RPTUNE: LEARNED CONTEXT CURATION FOR LLM CATALOG SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

For small merchant businesses (SMBs) whose catalogs fit within a long-context LLM, full-catalog prompting offers a compelling alternative to multi-stage retrieval designed primarily for large marketplaces with millions of items. However, fitting the full catalog into the context window does not ensure that the model can use it effectively, since LLMs do not exploit long contexts uniformly. We therefore study *in-context catalog search* through two complementary questions: (1) how to curate and present catalogs to the LLM, and (2) how to adapt the LLM for product selection on curated contexts.

We propose RPTUNE, an end-to-end framework that couples learned catalog curation with LLM post-training using automatically generated, catalog-grounded supervision. An encoder–reorganizer curator orders and prunes products guided by downstream LLM feedback, while the resulting curated catalogs in turn improve the effectiveness of LLM post-training with a context-relative reward. We evaluate RPTUNE on 7 real merchants spanning distinct retail verticals, using 100 complex conversational queries per merchant. RPTUNE consistently improves search accuracy across both proprietary and open-weight LLMs, with context curation yielding gains of up to 31.4 percentage points and post-training adding a further 10.3 points on average.

1 INTRODUCTION

Small merchant businesses (SMBs) are widespread online: Shopify alone hosts over 3 million active storefronts as of September 2026 (Store Leads, 2026). However, existing product search methods are developed primarily for large marketplaces operating over millions of items with dedicated machine learning teams and abundant behavioral signals. These assumptions do not hold for SMBs, making traditional multi-stage retrieval pipelines unnecessarily complex and costly for their catalog scales.

Long-context LLMs create a new opportunity in this regime. Among the 56 real-world Shopify storefronts we analyze, 92.9% of catalogs fit within 1M tokens, with a median catalog size of only 51K tokens. This enables a compelling alternative to conventional retrieval-based search: instead of first retrieving a small candidate set, an LLM can directly reason over the full catalog. Specifically, we find that full-catalog LLM search improves accuracy over the strongest retrieval-based approach by up to 20 percentage points.

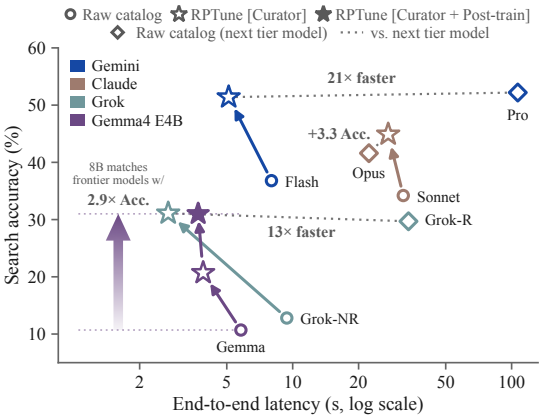


Figure 1: Search accuracy vs. end-to-end latency, averaged over 7 merchants; complete results and illustrations can be found in Table 2. RPTUNE context curation delivers accuracy gains comparable to a model upgrade with up to 21× speedup (dashed cross-model comparisons in ····), while RPTUNE post-training further improves `gemma-4-E4B-it` accuracy to comparable levels of frontier models ↑.

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

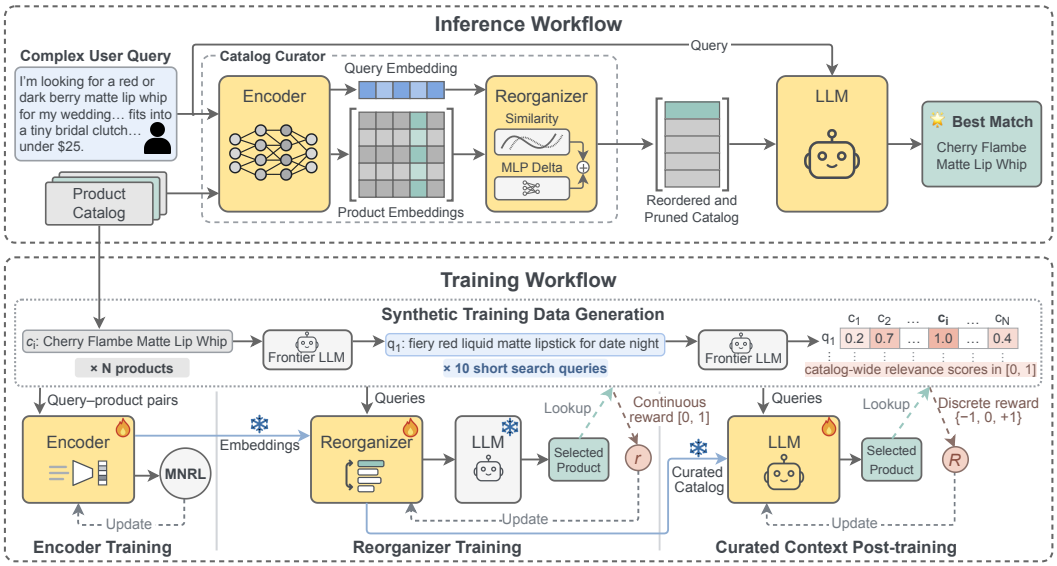


Figure 2: **RPTUNE architecture** illustrated with Q1. **Top:** At inference, the curator (encoder + reorganizer) orders and prunes the catalog to construct a compact context for downstream LLM product selection. **Bottom:** Synthetic queries and relevance scores support sequential encoder training, reorganizer training with frozen-LLM feedback, and LLM post-training on curated contexts. Flames/snowflakes indicate trainable/frozen modules at different stages.

However, simply prompting an LLM with the raw catalog leaves substantial room for improvement (Figure 1). First, the catalog itself can be curated and organized to better support LLM reasoning. Prior work has shown that LLMs are sensitive to both the ordering and length of their input context, with relevant information placed in the middle of a long context exploited less effectively than information near its beginning or end (Liu et al., 2023). Second, the LLM itself can be adapted to better interpret these curated contexts for product search.

We propose RPTUNE, a framework for in-context catalog search that combines learned context curation with LLM post-training in Section 3. As we show in Figure 2, an encoder–reorganizer curator orders and prunes products into compact, position-aware contexts, with the reorganizer trained using downstream LLM feedback. We then post-train the LLM on these curated contexts to select the best available product. All training uses synthetic, catalog-grounded supervision, requiring no merchant-provided relevance labels.

We evaluate RPTUNE on 7 real merchants spanning distinct retail verticals with 100 complex conversational queries per merchant. Across 8 evaluated proprietary LLMs, RPTUNE's context curation consistently improves search accuracy for all merchants, with gains of 14.0 percentage points on average and up to 31.4 points. These gains extend to open-weight LLMs: the full RPTUNE pipeline improves `gemma-4-E4B-it` search accuracy by 20.3 points on average, with post-training alone contributing an additional 10.3-point average gain. We highlight the resulting accuracy–latency gains in Figure 1 and report our detailed findings in Section 4.

2 PROBLEM: IN-CONTEXT CATALOG SEARCH

Use case. Consider the following query for *Beauty Bakerie* (Table 1), a cosmetics merchant:

> **Q1.** "I'm looking for a completely smudge-proof, long-wearing red or dark berry matte lip whip for my wedding day that will survive a 4-course meal and lots of champagne without budging! I'd love it to be 100% vegan and cruelty-free. Also, since I'm carrying a tiny bridal clutch, the product needs to be super lightweight, ideally under 10 grams, and I'm hoping to keep it under $25."

