text{RL}} = \{(x_u, \mathbf{y}_u)\}$, where $\mathbf{y}_u$ denotes the ground-truth item set. The reward is a scalar ranking metric (e.g., NDCG) computed on the concrete item IDs in the final fused top- K list produced under a sampled routing distribution. That is, the reward is assigned to the aggregated retrieval result rather than to the routing weights directly.

A direct application of vanilla GRPO is hindered by the continuous nature of routing actions, which reside in the simplex-valued probability space Δ^{M-1} . To address the non-differentiability induced by discrete token sampling, we introduce Soft-GRPO with Gumbel–Softmax reparameterization, which provides a differentiable exploration path on the simplex. This formulation preserves fine-grained gradient signals that would otherwise be discarded by hard discretization, enabling the router to learn nuanced personalized weight distributions that maximize fused-list coverage.

3.2.1 Soft-GRPO for Multi-Channel Recall

To enable differentiable exploration on the simplex, we adopt **Soft-GRPO**, a soft-token GRPO variant using Gumbel–Softmax relaxation.

Gumbel–Softmax sampling. Let $\mathbf{z}_u = f_\theta(x_u)$ be the logits and $\mathbf{p}_u = \text{softmax}(\mathbf{z}_u)$. For each group sample g , we draw i.i.d. Gumbel noise $g_m^{(g)} \sim \text{Gumbel}(0, 1)$ and compute relaxed weights

$$\mathbf{w}_u^{(g)} = \text{softmax}\left(\frac{\mathbf{z}_u + \mathbf{g}^{(g)}}{\tau}\right), \quad (11)$$

$$\mathbf{g}^{(g)} = [g_1^{(g)}, \dots, g_M^{(g)}],$$

where $\tau > 0$ controls sharpness.

Nucleus (top- p) restriction. To stabilize exploration, we optionally restrict sampling to a nucleus set $\mathcal{S}_u \subseteq \{1, \dots, M\}$ defined by the smallest set such that $\sum_{m \in \mathcal{S}_u} \mathbf{p}_{u,m} \geq p$. When enabled, we apply a masked softmax on $\mathcal{S}_u$ by setting $z_{u,m} = -\infty$ for $m \notin \mathcal{S}_u$ in Eq. (11); both sampling and probability computation are then carried out on $\mathcal{S}_u$.

Soft-GRPO loss (standard form). We optimize a standard advantage-weighted policy objective, while using the Gumbel–Softmax sample $\mathbf{w}_u^{(g)}$ as a differentiable, simplex-valued “soft action” that determines the fused list and reward. Specifically, for each x_u we reuse the same group advantages $\hat{A}_g$ from Eq. (6) and minimize

$$\mathcal{L}_{\text{Soft-GRPO}}(\theta) = -\mathbb{E}_{\substack{x_u \sim \mathcal{D}_{\text{RL}}, \\ \{\mathbf{w}_u^{(g)}\} \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{G} \sum_{g=1}^G \hat{A}_g \sum_{m \in \mathcal{S}_u} w_{u,m}^{(g)} \log \pi_\theta(m | x_u) \right], \quad (12)$$

where $\pi_\theta(m | x_u) = \text{softmax}(\mathbf{z}_u)_m$ (with masking on $\mathcal{S}_u$ if top- p restriction is enabled). Intuitively, when a sample yields a higher-than-average reward within the group ($\hat{A}_g > 0$), the update increases the log-probability mass on experts that receive larger weights under that sample (and decreases it when $\hat{A}_g < 0$). Compared to vanilla GRPO, this formulation (i) keeps the sampling path differentiable via reparameterization, (ii) enables continuous exploration on Δ^{M-1} , and (iii) avoids hard discretization that discards fine-grained weight signals.

To keep the policy close to the SFT initialization, we add a KL penalty to the SFT policy $\pi_{\theta_{\text{SFT}}}$:

$$\mathcal{J}_{\text{Soft-GRPO}}(\theta) = -\mathcal{L}_{\text{Soft-GRPO}}(\theta) - \beta \cdot \mathbb{E}_{x_u \sim \mathcal{D}_{\text{RL}}} \left[\text{KL}(\pi_\theta(\cdot | x_u) \parallel \pi_{\theta_{\text{SFT}}}(\cdot | x_u)) \right], \quad (13)$$

where $\beta > 0$ controls the regularization strength.

3.3 Multi-Channel Fusion Strategies

Given fusion weights $\mathbf{w}_u$, we instantiate the fusion function $\mathcal{F}$ in two ways. We adopt *Snake Fusion* as our default strategy, while weighted score merging is used as a comparison baseline and reported in the ablation study.

3.3.1 Weighted Score Merging (Baseline)

A straightforward approach is to aggregate channel scores via weighted summation:

$$S(u, i) = \sum_{m=1}^M w_{u,m} \tilde{s}_m(u, i), \quad (14)$$

where $\tilde{s}_m(u, i)$ denotes the (optionally normalized) score assigned by channel m , and $\tilde{s}_m(u, i) = 0$ if item i is not retrieved by that channel. Items are ranked by $S(u, i)$ and the top- K are returned. Although simple and widely used, this approach directly mixes heterogeneous score scales and may dilute strong channel signals.

3.3.2 Weighted Snake Fusion (Default)

We adopt *Snake Fusion* as the default fusion strategy. Instead of aggregating heterogeneous scores across channels as in weighted score merging, Snake Fusion treats each channel as an ordered candidate stream and constructs the final list by interleaving these streams according to the user-specific weights $\mathbf{w}_u$. This avoids cross-channel score distortion and enables explicit control over channel exposure. Let $c_{u,m}(t)$ denote the number of items already selected from channel m in the first t positions of the fused list. At step t , Snake Fusion computes the *ideal deficit* $d_{u,m}(t) = w_{u,m}t - c_{u,m}(t - 1)$ and selects

$$m_t = \arg \max_m d_{u,m}(t). \quad (15)$$

The next highest-ranked unseen item from channel m_t is appended, and the counter is updated as $c_{u,m_t}(t) = c_{u,m_t}(t - 1) + 1$.