Q1 combines explicit requirements on color, finish, wear performance, and vegan and cruelty-free attributes with flexible preferences on weight and price, signaled by "ideally" and "hoping." It also

includes contextual cues: the bridal clutch implies a preference for compact packaging, while the wedding and meal scenario conveys expectations about durability. Answering such queries therefore requires interpreting conversational language, jointly evaluating interacting constraints, and considering trade-offs when no variant satisfies them all.

Addressing these requirements demands a holistic view of the catalog, which is feasible at SMB scale: Beauty Bakerie's entire inventory contains around 100 product variants and occupies fewer than 40K tokens, well within a modern LLM's context window. Thus, prompting an LLM directly with the catalog allows it to reason jointly over all items and select the best-matching product.

Problem setting. We study *in-context catalog search*. Given a single merchant's catalog $\mathcal{C} = \{c_1, \ldots, c_n\}$ of product variants and a single-turn natural-language query q, the task is to select the best-matching variant $\hat{c} \in \mathcal{C}$ based on the available catalog metadata. Figure 2 illustrates an example workflow for Q1. In our setting, the catalog is the only merchant-provided data source required, including for training; no merchant-provided relevance labels or user behaviour logs are needed. We focus on catalogs that fit within the LLM's context window alongside the query and instructions. Unless otherwise specified, we use *product* as shorthand for *product variant*.

3 METHOD: RPTUNE

We propose RPTUNE (**R**ank, **P**rune, and Fine-**Tune**), a framework for in-context product search that combines a *catalog curator* with a downstream LLM (Figure 2). We illustrate the inference and training workflows in Sections 3.1 and 3.2, respectively.

3.1 RPTUNE INFERENCE WORKFLOW

Given a user query q and a product catalog $\mathcal{C} = \{c_1, \ldots, c_n\}$, RPTUNE predicts the best-matching product $\hat{c} = \text{RPTUNE}(q, \mathcal{C}) \in \mathcal{C}$. RPTUNE's catalog curator comprises an *encoder*, which maps the query and products into dense representations, and a *reorganizer*, which uses these representations to select and arrange products into a compact, query-dependent context $\widetilde{\mathcal{C}}_q$ for the downstream LLM.

The encoder maps the query and each product into L_2-normalized dense representations:

$$\mathbf{e}_q = \text{Encoder}(q), \quad \mathbf{e}_{c_i} = \text{Encoder}(c_i) \quad \forall c_i \in \mathcal{C}.$$

The reorganizer uses these representations to construct a compact, query-dependent catalog context:

$$\widetilde{\mathcal{C}}_q = \text{Reorganizer}\left(\mathcal{C}, \mathbf{e}_q, (\mathbf{e}_{c_i})_{i=1}^n; \rho\right),$$

where $\rho \in [0, 1)$ is the pruning rate, specifying the fraction of catalog products to discard. The output $\widetilde{\mathcal{C}}_q$ is an ordered subset containing $k = \lceil (1-\rho)n \rceil$ products. The reorganizer determines both which products to retain and where to place them in the context.

Internally, it assigns each product a context-priority score s_i that combines embedding-based semantic matching with a learned task-specific adjustment:

$$s_i = \underbrace{\tau \cdot (\mathbf{e}_q^\top \mathbf{e}_{c_i})}_{\text{Base similarity}} + \underbrace{\text{MLP}([\mathbf{e}_q; \mathbf{e}_{c_i}])}_{\text{Learned adjustment}},$$

where $[\cdot;\cdot]$ denotes vector concatenation, MLP is a multilayer perceptron with a scalar output, and τ controls the contribution of the base similarity (we set $\tau = 10$). Let π be a permutation satisfying $s_{\pi(1)} \geq \cdots \geq s_{\pi(n)}$. We retain the top k products and arrange them in ascending order of priority, placing higher-scoring products closer to the final generation prompt: $\widetilde{\mathcal{C}}_q = (c_{\pi(k)}, c_{\pi(k-1)}, \ldots, c_{\pi(1)})$. The downstream LLM then predicts the best-matching product from the curated context: $\hat{c} = \text{LLM}\left(q, \widetilde{\mathcal{C}}_q\right)$.

3.2 RPTUNE TRAINING WORKFLOW

RPTUNE training proceeds sequentially using synthetic, catalog-grounded data; at each stage, only the target module is updated while the others remain frozen.

Algorithm 1 RPTUNE Reorganizer Training via LLM-in-the-Loop RL

Input Catalog $\mathcal{C}$; relevance data $\mathcal{D}_{\text{rel}} = \{(q, c, y(q, c))\}$, where $y(q, c) \in [0, 1]$ is the pre-computed relevance of product c to query q; frozen encoder and downstream LLM

Initialize $\theta \leftarrow$ zero-initialized output layer.
for step $= 1, \ldots, T$ **do**
  Sample a batch $\mathcal{B}$ of B queries from $\mathcal{D}_{\text{rel}}$
  for each query $q \in \mathcal{B}$ **do**
    Compute context-priority scores $\{s_i\}_{i=1}^{|\mathcal{C}|}$ and construct curated $\widetilde{\mathcal{C}}_q$ following Section 3.1.
    Sample G selections $a_g \sim \text{LLM}(q, \widetilde{\mathcal{C}}_q)$.
    For each g, retrieve $r_g \leftarrow y(q, a_g)$ from $\mathcal{D}_{\text{rel}}$ if $a_g \in \mathcal{C}$; otherwise, set $r_g \leftarrow -1$.
    Compute $A_g \leftarrow (r_g - \mu_q)/(\sigma_q + \epsilon)$ from the group mean μ_q and std σ_q.
  Update only θ using $\nabla_\theta \mathcal{L}_{\text{Reorganizer}}(\theta)$ from Eq. 1.

Output θ

Training Data Generation. We generate synthetic query–product pairs $\mathcal{D}_{\text{pair}}$ and catalog-wide relevance scores $\mathcal{D}_{\text{rel}}$ in two steps. We document the details in Appendix A.1.

Query–Product Pair Generation. For each product variant c_i in the merchant's catalog, we prompt a frontier LLM (`gemini-3.1-pro`) to generate 10 diverse, 1-2 sentence search queries targeting c_i. This yields $\mathcal{D}_{\text{pair}} = \{(q, c_q)\}$, a set of $10n$ realistic queries paired with their target variants.

Catalog-Wide Relevance Scoring. To capture partial matches and near-substitutes, we prompt the LLM to assign a relevance score $y(q, c) \in [0.0, 1.0]$ to every catalog variant c for each query q. To maintain global consistency, scoring runs in batches of 10 variants per call alongside a running dictionary of previously assigned scores. This yields a dense relevance distribution $\mathcal{D}_{\text{rel}} = \{(q, c, y(q, c))\}$ over the entire catalog for every query.

Encoder Training. The encoder must capture the distinguishing features of each product and place queries near their matching products in a shared semantic space. To achieve this, we train the encoder using the Multiple Negatives Ranking Loss (MNRL) (Henderson et al., 2017).

Given a batch of B training pairs $\{(q_i, c_i)\}_{i=1}^{B} \sim \mathcal{D}_{\text{pair}}$, the encoder generates their L2-normalized dense embeddings $\mathbf{e}_{q_i} = \text{Encoder}(q_i)$ and $\mathbf{e}_{c_i} = \text{Encoder}(c_i)$. The objective increases the similarity of each positive pair (q_i, c_i) relative to the in-batch negatives (q_i, c_j) for $j \neq i$:

$$\mathcal{L}_{\text{Encoder}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp\left(\mathbf{e}_{q_i}^\top \mathbf{e}_{c_i}\right)}{\sum_{j=1}^{B} \exp\left(\mathbf{e}_{q_i}^\top \mathbf{e}_{c_j}\right)}$$

Reorganizer Training via LLM-in-the-Loop Reinforcement Learning (RL). The encoder places queries near semantically matching products, but its pairwise training objective does not account for how the downstream LLM actually behaves when presented with a curated catalog. We therefore train the reorganizer using LLM feedback, as summarized in Algorithm 1. Only the reorganizer's MLP parameters θ are updated; the encoder and downstream LLM remain frozen. The initial zero MLP adjustment preserves the encoder-based ordering.

For each query q, the frozen LLM (`gemma-4-E4B-it`) produces G single-product selections using nonzero-temperature decoding from the same deterministic context $\widetilde{\mathcal{C}}_q$. Rewards are retrieved from $\mathcal{D}_{\text{rel}}$, with -1 assigned to outputs outside $\mathcal{C}$.[1] We standardize rewards within each query group to obtain advantages A_g, following the group-relative normalization of GRPO (Shao et al., 2024). Groups with identical rewards yield zero advantages, so we design the optimization to operate over minibatches of B queries, $\mathcal{B} = \{q_b\}_{b=1}^{B}$, where each update aggregates multiple query groups and includes informative groups with learning signals.

We use the LLM selections, weighted by their advantages, as the training signal for updating the reorganizer scores. Specifically, we define a catalog-wide distribution w_θ induced by these scores

[1] Note that $a_g \in \mathcal{C} \setminus \widetilde{\mathcal{C}}_q$ still receives the relevance rewards $y(q, a_g)$; positive advantages encourage higher context priority, allowing the reorganizer to correct potentially mistaken omissions.

and minimize the surrogate objective $\mathcal{L}_{\text{Reorganizer}}$:

$$w_\theta(c_i \mid q) = \frac{\exp(s_i/\beta)}{\sum_{c_j \in \mathcal{C}} \exp(s_j/\beta)}, \qquad \mathcal{L}_{\text{Reorganizer}}(\theta) = -\frac{1}{B}\sum_{q \in \mathcal{B}} \frac{1}{G} \sum_{g \in \mathcal{V}_q} A_g \log w_\theta(a_g \mid q), \quad (1)$$

where β is the temperature and $\mathcal{V}_q = \{\, g \in \{1, \ldots, G\} : a_g \in \mathcal{C} \,\}$ denotes the valid rollouts. Invalid outputs contribute to the group statistics but are excluded from the loss.

Note that a_g is sampled from the frozen LLM rather than from w_θ; w_θ provides a differentiable link from the LLM selections to the reorganizer scores. A positive advantage increases the score of the selected product relative to the rest of the catalog, while a negative advantage decreases it. Because these scores determine product inclusion and positioning in $\widetilde{\mathcal{C}}_q$, the update directly adjusts subsequent context construction. Every optimizer update contains informative groups with non-zero advantages, with detailed training statistics and curves reported in Appendix A.2.

LLM Post-training on Curated Catalogs. After training the catalog curator, we freeze both the encoder and reorganizer and post-train only the downstream LLM. For each query q in $\mathcal{D}_{\text{rel}}$, the frozen curator constructs a query-dependent curated catalog $\widetilde{\mathcal{C}}_q$. We then use GRPO (Shao et al., 2024) to sample G responses from the LLM and optimize its product selection over $\widetilde{\mathcal{C}}_q$.

For each sampled response, let c_{pred} denote the predicted product. Using the pre-computed relevance scores $y(q, c)$ from $\mathcal{D}_{\text{rel}}$, we define a context-relative reward: $R = 1$ if $c_{\text{pred}} = \arg\max_{c \in \widetilde{\mathcal{C}}_q} y(q, c)$, $R = -1$ if $c_{\text{pred}} \notin \widetilde{\mathcal{C}}_q$, and $R = 0$ otherwise. The -1 case penalizes hallucination where the model hallucinates a product absent from the curated catalog. The $+1$ case trains the LLM to select the highest-relevance product available in the curated catalog, rather than requiring the globally best product in the original catalog to be present. This is important when catalog curation omits the globally best variant: the LLM can still receive a positive reward for selecting the best available alternative, preserving a useful learning signal.

4 EXPERIMENTS

We design our experiments around four questions: **(RQ1)** Is in-context catalog search a feasible and effective alternative to retrieval-based search for SMBs? **(RQ2)** Does RPTUNE's context curation improve search accuracy and efficiency across different LLMs? **(RQ3)** Does post-training on curated contexts provide further accuracy gains beyond inference-time curation? **(RQ4)** How well does RPTUNE generalize to newly added products and unseen merchants?

4.1 EXPERIMENT SETUP

Table 1: Statistics of the 7 evaluated SMBs.

Merchant	Retail vertical	# Listings	# Prod.	# Tok.
Beauty Bakerie (BB)	Cosmetics	96	115	37,506
Oats Overnight (OON)	Breakfast food	142	303	39,845
The Honest Kitchen (HK)	Dog & cat food	131	214	41,954
Wilsun Custom Products (WCP)	Custom gears	110	410	81,355
Wairua Beauty (WB)	Skin & hair care	277	409	84,870
Focus & Frame Eyewear (FF)	Eyewear	186	739	110,724
Package Free (PF)	Zero-waste goods	116	278	126,958

Evaluation Data. We evaluate RPTUNE and baselines on 7 real-world SMBs spanning distinct retail verticals (Table 1). To avoid relying on sensitive user logs, we synthesize 100 complex conversational queries per merchant using a three-stage pipeline following prior work on LLM-generated evaluation data with human quality checks (Yao et al., 2024; Chen et al., 2023; Qin et al., 2023). First, a multimodal LLM (`gemini-3.1-pro`) acts as a synthetic customer, generating multi-constraint queries based on high-level merchant context (e.g., "About Us" screenshots) while strictly withholding individual product records to prevent trivial keyword matching. Second, we use selection consensus across randomized catalog orderings as a difficulty-control heuristic, retaining queries with moderate consensus (31.9% of candidates) to avoid trivially easy or overly ambiguous cases. Finally, human experts verify pilot queries and audit the final dataset to establish ground-truth labels (resulting in a 9.6% rejection rate). Details in Appendix A.3.

Models and Baselines. We evaluate RPTUNE across 9 LLM backbones from multiple model families. For each frozen backbone, we compare raw full-catalog prompting (randomly shuffled) against RPTUNE-curated contexts retaining the top 25% of variants. To isolate curation quality, we compare

the RPTUNE curator against dense retrieval (EmbeddingGemma 300M, upon which the RPTUNE encoder is trained, and Gemini Embedding 2) and listwise ranking (Jina Reranker 3.5) at the exact same 25% retention budget. We post-train `gemma-4-E4B-it` (`gemma`) to evaluate the full RPTUNE pipeline. Furthermore, using gemini-3.7-flash as a shared backbone, we compare against traditional search pipelines: standard RAG (top-10), RAG-Fusion (Rackauckas, 2024; Medrano et al., 2026), TourRank (Chen et al., 2025), and LongLLMLingua (Jiang et al., 2024). Finally, Section 4.4 compares RPTUNE's GRPO post-training objective against DPO (Rafailov et al., 2024) and IRPO (Wu et al., 2025). We provide the full implementation details in Appendix A.4.

Metrics. For each SMB, we run each method 10 times on the same query set and report averages over queries and runs. We use exact-match accuracy (EM) as our primary effectiveness metric, measuring the fraction of predictions that exactly match the ground-truth product variant. We additionally report feature reward (FR), the weighted percentage of matching attributes between the selected and ground-truth variants, to capture partial feature-level agreement (formal definition in Appendix A.5); and end-to-end latency, measured from query submission to final product selection. We report EM and FR as percentages and latency in seconds.