Intuitively, this procedure tracks the target exposure $w_{u,m}t$ online: at each step, the most under-allocated channel is selected, preserving intra-channel ranking while aligning the fused list with the desired routing distribution. Beyond simple interleaving, Snake Fusion provides a rigorous prefix exposure guarantee. As formally proven in Proposition 1 (Appendix C), the actual exposure of each channel remains theoretically consistent with the LLM’s predicted weights at every prefix

of the fused list, with deviation strictly bounded within $(-1, 1)$. This theoretical consistency allows RoutePO to maintain intra-channel ranking integrity while precisely executing the LLM’s personalized routing intent.

4 Experiments

In this section, we aim to answer the following research questions: **RQ1:** How well does RoutePO improve multi-channel recall quality compared with individual recallers and competitive fusion baselines? **RQ2:** To what extent does each component of RoutePO contribute to the overall performance? **RQ3:** Can RoutePO learn meaningful personalized routing distributions across different user contexts?

4.1 Experiment Settings

We introduce the experiment settings. More details (e.g., hardware information and hyperparameter configurations) are provided in Appendix A.

Datasets. We adopt three representative datasets spanning different recommendation domains to evaluate the generalization of RoutePO. Specifically, we use: (1) **ML-1M** (Harper and Konstan, 2015), a widely-used movie recommendation benchmark containing approximately 1 million ratings from 6,040 users on 3,706 movies with rich user demographics (gender, age, occupation); (2) **Steam** (Kang and McAuley, 2018), a game recommendation dataset crawled from the Steam platform, with item features including genre, developer, and price; and (3) **Food** (Majumder et al., 2019), a recipe recommendation dataset with rich meta-data such as ingredients, preparation steps, and tags. For all datasets, we follow a last-split strategy ordered by timestamp: the most recent 10% interaction of each user is used for testing, the second-most-recent 10% for validation, and the remaining for training (80%/10%/10% split). Please see Appendix A.1 for the detailed statistics of all three datasets used here.

Base Recall Models. In our instantiation, we select representative experts from three distinct paradigms to enable a controlled evaluation of RoutePO’s routing capabilities over a shared channel pool: Specifically, we employ: (1) **POP**, a non-personalized baseline that ranks items by global popularity; (2) **ITEMKNN** (Sarwar et al., 2001), an item-based collaborative filtering method leveraging item-item similarity; (3) **LIGHTGCN** (He

et al., 2020), a graph neural network that captures high-order collaborative signals via light graph convolution; All base recall models are implemented via RecBole (Zhao et al., 2021). All fusion methods are evaluated on this same channel pool to isolate the effect of routing and fusion policies. RoutePO operates above the base recallers, and is therefore orthogonal to the choice of retrieval experts and can be applied to larger or stronger recall pools.

Backbone LLMs & Prompts. We instantiate the routing LLM $\mathcal{G}_\theta$ with representative open-source language models of varying scale. Specifically, we examine: (1) **Qwen2.5-1.5B-Instruct** (Team, 2024) and (2) **Qwen3-4B-Instruct** (Yang et al., 2025) from the Qwen series. These models are chosen to study the trade-off between LLM capacity and routing performance. For each user, the LLM receives a natural-language prompt containing the user demographics (when available), interaction history, and descriptions of the available recall channels including their top- k_{prompt} retrieved items ($k_{\text{prompt}} = 3$). Detailed prompt design is illustrated in Appendix D.

Baselines. We compare RoutePO against multiple representative baselines, including individual recallers (POP, ITEMKNN, LIGHTGCN), static fusion strategies (Uniform Score and Uniform Snake), learned fusion policies (CEM Fusion and PG Fusion), and an LLM routing ablation (Zero-shot LLM). Details of baselines are in Appendix A.2.

Evaluation Metrics. All queries in the adopted datasets are accompanied by ground-truth items. We use two standard top- K ranking metrics to evaluate recall quality: (1) **NDCG@ K** (Normalized Discounted Cumulative Gain), which measures the quality of ranking; and (2) **Recall@ K** , which measures the fraction of ground-truth items successfully retrieved in the top- K list. We report results at $K \in 20, 50$, as recall-oriented settings typically require larger cutoff values to better reflect candidate coverage.

4.2 Main Results (RQ1)

In this subsection, we answer RQ1 through evaluating RoutePO with two backbone LLMs against all baselines on the ML-1M, Steam, and Food datasets. The results are shown in Table 1, where we make the following observations: (i) RoutePO achieves the best performance across all three datasets and nearly all metrics, with pronounced gains on Food (over 15% relative NDCG improvement), where

Method	ML-1M				Steam				Food			
	R@20	R@50	N@20	N@50	R@20	R@50	N@20	N@50	R@20	R@50	N@20	N@50
<i>Individual Recallers</i>												
POP	3.29	5.25	2.01	2.59	7.54	8.39	3.00	3.21	2.10	2.46	1.13	1.22
ITEMKNN	5.94	8.99	3.56	4.47	4.84	6.89	2.21	2.67	6.89	8.51	5.91	6.32
LIGHTGCN	6.03	8.58	4.01	4.78	8.25	8.62	5.50	5.60	10.11	11.41	7.22	7.59
<i>Fusion Baselines</i>												
Uniform Score	6.04	8.65	3.69	4.46	8.10	9.04	3.63	3.88	11.77	14.15	7.37	7.75
Uniform Snake	6.79	9.21	4.22	4.87	8.20	9.08	5.18	5.29	12.76	13.58	8.69	8.94
CEM Fusion	5.93	8.99	3.56	4.46	8.47	9.11	3.39	3.73	12.38	13.30	7.01	7.16
PG Fusion	6.24	9.15	3.65	4.50	8.65	9.16	4.48	4.75	13.01	14.22	7.86	8.07
Zero-shot LLM	5.97	8.80	3.83	4.67	5.56	7.01	3.24	3.56	10.85	12.02	8.55	8.86
RoutePO (1.5B)	7.09	9.41	4.39	5.10	8.80	9.30	5.65	5.78	13.16	14.35	9.82	10.16
RoutePO (4B)	7.12	9.43	4.39	5.12	8.86	9.31	5.69	5.82	13.50	14.56	10.05	10.36
Rel. Improv.%	4.86	2.39	4.03	5.13	2.43	1.64	3.45	3.93	3.77	2.39	15.65	15.88

Table 1: Main results across the ML-1M, Steam, and Food datasets. The metric values of Recall@ K (R@ K) and NDCG@ K (N@ K) are presented as percentages. The best result of each column is represented in **Bold**. We also show the relative improvement of RoutePO compared to the runner-up baseline for each metric.

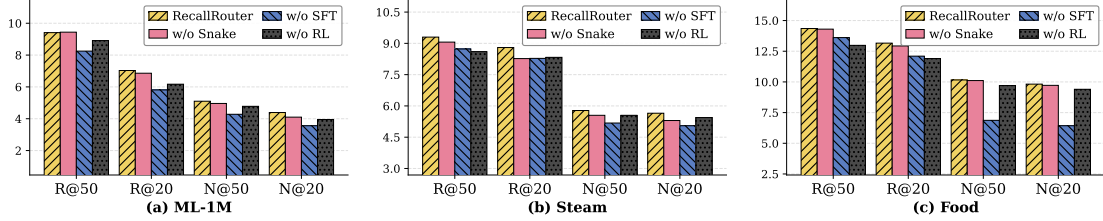


Figure 2: Ablation study of RoutePO on ML-1M, Steam, and Food.

diverse user preferences and sparse interaction histories make personalized routing especially rewarding. **(ii)** Naive fusion can degrade ranking quality when channel strengths are imbalanced: on Steam, LIGHTGCN alone achieves higher NDCG than all static fusion baselines, as uninformed fusion dilutes the dominant channel’s signal. RoutePO avoids this by learning user-specific weight allocation. **(iii)** The Zero-shot LLM baseline performs inconsistently, where its Recall@20 falls below even the POP baseline, confirming that task-specific training is essential for grounding LLM in recommendation objectives. In conclusion, RoutePO consistently achieves the best performance on both recall coverage and ranking across all datasets. Appendix B.1 provides an uncertainty analysis for the main improvements.

4.3 Ablation Study (RQ2)

We aim to answer RQ2 by evaluating the effect of different components in RoutePO with three variants: **(i) w/o SFT**: we remove the SFT warm-start stage and directly train the router with RL from a pretrained LLM checkpoint; **(ii) w/o RL**: we remove the RL stage and use only the SFT-trained router for inference; **(iii) w/o Snake**: we

replace Snake Fusion with weighted score merging as the fusion strategy. We conduct experiments with Qwen2.5-1.5B on all three datasets and show the results in Fig. 2 with the following observations: **(i)** RoutePO’s performance decreases when any component is removed, confirming that each component positively contributes to the overall routing quality. **(ii)** Removing SFT causes the most severe degradation, particularly in NDCG. The drop exceeds 30% on Food, indicating that SFT provides a crucial routing prior that RL alone cannot recover and that the two-stage design is essential rather than redundant. **(iii)** w/o SFT can achieve comparable Recall to w/o RL while exhibiting substantially lower NDCG. This asymmetry suggests that RL alone learns to diversify retrieval for broad coverage but fails to distinguish channel quality without SFT’s warm-start, confirming that SFT provides a quality-aware routing prior that RL refines rather than replaces.

4.4 Fusion Weight Analysis (RQ3)

We answer RQ3 by examining the fusion weights learned by RoutePO and comparing them with fusion baselines. In this section, we provide an aggregate-level analysis of the learned fusion

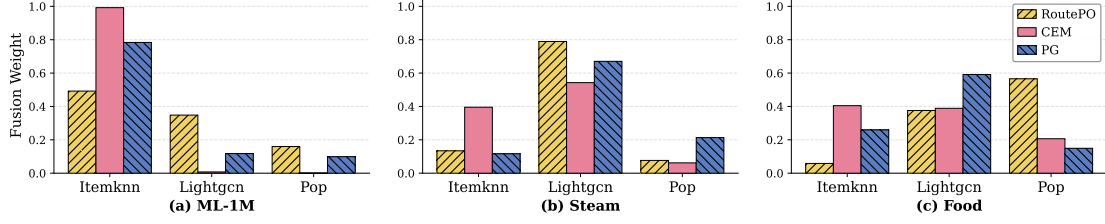


Figure 3: Comparison of average fusion weight across users on ML-1M, Steam, and Food. CEM and PG represent the cross-entropy method and policy gradient, respectively.

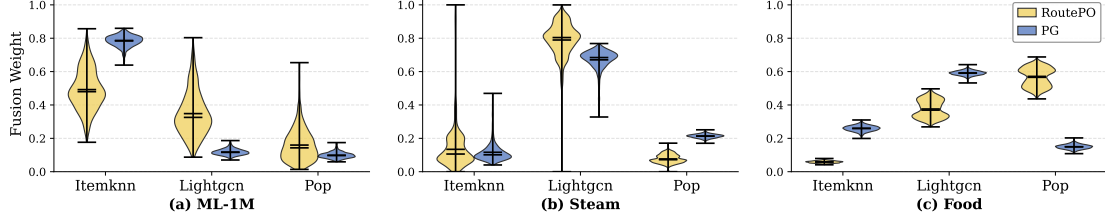


Figure 4: Comparison of fusion weight distribution across users on ML-1M, Steam, and Food. PG represents the policy gradient baseline.

weights, while representative user-level case studies illustrating RoutePO’s adaptive routing behaviors are presented in Appendix B.4. When averaged across users (Fig. 3), RoutePO and PG assign their highest weights to different channels on ML-1M and Steam, indicating distinct routing preferences. While both RoutePO and PG learn personalized weights, Fig. 4 reveals a clear difference in their per-user distributions. Specifically, at the per-user level (Fig. 4), RoutePO exhibits much larger variance in fusion weights across users, whereas PG’s weights are tightly concentrated around the mean, effectively collapsing to a quasi-global solution. For example, on Food, RoutePO assigns highly diverse weights to POP across users, while PG produces nearly identical values, indicating that RoutePO captures meaningful user heterogeneity and achieves genuinely personalized routing. Appendix B.2 further analyzes when personalized routing yields larger gains across datasets.