4.2 MAIN RESULTS

Table 2 summarizes the main results for RQ1–RQ3 with full underlying results in Appendix A.6.

In-context catalog search outperforms retrieval-based baselines (RQ1). Using gemini-3.7-flash as the common LLM backbone, directly prompting the LLM with the full catalog outperforms all baselines by at least 9.0 EM points. It also maintains single-digit latency (8 s), faster than or comparable to retrieval-based methods, yielding a substantially better accuracy–latency trade-off.

RPTUNE's curation improves both search quality and efficiency across all backbones (RQ2). Across 9 frozen LLM backbones, RPTUNE improves EM by 13.5 percentage points and FR by 10.1 points on average, with maximum average EM gains reaching 20 points. These improvements are highly consistent, increasing both metrics across all 63 backbone–merchant combinations. The effect is particularly pronounced for `grok-4.1-fast-non-reasoning`, where RPTUNE more than doubles EM from 12.8% to 31.1%. Crucially, these gains are specific to RPTUNE's learned curator rather than off-the-shelf context pruning: on gemini-3.7-flash, substituting Jina Reranker 3.5 or Gemini Embedding 2 at the same retention budget degrades EM on 5 and 3 of the 7 merchants respectively, whereas RPTUNE improves all 7. Furthermore, despite the additional curation step, RPTUNE reduces end-to-end latency for all 9 LLMs by 28% on average (and up to 71%).

Post-training provides substantial gains beyond curation (RQ3). Combining catalog curation with post-training nearly triples EM over the raw `gemma` (from 10.7% to 31.0%), yielding an additional 10.3 EM and 9.3 FR points beyond inference-time curation alone. Crucially, performance increases monotonically across all seven merchants at each stage of the pipeline (raw backbone $\rightarrow$ encoder-only $\rightarrow$ full curator $\rightarrow$ post-training), validating RPTUNE's design by confirming that learned context organization and downstream post-training provide compounding, additive benefits.

Case Study. For Q1, RPTUNE retains the ground-truth product, *Cherry Flambe Matte Lip Whip* (15 g), at the 8th position of a compact context, increasing EM from 0% to 100% for `grok-4.1-fast-reasoning` and from 0% to 70% for `gemma-4-E4B-it`. With the same curated context, RPTUNE's post-training raises `gemma`'s EM from 70% to 100%. Before post-training, `gemma` sometimes selects 23 g alternatives, describing them as "close to the lightweight preference", despite the 15 g ground-truth being available; after post-training, it consistently selects ground-truth, better matching the user's soft preference of ideally under 10 grams. We also present a detailed failure analysis in Appendix A.7.

4.3 GENERALIZABILITY TEST

For RQ4, we show that RPTUNE generalizes well to (1) newly added products within the same merchant (Figure 3a) and (2) unseen merchants (Figure 3b).

RPTUNE generalizes to new products without retraining. Merchants frequently update inventory, making constant retraining impractical. To evaluate zero-shot generalization, we train RPTUNE on a reduced Beauty Bakerie catalog, formed by randomly removing half of the products

Table 2: Main results on seven SMBs. Merchant columns report exact variant match (EM); the final block averages EM, feature reward (FR), and end-to-end latency across merchants. Arrowed values show EM/FR gains and latency reductions over the same backbone with full-catalog prompting.

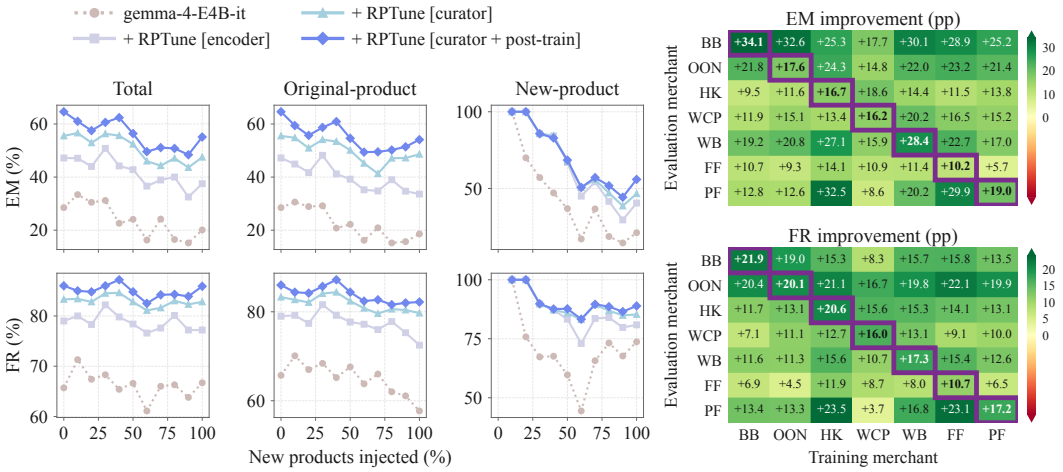
Method / Model	EM (%) ↑							Average		
	BB	OON	HK	WCP	WB	FF	PF	EM (%) ↑	FR (%) ↑	Latency (s) ↓
Main method comparison (LLM backbone: gemini-3.7-flash*)*										
RAG-Fusion	18.8	0.1	1.8	15.0	11.3	2.0	16.0	9.3	48.5	6.4
LongLLMLingua	9.1	12.3	18.4	8.6	9.1	5.3	31.4	13.5	52.2	64.2
TourRank	6.1	13.0	16.7	9.0	10.1	14.3	26.2	13.6	54.6	56.7
RAG [EmbeddingGemma 300M]	28.8	23.8	17.0	16.7	26.1	7.3	45.7	23.6	62.9	5.7
RAG [Gemini Embedding 2]	37.3	18.4	10.9	32.3	31.8	12.9	44.6	26.9	62.3	3.4
RAG [Jina Reranker 3.5]	42.5	23.8	16.7	21.9	36.5	11.5	41.9	27.8	64.1	9.2
gemini-3.7-flash	38.6	44.0	30.7	30.3	37.4	30.4	46.3	36.8	69.0	8.0
Context selectors (25% catalog retention)										
+ EmbeddingGemma 300M	34.5	39.3	25.5	20.9	44.9	26.0	45.7	33.8	67.9	7.1
+ Gemini Embedding 2	41.5	37.2	29.2	32.2	45.8	34.2	40.4	37.2	70.1	6.8
+ Jina Reranker 3.5	44.7	43.5	29.5	29.8	44.8	28.6	43.8	37.8	69.9	12.4
+ RPTUNE	51.2	56.3	51.1	42.4	60.0	46.6	52.2	**51.4** (↑**+14.6**)	77.3 (↑+8.3)	5.1 (↓**2.9**)
RPTUNE *full pipeline evaluation*										
gemma-4-E4B-it	15.2	7.5	2.5	6.6	7.8	11.1	24.3	10.7	56.7	5.8
+ RPTUNE [encoder]	25.2	9.6	10.3	12.5	14.8	13.2	28.9	**16.4** (↑**+5.6**)	60.6 (↑+3.9)	3.8 (↓**2.0**)
+ RPTUNE [curator]	38.2	11.7	12.9	13.1	17.9	14.7	36.2	**20.7** (↑**+10.0**)	65.1 (↑+8.4)	3.9 (↓**1.9**)
+ RPTUNE [curator + post-train]	49.3	25.1	19.2	22.8	36.2	21.3	43.3	**31.0** (↑**+20.3**)	74.4 (↑+17.7)	3.7 (↓**2.1**)
RPTUNE *context curation across other frozen LLM backbones*										
grok-4.1-fast-reasoning	40.6	44.0	25.7	18.1	29.8	21.6	28.1	29.7	65.4	33.8
+ RPTUNE	58.9	50.6	46.8	39.2	61.1	39.6	53.6	**50.0** (↑**+20.3**)	78.2 (↑+12.9)	26.5 (↓**7.3**)
grok-4.1-fast-non-reasoning	14.8	21.6	12.6	18.3	11.8	3.1	7.7	12.8	50.3	9.4
+ RPTUNE	46.2	39.0	22.3	31.9	29.7	22.2	26.2	**31.1** (↑**+18.2**)	69.3 (↑+18.9)	2.7 (↓**6.8**)
kimi-k2-thinking	40.1	39.4	21.9	22.4	34.6	23.3	32.0	30.5	65.8	70.7
+ RPTUNE	57.1	49.5	33.9	37.9	54.5	38.4	47.1	**45.5** (↑**+15.0**)	75.8 (↑+9.9)	40.5 (↓**30.2**)
glm-5.2	32.6	28.8	15.8	31.2	42.4	33.1	47.0	33.0	60.2	122.7
+ RPTUNE	54.1	43.4	30.5	42.5	56.0	43.7	59.2	**47.1** (↑**+14.1**)	73.9 (↑+13.7)	116.7 (↓**6.0**)
claude-sonnet-5	40.8	36.3	21.6	31.6	40.5	30.7	37.7	34.2	67.1	32.0
+ RPTUNE	54.1	43.5	34.1	36.0	52.4	39.8	54.2	**44.9** (↑**+10.7**)	74.5 (↑+7.4)	27.3 (↓**4.7**)
gemini-3.1-pro	57.5	54.7	48.8	49.1	51.2	51.5	52.5	52.2	77.0	106.7
+ RPTUNE	69.9	70.8	68.7	55.7	52.3	55.5	65.0	**62.6** (↑**+10.4**)	83.3 (↑+6.3)	81.8 (↓**24.9**)
claude-opus-5	54.0	43.8	30.0	31.0	44.4	34.1	53.7	41.6	71.5	22.3
+ RPTUNE	54.6	49.2	41.9	37.4	55.4	48.6	64.3	**50.2** (↑**+8.6**)	76.6 (↑+5.1)	21.1 (↓**1.2**)