5 Related Works

Multi-Channel Recall Modern recommender systems usually follow a retrieve-then-rank pipeline (Covington et al., 2016; Hron et al., 2021). Recall channels include non-personalized heuristics and personalized models such as collaborative filtering, sequential retrieval, and multimodal retrieval (Huang et al., 2025a; Geng et al., 2023). Multi-channel recall improves coverage by fusing heterogeneous candidate lists (Hron et al., 2021), while MIC integrates cross-channel user-item, item-item, and user-item signals in a model-agnostic retrieval system (Nie et al., 2022). How-

ever, fixed quotas or naive interleaving are often brittle under score mismatch and distribution shift; recent work therefore studies learned channel alignment and weighting (Huang et al., 2025b).

LLMs in Recommendation Large Language Models (LLMs) are increasingly used in recommender systems, and can be integrated into different stages of the pipeline (Lin et al., 2025). A common direction adapts LLMs to model textual user-item information or recommendation instructions (Ren et al., 2024; Bao et al., 2023). Beyond feature enrichment, P5 formulates recommendation tasks as text-to-text language processing, while InstructRec adapts open-source LLMs through recommendation-oriented instruction tuning (Geng et al., 2022; Zhang et al., 2023). Recent work also integrates explicit in-text reasoning with dialogue and personalized recommendation in a unified generative framework (Liu et al., 2025).

6 Conclusion

We present RoutePO, which shifts the LLM role from item reranking to user-level routing over heterogeneous recall channels. Rather than scoring every candidate item, RoutePO uses compact user context to predict simplex-valued fusion weights that allocate recall budget across popularity, similarity, and graph-based channels. It combines Coarse Routing Alignment with Fine-grained Metric Optimization, using SFT for a quality-aware warm start and RL to directly optimize list-level metrics, while Snake Fusion stabilizes candidate aggregation. Experiments on real-world datasets show consistent gains over strong single-channel

recallers and competitive fusion baselines, and the weight and case analyses confirm meaningful user-specific routing behavior. Future work will distill and cache the router for lower-cost deployment.

7 Limitations and Future Directions

RoutePO also opens several directions for future work. First, our experiments use a controlled three-channel pool to isolate the effect of personalized routing and fusion; a natural extension is to incorporate additional modern recallers and larger expert pools. Second, as shown in Appendix A.3, the measured routing cost is modest because the router performs a single classification-style forward pass over compact user context, and future deployments can further reduce cost through distillation, caching, and batched routing. Finally, practical systems may benefit from continual adaptation to evolving user intent, catalog dynamics, and fairness-aware objectives that balance recall utility with long-tail exposure. We leave these extensions to future work.

8 GenAI Usage Disclosure

In this work, GenAI tools (ChatGPT and Claude AI) are solely used to polish the writing and check grammar issues. The authors are fully accountable for the content.

Acknowledgments

The authors would like to acknowledge the gift funding support from Snap Inc.

References

- Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *Proceedings of the 17th ACM Conference on Recommender Systems*.
- Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198.
- 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*, pages 299–315.
- Shijie Geng, Juntao Tan, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2023. Vip5: Towards multi-modal foundation models for recommendation. *arXiv preprint arXiv:2305.14302*.
- F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.
- 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*, pages 639–648.
- Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. *Proceedings of the European Conference on Information Retrieval (ECIR)*.
- Jiri Hron, Karl Krauth, Michael Jordan, and Niki Kilbertus. 2021. On component interactions in two-stage recommender systems. *Advances in neural information processing systems*, 34:2744–2757.
- Junjie Huang, Jizheng Chen, Jianghao Lin, Jiarui Qin, Ziming Feng, Weinan Zhang, and Yong Yu. 2025a. A comprehensive survey on retrieval methods in recommender systems. *ACM Transactions on Information Systems*, 44(1):1–43.
- Junjie Huang, Jiarui Qin, Jianghao Lin, Ziming Feng, Weinan Zhang, and Yong Yu. 2025b. Unleashing the potential of multi-channel fusion in retrieval for personalized recommendations. In *Proceedings of the ACM on Web Conference 2025*, pages 483–494.
- Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206.
- Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A. Konstan, and Paul Schrater. 2015. "I like to explore sometimes": Adapting to dynamic user novelty preferences. In *RecSys 2015 - Proceedings of the 9th ACM Conference on Recommender Systems*, pages 19–26. Association for Computing Machinery.
- Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xi-angyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2025. How can recommender systems benefit from large language models: A survey. *ACM Trans. Inf. Syst.*, 43(2).
- Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, Yifei Hu, Qigen Hu, Xinchun Luo, Lejian Ren, Zixing Zhang, Qianqian Wang, Kuo Cai, Yunfan Wu, Hongtao Cheng, and 7 others. 2025. Onerec-think: In-text reasoning for generative recommendation. *Preprint*, arXiv:2510.11639.

- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). *Preprint*, arXiv:1711.05101.
- Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. [Generating personalized recipes from historical user preferences](#). 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)*, pages 5976–5982, Hong Kong, China. Association for Computational Linguistics.
- Ping Nie, Yujie Lu, Shengyu Zhang, Ming Zhao, Ruobing Xie, William Yang Wang, and Yi Ren. 2022. [MIC: Model-agnostic integrated cross-channel recommender](#). In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3400–3409.
- Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE.
- Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM web conference 2024*, pages 3464–3475.
- Reuven Rubinstein. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190.
- Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.
- Markus Schedl and David Hauger. 2015. [Tailoring music recommendations to users by considering diversity, mainstreaminess, and novelty](#). In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 947–950. ACM.
- Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 14918–14937.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. [Recommendation as instruction following: A large language model empowered recommendation approach](#). *Preprint*, arXiv:2305.07001.
- Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. [Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms](#). In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21*, page 4653–4664, New York, NY, USA. Association for Computing Machinery.

A Implementation Details

All experiments are conducted on a 4-A100 server with PyTorch and HuggingFace Transformers, with base recall models built on RecBole (Zhao et al., 2021). For the **SFT stage**, we fine-tune the LLM router with AdamW (Loshchilov and Hutter, 2019) using a learning rate of 10^{-5} , per-device batch size of 4, warmup of 100 steps, and a maximum sequence length of 1,536 tokens. Training runs for 3 epochs with a constant learning rate schedule. For the **RL stage**, we use a learning rate of 10^{-6} , per-device batch size of 2 with gradient accumulation over 8 steps, $G = 8$ generations per prompt, Gumbel temperature $\tau = 1.0$, nucleus threshold $p = 0.9$, noise scale 0.1, clipping threshold $\epsilon = 0.2$, and KL penalty coefficient $\beta = 0.1$. We synchronize the reference model every 500 steps. The default number of implicit tokens at training is $k_{\text{train}} = 20$ and the evaluation top- K is set to $K = 50$. Multi-GPU training uses DeepSpeed ZeRO Stage 2 (Rajbhandari et al., 2020) with mixed-precision (bf16) across 4 GPUs. All experiments use random seed 42.

A.1 Dataset

We summarize the statistics of the three evaluation datasets after 5-core filtering in Table 2, and list the available feature fields for each dataset in Table 3. We apply 5-core filtering by iteratively removing users and items with fewer than five interactions until all remaining users and items satisfy the five-interaction threshold. Below we provide representative examples of user demographics, item metadata, and interaction records for each dataset.

ML-1M. ML-1M is a movie recommendation dataset with rich user demographics.

- **User example:** user_id = 1, age=1, gender=F, occupation=10, zip_code = 48067
- **Item example:** item_id=1, movie_title="Toy Story", release_year=1995, genre=[Animation, Children's, Comedy]
- **Interaction example:** user_id=1, item_id=1193, rating=5, timestamp=978300760

Steam. Steam is a game recommendation dataset crawled from the Steam platform, containing detailed game metadata but no explicit user demographics.

- **User features:** Not available.
- **Item example:** product_id=643980, app_name="Ironbound", developer="Secret Level SRL", genres=[Free to Play, Indie,

RPG, Strategy], price=Free To Play, sentiment=Mostly Positive

- **Interaction example:** user_id=0, product_id=328100, play_hours=51.1, timestamp=1514332800

Food. Food is a recipe recommendation dataset with rich recipe metadata including ingredients, nutrition, and preparation information.

- **User features:** Not available.
- **Item example:** item_id=137739, name="arriba baked winter squash mexican style", n_steps=11, minutes=55, nutrition=[51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0], tags=[60-minutes-or-less, mexican, vegetarian, squash]
- **Interaction example:** user_id=38094, item_id=40893, rating=4, timestamp=1045411200

A.2 Baselines

- **Individual Recallers.** Each base recall model (POP, ITEMKNN, and LIGHTGCN) is used independently as a single-channel recall method without fusion.
- **Uniform Fusion.** A static fusion strategy where each channel is assigned equal weight $w_m = 1/M$. We implement two variants: (i) *Uniform Score*, which applies equal weights under weighted score merging; (ii) *Uniform Snake*, which performs equal-proportion interleaving across channels.
- **CEM Fusion (Rubinstein, 1999).** The Cross-Entropy Method optimizes a single global fusion weight vector shared across all users by iteratively sampling, evaluating, and selecting elite weight vectors.
- **PG Fusion (Huang et al., 2025b).** A policy-gradient-based method that learns personalized fusion weights conditioned on user interaction history and learned embeddings.
- **Zero-shot LLM.** We use Qwen3-4B-Instruct with the same user-context and channel-description prompt as RoutePO, but without task-specific SFT or RL. The model is prompted to directly output a single channel name from the available set {pop, itemknn, lightgcn}. We use deterministic greedy decoding, parse the first case-insensitive occurrence of an available channel name, and map the parsed channel to a one-hot routing decision. If the output cannot be parsed, we fall

Dataset	Domain	#Users	#Items	#Interactions	Sparsity	Avg. History
ML-1M	Movie	6,040	3,416	999,611	95.16%	165.5
Steam	Game	25,389	4,089	328,278	99.68%	12.9
Food	Recipe	17,813	41,240	555,618	99.92%	31.2

Table 2: Statistics of the evaluation datasets. Avg. History represents the average interactions number of each user in the entire dataset.

Dataset	User Demographics	Item Features	Interaction Features
ML-1M	age, gender, occupation, zip_code	movie_title, release_year, genre	rating, timestamp
Steam	—	app_name, developer, genres, price, publisher, sentiment, tags	play_hours, timestamp
Food	—	name, n_steps, minutes, nutrition, tags	rating, timestamp

Table 3: Available feature fields for each dataset.

back to the first channel in a fixed deterministic order: pop, itemknn, lightgcn.

A.3 Latency and Cost

We profile the serving efficiency of RoutePO with the Qwen2.5-1.5B-Instruct router backbone in Table 4. The router performs a single classification-style forward pass over the compact user context and channel descriptions, rather than autoregressive token generation. Thus, the additional computation is limited to input encoding and one router forward pass. We report per-user latency on a single A100 80GB GPU. Total latency includes the online recall calls and the routing step, while model initialization and prompt construction are excluded. The measurements show that the extra routing step remains modest in this setting; the relatively higher latency on Food is primarily due to its longer textual prompts. In a separate single-GPU BF16 training profile on ML-1M (batch size 4 and sequence length 1,536), RoutePO takes 0.34 s per step, processes 11.9 samples/s, and uses 19.6 GB of peak GPU memory on one NVIDIA A100.

B Supplementary Experiments

B.1 Paired Bootstrap Significance Test

To check whether RoutePO’s gains are stable across users rather than driven by a particular user sample, we estimate the sampling uncertainty of its improvement over a reference single-channel baseline. For ML-1M and Steam, the reference channel is LIGHTGCN; for Food, it is ITEMKNN. For each user, we first compute the paired metric delta between RoutePO and the reference baseline. We then repeatedly resample users with replacement, recompute the mean delta for each resampled user

set, and use these bootstrap estimates to form 95% confidence intervals. Table 5 reports the resulting intervals for NDCG@50 and Recall@50.