within each product type. For evaluation queries whose original ground-truth product was removed, we re-establish the new best-matching variant by re-running our pipeline's self-consistency filtering and human verification steps on the reduced catalog. We then measure performance as the withheld products are incrementally injected back (Figure 3a) with the same 100 queries evaluated at every injection level (details in Appendix A.8). RPTUNE proves highly resilient: as the catalog doubles, post-trained RPTUNE experiences only a 14.7% relative EM decline, half of the baseline's 29.5% drop, with FR remaining practically unchanged. Crucially, RPTUNE *consistently outperforms the baseline at every injection level* for queries targeting both original and new products. Even at double the initial catalog size, RPTUNE maintains absolute gains of 35.0 EM and 19.2 FR over the baseline.

RPTUNE generalizes to unseen merchants. To evaluate zero-shot transfer across domains, we cross-evaluate each of our seven merchant-specific RPTUNE models on all available catalogs (Figure 3b). The results are uniformly positive: *all* 7×7 *combinations improve both EM and FR over the* `gemma` *baseline*. These robust improvements persist despite severe lexical shifts across catalogs with mean pairwise Jaccard similarity of 0.146.

4.4 Ablation Analysis

We ablate five design choices in RPTUNE: (1) post-training objective (Figure 4); (2) use of curated contexts during post-training (Figure 5); (3) pruning budget (Figure 6); (4) context ordering (Figure 7); and (5) reorganizer training objective. Supplementary FR results in Appendix A.9 are consistent with all findings in this section.



(a) Generalizability to new products

(b) Generalizability to unseen merchants

Figure 3: RPTUNE generalizes well without retraining. (a) EM and FR on Beauty Bakerie as withheld products are injected back, for all queries and for queries whose target is an original or a new product. 0% is the reduced catalog RPTUNE was trained on and 100% restores the full catalog. (b) Gains over `gemma` for every train/eval merchant pair, **all positive**; boxed diagonal cells are in-domain. Average out-of-domain gains are +18.1 EM and +13.6 FR, in-domain +20.3 and +17.7.

RPTUNE improves performance across post-training objectives. We compare GRPO with DPO and IRPO on the same synthetic training data, with and without RPTUNE curation (Figure 4), and additionally train GRPO on RPTUNE-curated contexts with two alternatives to our context-relative reward: (i) a *continuous* reward, $R = y(q, c_{\text{pred}})$, and (ii) a *global* top-1 reward that grants $R = 1$ only when c_{pred} is the best variant across the entire $\mathcal{C}$ rather than in $\widetilde{\mathcal{C}}_q$, on Beauty Bakerie. All methods consistently improves the `gemma` baseline, validating our training data pipeline. RPTUNE curation further enhances all methods (averaging +22.2 pp EM and a 31% latency reduction). Of all, GRPO achieves the strongest overall performance and benefits most from RPTUNE's curated context. RPTUNE's context-relative reward is essential: EM drops 10 pp with both (i) and (ii).

RPTUNE-curated contexts improve post-training. We isolate the impact of context curation during post-training by training `gemma` on either raw (R) or curated (C) catalogs and evaluating on both on Beauty Bakerie (Figure 5). Beyond accelerating training via shorter rollouts, training on curated contexts strictly improves downstream model quality across all evaluation settings. Specifically, curated training improves EM by 5.7 points on raw evaluation contexts (C/R vs. R/R) and 9.0 points on curated evaluation contexts (C/C vs. R/C). Ultimately, applying curation at both training and inference (C/C) maximizes EM and cuts latency by 35% versus the raw baseline (R/R).

RPTUNE maintains gains across pruning budgets. We sweep the context retention budget from 5% to 100% for `gemma` with full RPTUNE (Figure 6). First, RPTUNE outperforms the raw full-catalog baseline at every tested budget, including full retention. Second, our curation achieves high recall with small contexts: RPTUNE's default 25% retention rate yields 77.1% average context recall (over 3× better than random selection), while 50% retention reaches 90.7%. Finally, downstream accuracy peaks at an intermediate budget: on Beauty Bakerie, EM and FR peak at 25% retention (with 92% recall). Larger budgets add marginal recall but lower accuracy and raise latency: RPTUNE's pruning effectively filters out noise for this sweet spot.

RPTUNE improves performance across context orderings. We evaluate the raw and RPTUNE-post-trained `gemma` on Beauty Bakerie under different context layouts (Figure 7). Baseline contexts use *(1)* random, *(2)* native, or *(3)* reversed native ordering of the full catalog, whereas, based on context priority scores, RPTUNE-curated contexts use *(4)* ascending, *(5)* descending, *(6)* U-shaped (most relevant at both ends), or *(7)* randomly shuffled ordering on the 25% retained items. For both models, every RPTUNE curated layout outperforms all baseline layouts, gaining 16.8 EM points while cutting latency by 32%, where the ascending order performs best, followed by U-shaped.

RPTUNE's LLM-in-the-loop reorganizer objective optimizes contexts for LLM interpretation. To isolate the value of LLM feedback, we replace the reorganizer objective in Eq. 1 with a supervised

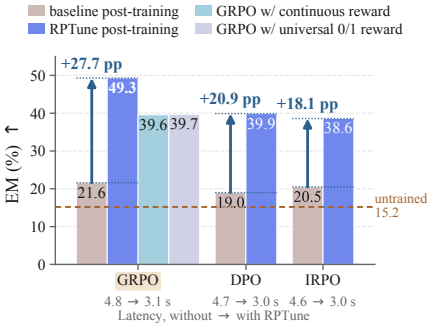


Figure 4: RPTUNE improves `gemma`'s EM for all post-training objectives with low latency. RPTUNE uses GRPO.

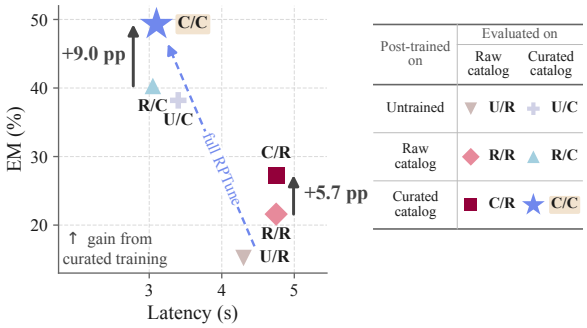


Figure 5: EM vs. latency for every post-train/eval combination; C/C is RPTUNE. Post-training on curated contexts consistently improves search accuracy.

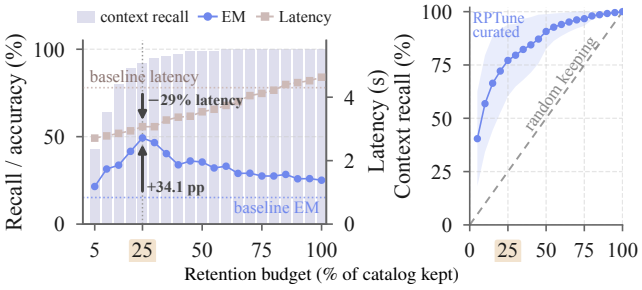


Figure 6: (Left) Pruning sweep on Beauty Bakerie. (Right) Context recall over 7 merchants; shading is 1 STD across merchants. RPTUNE uses 25% as the default budget.

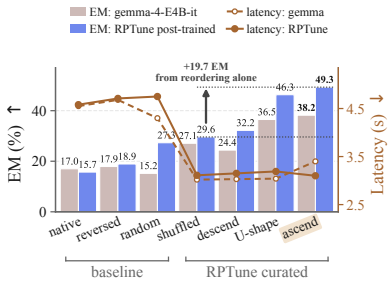


Figure 7: EM vs. latency for catalog ordering variants on Beauty Bakerie; RPTUNE uses ascend as default.

listwise cross-entropy that fits w_θ directly to the relevance labels in $\mathcal{D}_{\text{rel}}$ (Appendix A.10). With frozen `gemma`, the supervised reorganizer reaches 16.4% average EM across all 7 merchants, no better than the encoder alone and 4.3 points below RPTUNE's reorganizer.

5 Related Work

Retrieval and context construction. RAG retrieve candidates using lexical methods (Robertson & Zaragoza, 2009), dense encoders (Gemini Team, Google, 2026; Gemma Team, 2025; Zhang et al., 2025), or rankers (Zhang et al., 2025; Nasika et al., 2026). More elaborate frameworks combine LLMs with fusing multi-query retrieval (Rackauckas, 2024) or tournament-based ranking (Chen et al., 2025). Unlike likelihood-supervised retrieval (Shi et al., 2024; Lin et al., 2024), RPTUNE learns to curate catalogs from relevance rewards on LLM selections. Beyond retrieval, query-aware compression and context reordering improve the information density of LLM inputs (Jiang et al., 2023; 2024). We show that in-context product search is more promising for SMBs.

LLM post-training. LLM post-training adapts models using preference supervision or task-specific rewards. DPO (Rafailov et al., 2024) learns from preferred and dispreferred responses, which IRPO (Wu et al., 2025) extends to ranked candidate lists. RL-based methods (Shao et al., 2024; Yu et al., 2025) have improved LLM performance further on complex reasoning tasks. Building on these advances, we post-train LLMs for in-context product search on curated catalogs.

6 Conclusion

We introduced RPTUNE, an in-context catalog search framework for SMBs. Trained entirely via synthetic supervision, RPTUNE uses downstream LLM feedback to curate compact, position-aware catalog contexts, which subsequently enhance LLM post-training for product selection. Across 700 complex conversational queries from 7 real-world merchants, RPTUNE significantly improves search accuracy for all 9 evaluated proprietary and open-weight LLMs at low latency.

AI USE STATEMENT

In this work, we used generative AI (`gemini-3.1-pro`) to generate synthetic data sets: the training data described in Section 3.2 and Appendix A.1, and the evaluation data described in Section 4.1 and Appendix A.3. We verified all generated data, as documented in those sections. We did not use generative AI for any other task requiring disclosure: no LLMs were used in the ideation, writing, or preparation of this paper. All content was conceived, drafted, and revised solely by the authors. The authors take responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human subjects or personally identifiable information. Our experiments use publicly available, non-sensitive data (Appendix A.3).

REPRODUCIBILITY STATEMENT

Our method is described in Section 3, and the experimental setup, including hyperparameters and hardware, in Section 4.1 and Appendix A.4. Training curves are reported in Appendix A.2. The data generation pipeline, including prompts and verification procedures, is documented in detail in Appendix A.1 and Appendix A.3. Because the data were generated with a proprietary model whose outputs may change across versions, we will release the exact generated datasets, along with all code and model checkpoints, upon acceptance.

REFERENCES

Anthropic. Claude Opus 5. Claude Platform Documentation, 2026a. URL https://platform.claude.com/docs/en/models/opus-5/overview.

Anthropic. Claude Sonnet 5. Claude Platform Documentation, 2026b. URL https://platform.claude.com/docs/en/models/sonnet-5/overview.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation, 2023. URL https://arxiv.org/abs/2309.01431.

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Xinyu Ma, Wei Yang, Daiting Shi, Jiaxin Mao, and Dawei Yin. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy, 2025. URL https://arxiv.org/abs/2406.11678.

Gemini Team, Google. Gemini Embedding 2: Generalizable multilingual text embeddings from Gemini, 2026. URL https://arxiv.org/abs/2603.07891.

Gemma Team. EmbeddingGemma: Powerful and lightweight text representations, 2025. URL https://arxiv.org/abs/2509.20354.

Google DeepMind. Gemini 3.1 Pro, 2026a. URL https://deepmind.google/models/gemini/pro/.

Google DeepMind. Gemini 3.7 Flash: Model card, August 2026b. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Google DeepMind. Gemma-4-E4B-it. Hugging Face model card, 2026c. URL https://huggingface.co/google/gemma-4-E4B-it.

Daniel Han, Michael Han, and Unsloth team. Unsloth, 2023. URL https://github.com/unslothai/unsloth.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply, 2017. URL https://arxiv.org/abs/1705.00652.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression, 2024. URL https://arxiv.org/abs/2310.06839.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. Ra-dit: Retrieval-augmented dual instruction tuning, 2024. URL https://arxiv.org/abs/2310.01352.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Luigi Medrano, Arush Verma, and Mukul Chhabra. Scaling retrieval augmented generation with rag fusion: Lessons from an industry deployment, 2026. URL https://arxiv.org/abs/2603.02153.

Moonshot AI. Kimi-K2-Thinking. Hugging Face model card, 2025. URL https://huggingface.co/moonshotai/Kimi-K2-Thinking.