Dataset	Reference single	$\Delta N@50$ (95% CI)	p	$\Delta R@50$ (95% CI)	p
ML-1M	LIGHTGCN	0.36 [0.21, 0.50]	$< 10^{-4}$	0.92 [0.70, 1.15]	$< 10^{-4}$
Steam	LIGHTGCN	0.03 [-0.01, 0.08]	0.187	0.45 [0.36, 0.55]	$< 10^{-4}$
Food	ITEMKNN	0.98 [0.74, 1.22]	$< 10^{-4}$	2.35 [2.06, 2.64]	$< 10^{-4}$

Table 5: Paired user-level bootstrap confidence intervals for RoutePO gains over the listed reference single-channel baseline. Improvements are reported in percentage points.

The bootstrap results support the main improvements on ML-1M and Food: both datasets have strictly positive confidence intervals for NDCG@50 and Recall@50. On Steam, RoutePO yields a robust Recall@50 improvement, but the NDCG@50 interval crosses zero. Thus, the Steam result should be interpreted as a reliable coverage gain rather than a statistically reliable ranking-quality gain over LIGHTGCN. This analysis measures robustness across users and does not estimate multi-seed training variance.

B.2 Routing Complementarity Analysis

To explain why RoutePO yields larger gains on some datasets than on others, we examine whether users benefit from different recall channels and whether the learned router captures this variation. For each user, we define an oracle channel as the single recall channel that achieves the highest NDCG@50 for that user in hindsight; this oracle is used only as a diagnostic upper-bound signal, not as a deployable method. Table 6 reports four quantities: oracle channel entropy, which measures how evenly the oracle-best channels are distributed across users; routing weight variance, computed as

Dataset	Input tokens	Routing latency (ms/user)	Total latency (ms/user)	Throughput (users/s)	GPU memory (GB)
ML-1M	1433.2	230.0	234.2	4.24	5.55
Steam	964.9	120.0	125.0	6.41	5.58
Food	3794.8	994.4	1033.5	1.14	12.21

Table 4: Serving efficiency of Qwen2.5-1.5B-Instruct RoutePO with history length capped at 20. Latency is reported as median per-user wall-clock time on one A100 80GB GPU.

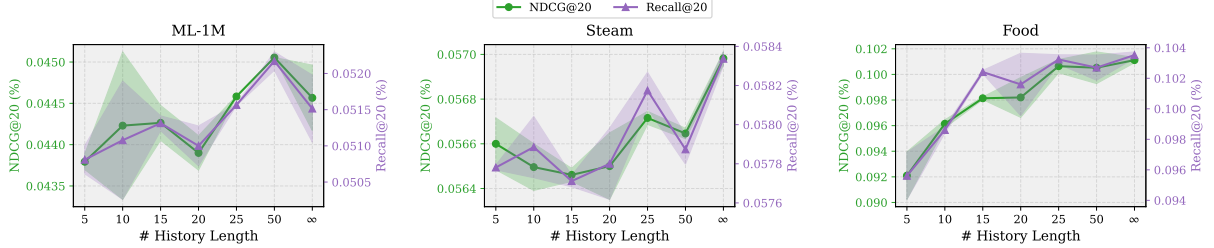


Figure 5: Effect of history length k_{hist} on NDCG@20 and Recall@20 across three datasets. The default setting ($k_{\text{hist}}=20$) is used in all main experiments.

the across-user variance of RoutePO’s predicted channel weights; the NDCG@50 gain over the corresponding reference single-channel baseline; and the fraction of oracle headroom captured by RoutePO. We use the same reference channels as in Table 5: LIGHTGCN for ML-1M and Steam, and ITEMKNN for Food. Together, these quantities test whether channel complementarity exists, whether RoutePO learns non-uniform user-specific routing weights, and whether this learned routing converts the available oracle headroom into actual ranking gains.

Dataset	Oracle entropy	Routing weight variance	N@50 gain (pp)	Headroom captured
ML-1M	1.0511	0.0248	0.36	18.8%
Steam	0.6970	0.0935	0.03	4.3%
Food	0.7150	0.1469	0.98	42.2%

Table 6: Dataset-level routing complementarity analysis. The NDCG@50 gain is reported in percentage points (pp).

The results show that oracle diversity alone does not determine the final gain. ML-1M has the highest oracle entropy, indicating substantial per-user variation in the best recall channel, but RoutePO captures a smaller fraction of oracle headroom than on Food. In contrast, Food combines meaningful oracle complementarity with the largest routing weight variance and the highest captured headroom, yielding the largest NDCG@50 gain. Steam has non-trivial oracle headroom, but its LIGHTGCN reference channel dominates the average ranking quality, leaving less usable room for learned routing. Taken together, these results indicate that personalized routing is most effective when channel

complementarity is both present and captured by the learned routing distribution.

B.3 Parameter Analysis

We investigate the sensitivity of RoutePO (Qwen2.5-1.5B) to two key prompt-design hyperparameters: the *history length* k_{hist} (number of interaction-history items included in the prompt) and the *prompt top-k* k_{prompt} (number of top-retrieved items shown per recaller). We report NDCG@20 and Recall@20 while varying one parameter and fixing the other at its default ($k_{\text{hist}}=20$, $k_{\text{prompt}}=3$). Results are shown in Figures 5 and 6.

Effect of history length. As shown in Figure 5, performance is remarkably stable across all three datasets: varying k_{hist} from 5 to the full history produces less than 3% relative change in both metrics. Even with as few as 5 history items, RoutePO achieves near-optimal routing quality, indicating that the LLM can extract sufficient preference signals from a compact context. The one exception is Food, where a monotonic improvement from $k_{\text{hist}}=5$ to $k_{\text{hist}}=25$ reflects the dataset’s diverse recipe preferences (avg. history length 31.2): more context helps disambiguate broad-interest users whose routing signal is spread across many items. Beyond 25, performance plateaus, suggesting diminishing returns from additional history items. This stability is practically valuable, as shorter prompts reduce both inference latency and token cost without sacrificing routing quality.