Christina Nasika, Feng Wang, Antonis Krasakis, and Han Xiao. jina-reranker-v3.5: An efficient listwise reranker with hybrid attention and self-distillation, 2026. URL https://arxiv.org/abs/2607.18152.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Zackary Rackauckas. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47, February 2024. ISSN 2319-4111. doi: 10.5121/ijnlc.2024.13103. URL http://dx.doi.org/10.5121/ijnlc.2024.13103.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Store Leads. The state of shopify in 2026, September 2026. URL https://storeleads.app/reports/shopify. Updated September 4, 2026. Accessed September 11, 2026.

Junda Wu, Rohan Surana, Zhouhang Xie, Yiran Shen, Yu Xia, Tong Yu, Ryan A. Rossi, Prithviraj Ammanabrolu, and Julian McAuley. In-context ranking preference optimization, 2025. URL https://arxiv.org/abs/2504.15477.

xAI. Grok 4.1 Fast and Agent Tools API, November 2025. URL https://x.ai/news/grok-4-1-fast.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Z.ai. GLM-5.2. Hugging Face model card, 2026. Accessed: 2026-09-14.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

A Appendix

A.1 Training Data Generation

We generate synthetic query–product pairs $\mathcal{D}_{\text{pair}}$ and catalog-wide relevance scores $\mathcal{D}_{\text{rel}}$ in two steps.

Query–Product Pair Generation. For each product variant c_i in the merchant's catalog, we prompt a frontier LLM (`gemini-3.1-pro`) to generate 10 diverse search queries that intentionally target c_i. To ensure diversity and realism, the generation prompt enforces that the queries span various user intents (e.g., shade-matching, finish-specific, and/or occasion-based) and combine multiple product features (e.g., price, taxonomy tags, size, and/or product type) into a short, natural 1-2 sentence search query. This yields $\mathcal{D}_{\text{pair}} = \{(q, c_q)\}$, a set of $10n$ realistic queries, each paired with its target variant c_q. For example, one query generated for the ground-truth variant of Q1 in *Beauty Bakerie* (`Cherry Flambe Matte Lip Whip`) is ``fiery red liquid matte lipstick for date night''.

Catalog-Wide Relevance Scoring. A single positive per query is too coarse a signal: retrieval training also needs to know which variants are partial matches or near-substitutes. For each query q, we therefore prompt `gemini-3.1-pro` as an LLM relevance judge to assign a graded score $y(q, c) \in [0.0, 1.0]$ to every variant c in the catalog. Scoring runs in batches of 10 variants per call, and each call additionally receives a running dictionary of the scores already assigned for q; the judge may revise earlier scores so that the final ranking is globally consistent across the catalog. The result is $\mathcal{D}_{\text{rel}} = \{(q, c, y(q, c))\}$, a dense relevance distribution over the entire catalog for every query. Note that the target product c_q used during query generation does not necessarily receive a score of 1.0.

This relevance-scoring procedure is distinct from in-context catalog search: it assigns graded scores to query–product pairs rather than selecting a single best-matching product for a user query. Two design choices simplify the labeling task. First, each call evaluates only 10 product records, rather than processing all product descriptions at once. Second, the synthetic queries are short and simple, in contrast to the complex, multi-constraint real-world conversational queries (e.g., Q1).

A.2 Training Curves

We use $G = 16$ rollouts per query and minibatches of $B = 128$ queries for reorganizer training. Figure 8 reports the average relevance score of the products selected by the frozen LLM during training, which increases steadily over the course of training. Every optimizer update contains informative query groups with non-zero advantages, with at least 44 such groups per update. The invalid-output (where $r_g = -1$) rate is only 0.83%, while cases in which different selected products receive identical relevance scores account for 6.2% of query groups; thus, the training signal is well-defined and informative.

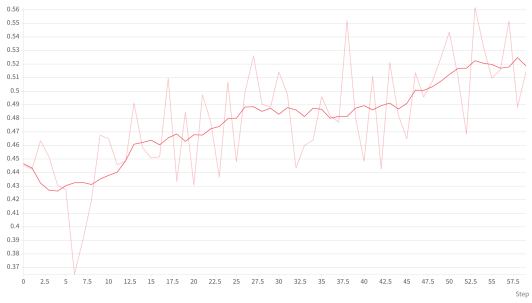


Figure 8: Average relevance score of the products selected by the frozen LLM during RPTune reorganizer training.

A.3 Evaluation Data

We evaluate RPTune and baselines on 7 real-world SMBs spanning distinct retail verticals, including pet supplies, sustainable lifestyle, and eyewear.

Catalog collection. All seven storefronts run on Shopify, which exposes the full product listing at a public endpoint: appending `/products.json` to the storefront URL returns every product together with its variants, options, prices, images and descriptions. The entire catalog is therefore obtainable in a single request per merchant, with no scraping, credentials or merchant cooperation required.[2] This is precisely the regime RPTUNE targets: an SMB can hand its whole catalog to a long-context model without any retrieval infrastructure. The storefronts are Beauty Bakerie (`https://www.beautybakerie.com/`), Oats Overnight (`https://oatsovernight.com/`), The Honest Kitchen (`https://www.thehonestkitchen.com/`), Wilsun Custom Products (`https://wilsuncustomproducts.com/`), Wairua Beauty (`https://wairuabeauty.com/`), Focus & Frame Eyewear (`https://focusandframeeyewear.com/`) and Package Free (`https://packagefreeshop.com/`).

To avoid relying on sensitive real-user search logs, we design a three-stage pipeline to synthesize conversational queries that reflect realistic user intents without trivially matching individual product descriptions following prior work on LLM-generated evaluation data with human quality checks (Yao et al., 2024; Chen et al., 2023; Qin et al., 2023). The resulting evaluation set contains 100 queries per merchant.

Stage 1: *Query Generation.* We instruct a multimodal LLM (`gemini-3.1-pro`) to act as a synthetic customer by providing it with high-level context: (1) general merchant background, (2) a full-page screenshot of the merchant's "About Us" webpage capturing brand identity and aesthetics, and (3) aggregate catalog feature distributions (e.g., product types, tags, prices, and weights). Importantly, we withhold individual product records to discourage queries tailored to specific product descriptions. We prompt the model to generate a single natural-sounding and realistic conversational query with multiple complex constraints to encourage reasoning about trade-offs rather than simple keyword matching.

Stage 2: *Query Refinement.* We provide each candidate query and the complete catalog to the same model and collect 10 recommendations, each using a different randomly shuffled catalog order. In each trial, the model selects the single product variant that best satisfies the query's constraints.

We define the *consensus score* as the maximum number of trials selecting the same variant. Because catalog order is randomized across trials, a query with one clearly best variant is resolved consistently regardless of order, whereas a query with several near-equivalent candidates yields selections that vary with order. We therefore use consensus as a heuristic for query difficulty (see *Evaluation robustness to consensus-based filtering* in the evaluation characteristics below): scores ≥ 8 indicate highly stable selections characteristic of easy queries, whereas scores ≤ 3 indicate unstable selections, reflecting potential ambiguity or overly restrictive constraints. We retain queries with moderate consensus (scores between 4 and 7, inclusive). For queries with a unique maximum, we assign the variant that achieved this consensus as the reference label. In cases where multiple variants tie for the maximum score (which occurs in 10.9% of the retained queries), we preserve the set of tied variants as candidate labels to be resolved in Stage 3. For queries outside this range, we return the query and its consensus score to the generator in Stage 1, instructing it to introduce more complex trade-offs when the score is too high and to relax or clarify constraints when it is too low. This refinement loop converges in $\sim$3 rounds on average.