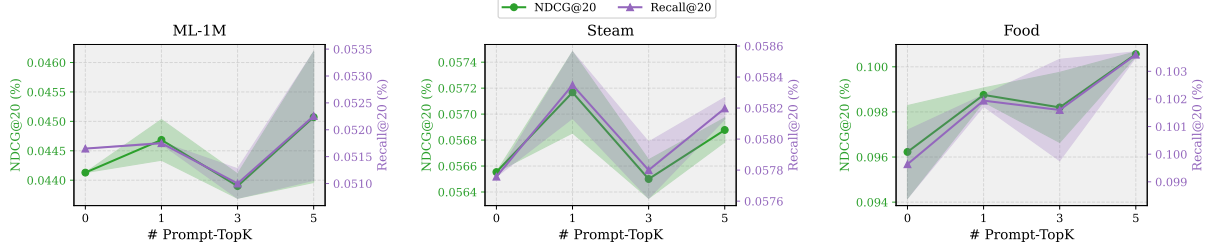


Figure 6: Effect of prompt top- k (k_{prompt}) on NDCG@20 and Recall@20 across three datasets. The default setting ($k_{\text{prompt}}=3$) is used in all main experiments.

Effect of prompt top- k . Figure 6 shows that the number of retrieved items shown per recaller has limited impact on routing performance. Notably, even at $k_{\text{prompt}}=0$, where the prompt contains *no* retrieved items and the router must decide purely from user history, RoutePO still achieves competitive results. This indicates that user interaction history alone provides strong signals for channel routing, and the retrieved-item previews serve as a supplementary rather than essential cue. On Food, performance improves modestly from $k_{\text{prompt}}=0$ to $k_{\text{prompt}}=5$, consistent with the observation that this dataset’s heterogeneous item space benefits from grounded item previews.

B.4 Case Study

We present representative cases from each dataset to illustrate how RoutePO adapts routing weights to user-specific contexts, contrasting them with static CEM weights and quasi-global PG weights.

ML-1M: Mainstream Movie Fan. User Demographic: A 35-year-old male technician/engineer (zip code 95401) with 46 movie ratings.

Interaction History: The user’s history is dominated by mainstream blockbusters: *Star Wars: Episode V*, *The Godfather* (Parts I & II), *Indiana Jones and the Last Crusade*, *Jaws*, *The Matrix*, *Saving Private Ryan*, *Braveheart*, and multiple *Star Trek* films. The ratings (mostly 3–4/5) and genre distribution (Action, Sci-Fi, Drama) indicate a preference for popular, high-budget productions.

Fusion Weights:

Method	ITEMKNN	LIGHTGCN	POP
RoutePO	22.5%	13.0%	64.5%
CEM	99.2%	0.8%	0.0%
PG	78.4%	11.8%	9.9%

Analysis: RoutePO correctly identifies that for a user with mainstream taste, the POP (popularity-based) channel is most effective. In contrast, CEM assigns 99% weight to ITEMKNN (a global op-

timum that ignores user heterogeneity), and PG similarly favors ITEMKNN at 78%. As a result, RoutePO retrieves 3 ground-truth items in the top-50 ($\text{NDCG@50}=0.248$), while CEM and PG retrieve only 1 each ($\text{NDCG@50}\approx 0.073$).

Steam: Niche Simulator Gamer. User Demographic: The user has 5 interactions in total.

Interaction History: The user plays niche simulator titles: *Farming Simulator 2013*, strategy games like *Heroes of Might & Magic V*, and simulation titles like *Fireworks Simulator*. This sparse, highly specialized demographic contrasts with the typical Steam user who plays mainstream titles.

Fusion Weights:

Method	ITEMKNN	LIGHTGCN	POP
RoutePO	94.9%	4.9%	0.2%
CEM	39.5%	54.3%	6.2%
PG	11.9%	65.3%	22.9%

Analysis: The global optimal weight on Steam favors LIGHTGCN (79%), which captures community-level preferences. However, for this niche user, item co-occurrence patterns (ITEMKNN) are more informative than graph-based signals, because users who play *Farming Simulator* tend to also play other simulators: a pattern that ITEMKNN directly captures. RoutePO assigns 94.9% to ITEMKNN and retrieves the ground-truth item (*Farming Simulator 2013 Modding Tutorials*) at position 1 ($\text{NDCG}=1.0$), while CEM and PG both place it at position 2. This demonstrates RoutePO’s *bidirectional adaptivity*: it can deviate from the global optimum in either direction based on user context.

Food: Diverse Recipe Enthusiast. User Demographic: The user has rated 34 recipes.

Interaction History: The user’s recipe history spans diverse cuisines and categories: *bacon and egg empanadas*, *baked tilapia with spice*, *lisa’s swirled chocolate chip cookies*, *tofu egg salad*, *whole wheat pizza crust*, *lentil soup*, and *chicken*

chimichangas. The lack of a coherent theme (spanning Mexican, Italian, Asian, baked goods, soups) suggests an “omnivore” user with broad interests.

Fusion Weights:

Method	ITEMKNN	LIGHTGCN	POP
RoutePO	4.5%	32.8%	62.7%
CEM	40.5%	38.9%	20.6%
PG	24.6%	57.6%	17.9%

Analysis: For a user with no clear item co-occurrence pattern (diverse recipe types do not cluster), collaborative filtering signals (ITEMKNN/LIGHTGCN) are weak. RoutePO recognizes this and assigns 62.7% weight to POP, retrieving generally popular recipes that appeal to broad tastes. CEM and PG distribute weights more evenly across channels, failing to exploit the insight that popularity is the best predictor for such users. RoutePO retrieves 2 ground-truth items ($N@50=0.288$), compared to 1 for both baselines ($N@50 \approx 0.12-0.14$).

Summary. These cases illustrate that RoutePO learns to adapt routing weights based on user-specific signals: mainstream users benefit from popularity-based retrieval, niche users benefit from item-based collaborative filtering, and users with diverse interests benefit from popularity as a fallback. In contrast, CEM learns a single global weight vector, and PG produces quasi-global weights with low variance across users, so both miss user variation.

C Prefix Exposure Guarantee of Snake Fusion

Snake Fusion provides a simple discrepancy guarantee: the exposure of each channel closely follows the desired weighted allocation throughout the ranking process.

Proposition 1. Let $\mathbf{w}_u = (w_{u,1}, \dots, w_{u,M})$ be the fusion weights for user u with $\sum_m w_{u,m} = 1$. Let $c_{u,m}(t)$ denote the number of items contributed by channel m within the first t positions of the fused list produced by Snake Fusion. Then for every channel m and prefix length t ,

$$|c_{u,m}(t) - w_{u,m}t| < 1. \quad (16)$$

Proof. For simplicity we omit the user index and write w_m and $c_m(t)$. Define the deficit before step t as

$$d_m(t) = w_m t - c_m(t-1). \quad (17)$$

Because $\sum_m w_m = 1$ and $\sum_m c_m(t-1) =$

$t-1$, the deficits satisfy

$$\sum_m d_m(t) = 1. \quad (18)$$

Snake Fusion selects the channel with the largest deficit $m_t = \arg \max_m d_m(t)$ and appends one item from that channel, giving $c_{m_t}(t) = c_{m_t}(t-1) + 1$.

For the selected channel, $c_{m_t}(t) - w_{m_t}t = 1 - d_{m_t}(t)$. Since the deficits sum to one, the largest deficit satisfies $0 < d_{m_t}(t) \leq 1$, implying $0 \leq c_{m_t}(t) - w_{m_t}t < 1$.

For any other channel $m \neq m_t$, the counter remains unchanged so $c_m(t) - w_m t = -d_m(t)$. Because $d_m(t) \leq d_{m_t}(t) \leq 1$, we have $c_m(t) - w_m t \geq -1$.

If $c_m(t) - w_m t \geq 1$, then $d_m(t) \leq -1$, which contradicts $\sum_m d_m(t) = 1$ together with the maximality of $d_{m_t}(t)$. Hence $-1 < c_m(t) - w_m t < 1$, completing the proof. $\square$

D Prompt Design

We use an inference prompt that exposes only the user profile, interaction history, channel descriptions, and short previews of each channel’s retrieved items. It does not include oracle labels or ground-truth test items. Figure 7 shows the prompt template, and Figure 8 instantiates it with a concrete user from ML-1M.

E Comparison with an Embedding-Based Router

The recall experts already encode complementary collaborative evidence: ITEMKNN captures item co-occurrence, while LIGHTGCN models latent user-item structure through graph propagation. RoutePO uses interaction-derived oracle labels and ranking rewards to learn how to fuse these expert outputs from natural-language user context. To evaluate the resulting routing signal, we compare RoutePO with a learned MLP gate that consumes LightGCN user and item embeddings on ML-1M. Both routers use the same oracle-channel supervision.

Router	Input signal	R@50 $\uparrow$	N@50 $\uparrow$
Learned-embedding MLP gate (LightGCN emb.)	Learned user/item embeddings	9.01	4.50
RoutePO	Natural-language context	9.43	5.12

Prompt Template

You are an assistant in a recommendation system. Given a user profile, output ONLY one available recaller model name.

User Profile:

Demographic: (if available)

<user demographic features>

purchased item numbers: <N>

purchase history:

 '1':

<item features>

 rating: <r>/5

 timestamp: <t>

 '2':

<item features>

 ...

 ...

Available models:

- <model_name>: <description>. When to use: <when_to_use>

Top retrieved items: [{"rank": 1, <item features>}, ...]

- ...

Your response:

Figure 7: Inference prompt template for the multi-channel routing task. Blue placeholders are dynamically filled based on the user and dataset. The demographic section is included only when available (e.g., ML-1M provides age, whereas Steam and Food do not).

Table 7: Comparison with an embedding-based router on ML-1M under the same oracle-channel supervision. Recall@50 (R@50) and NDCG@50 (N@50) are reported as percentages.

As shown in Table 7, RoutePO improves both recall and ranking quality over the learned-embedding gate. The result shows that language-conditioned routing provides useful personalized signals while retaining the collaborative evidence encoded by the underlying recall experts.

Prompt Example (ML-1M)

You are an assistant in a recommendation system. Given a user profile, output ONLY one available recaller model name.

User Profile:

Demographic:

age: '35'

gender: male

occupation: technician/engineer

zip_code: '95401'

purchased item numbers: 46

purchase history:

1:

movie_title: 'Star Wars: Episode V - The Empire Strikes Back'

release_year: 1980

genre: Action Adventure Drama Sci-Fi War

rating: 4.0/5

timestamp: 15 minutes ago

2:

movie_title: The Godfather

release_year: 1972

genre: Action Crime Drama

rating: 4.0/5

timestamp: 15 minutes ago

3:

movie_title: 'The Godfather: Part II'

release_year: 1974

genre: Action Crime Drama

rating: 3.0/5

timestamp: 15 minutes ago

...

46:

movie_title: Schindler's List

release_year: 1993

genre: Drama War

rating: 5.0/5

timestamp: just now

Available models:

- itemknn: Item-based collaborative filtering using similarity.

When to use: Item similarity patterns and co-interaction signals

Top retrieved items: [{"rank": 1, "movie_title": "Raiders of the Lost Ark", "release_year": 1981, "genre": "Action Adventure"}, ...]

- lightgcn: Graph neural network leveraging user-item connectivity.

When to use: Graph structures and higher-order neighbor relations

Top retrieved items: [{"rank": 1, "movie_title": "Terminator 2: Judgment Day", "release_year": 1991, "genre": "Action Sci-Fi Thriller"}, ...]

- pop: Non-personalized popularity-based recommendations.

When to use: Baseline or cold-start situations

Top retrieved items: [{"rank": 1, "movie_title": "Star Wars: Episode IV - A New Hope", "release_year": 1977, "genre": "Action Adventure Fantasy Sci-Fi"}, ...]

- A New Hope", "release_year": 1977, "genre": "Action Adventure Fantasy Sci-Fi"}, ...]

- A New Hope", "release_year": 1977, "genre": "Action Adventure Fantasy Sci-Fi"}, ...]

- A New Hope", "release_year": 1977, "genre": "Action Adventure Fantasy Sci-Fi"}, ...]

Your response:

Figure 8: A concrete inference prompt instance from the ML-1M dataset. The user is a 35-year-old male engineer whose history consists of 46 mainstream blockbusters. Purchase history and retrieved item lists are abbreviated.