Stage 3: *Human Verification.* For each merchant, we first generate 10 pilot queries using Stages 1 and 2. An expert assesses whether each query is natural-sounding and realistic, then selects the best-matching product variant without seeing the reference label(s). We compare the expert's selections with the reference labels and revise the generation prompt to address any issues identified during review. Large-scale generation begins only after all 10 pilot queries pass expert review.

After large-scale generation, we manually audit a 10% subset of standard queries and all tied cases using the same blinded selection procedure. For tied cases, if the expert's selection matches a variant within the tie set, that specific variant is finalized as the single ground-truth label. Any disagreements are reviewed by a panel of 6 experts, and pairs whose reference labels are not supported by full expert consensus are discarded. The overall rejection rate across this auditing process is 9.6%.

Evaluation characteristics. We discuss the following four aspects of the evaluation data.

[2]The endpoint pages at 250 products per request, so catalogs above that size are fetched with `?limit=250&page=`n.

Query-level distribution shift. The generated evaluation queries represent out-of-distribution data relative to the training data described in Section 3.2. While the training queries are typically short, concise, and easily resolved by standard LLMs, our evaluation queries combine multiple interacting constraints, implicit preferences, and trade-offs in natural conversational language. Q1 is a representative evaluation query for *Beauty Bakerie* that introduces conflicting constraints (e.g., wedding-grade performance vs. a sub-$25 budget) and implicit requirements (e.g., compact clutch-sized packaging), establishing a highly realistic and challenging retrieval scenario.

Evaluation scope. We focus on *single-turn* queries and leave multi-turn evaluation to future work. Nevertheless, these queries consolidate preferences and constraints that could emerge across multiple turns of a shopping conversation. For example, Q1 could be decomposed into successive turns specifying shade and finish, occasion and wear context, ethical preferences, product weight, and budget. RPTUNE can be easily adapted to this setting by running the encoder–reorganizer at each turn, conditioned on the accumulated constraints.

Validity of model-assigned labels. Because `gemini-3.1-pro` assigns the reference labels, they may favor Gemini-family backbones. We address this risk in three ways. First, Stage 3 includes blinded human verification of the pilot queries, all tied cases, and a 10% sample of the remaining queries, which rejects 9.6% of query–label pairs overall. Second, RPTUNE's gains are not specific to the labeler's family: in Table 2, the average gain for Grok (+19.3 EM) exceeds that for Gemini (+12.5 EM). Third, we audit the labels with a panel of independent LLM judges.

For the audit, we sample 154 evaluation queries, stratified by merchant (22 per merchant). Each query's candidate pool (11.1 candidates on average) draws on four sources: (i) the reference label and every variant selected in the labeler's 10 Stage 2 runs; (ii) `claude-opus-5`'s modal selection over 10 shuffled full-catalog runs, i.e., Stage 2 with a different labeler; (iii) the selections of the evaluated backbones for which we retain per-run predictions (`glm-5.2`, `kimi-k2-thinking`, both Grok backbones, and `gemini-3.7-flash` with Gemini Embedding 2 selection for every merchant, plus `gemini-3.1-pro`, `gemini-3.7-flash`, `claude-opus-5`, and `claude-sonnet-5`); and (iv) five distractors: the three top-ranked BM25 results not already in the pool and two random catalog items. Four non-Gemini judges (`claude-opus-5`, `gpt-oss-120b`, `qwen3-235b-a22b-instruct-2507`, and `deepseek-r1-0528`) see each pool in random order with sources hidden. They grade every candidate from 0 to 3 (0: wrong product; 1: right kind of product but misses a stated priority or firm requirement; 2: a defensible recommendation that satisfies the customer's top priorities; 3: meets every stated requirement) and name the single best variant. Each judge makes three passes over reshuffled pools, yielding 12 judgments per candidate (4 judges $\times$ 3 passes). A variant is *acceptable* if at least 6 of these judgments grade it ≥ 2.

The labels are informative rather than arbitrary. The panel accepts the reference label for 64.9% of queries, compared with 1.1% for BM25 distractors and 0% for random ones. It accepts `claude-opus-5`'s full-catalog selection for only 55.8% of queries, even though `claude-opus-5` is itself a judge and may favor its own choice, whereas no judge shares the labeler's model family. A single reference nonetheless cannot capture every acceptable answer, because many queries admit several defensible choices. On average, 1.85 variants per query are acceptable (44.8% of queries have at least two), and on identical pools, pairwise agreement between judges on the best variant ranges from only 36.4% to 46.1%. `claude-opus-5`'s full-catalog selection matches the reference label for 44.2% of queries. That falls within the inter-judge range, so the labeler and an independent model disagree no more often than the judges disagree with each other. We therefore do not interpret `gemini-3.1-pro`'s absolute scores in Table 2 as evidence that it is the strongest backbone. Our conclusions about RPTUNE instead emphasize within-backbone improvements observed across both Gemini and non-Gemini models.

Evaluation robustness to consensus-based filtering. Because Stage 2 retains only moderate-consensus queries, our evaluation set could favor methods that reduce sensitivity to catalog order. To assess whether RPTUNE's benefits extend beyond these queries, we set aside 100 Beauty Bakerie candidates that received consensus scores ≥ 8 during Stage 2 (that have been returned to the generator for revision), and use them as a test set. We label each with its majority variant and evaluate both raw `gemma-4-E4B-it` and the full RPTUNE pipeline. As expected, these queries are considerably easier: raw Gemma reaches 33.9 EM and 72.0 FR, compared with 15.2 EM and 64.5 FR on Beauty Bakerie's main evaluation set (Table 3). RPTUNE nonetheless delivers large gains, reach-

ing 87.8 EM and 94.6 FR and eliminating roughly 80% of the baseline's remaining errors on both metrics (EM error falls from 66.1 to 12.2, and FR error from 28.0 to 5.4). The higher baseline performance on these high-consensus queries supports using consensus as a difficulty-control heuristic, and RPTUNE's large gains on them indicate that consensus filtering does not bias the evaluation set toward methods that reduce catalog-order sensitivity.

A.4 MODELS AND BASELINES

We evaluate RPTUNE across 9 LLM backbones from multiple model families, including `gemini-3.1-pro` (Google DeepMind, 2026a), `gemini-3.7-flash` (Google DeepMind, 2026b), `claude-sonnet-5` (Anthropic, 2026b), `claude-opus-5` (Anthropic, 2026a), `grok-4.1-fast-reasoning` and `grok-4.1-fast-non-reasoning` (xAI, 2025), `kimi-k2-thinking` (Moonshot AI, 2025), `glm-5.2` (Z.ai, 2026), and `gemma-4-E4B-it` (Google DeepMind, 2026c). For each backbone, we compare raw full-catalog prompting, with a randomly shuffled catalog in each trial, against RPTUNE-curated contexts retaining the top-ranked 25% of catalog variants. To further isolate curation quality, we compare the RPTUNE curator against three alternative context selectors: two dense retrieval models (EmbeddingGemma 300M (Gemma Team, 2025), upon which the RPTUNE encoder is trained, and Gemini Embedding 2 (Gemini Team, Google, 2026)) and a learned listwise ranker (Jina Reranker 3.5 (Nasika et al., 2026)). All baselines retain the top-ranked 25% of catalog variants to match RPTUNE's retention budget. These comparisons keep the downstream LLM weights frozen to isolate the effect of context curation. We evaluate RPTUNE with `gemma-4-E4B-it` post-trained on curated contexts.

We compare RPTUNE with retrieval, ranking, and context-compression baselines, using `gemini-3.7-flash` as the shared backbone for their LLM-based configurations. We evaluate Gemini Embedding 2 and Jina Reranker 3.5 within a RAG pipeline where the LLM selects from the top 10 variants. We additionally compare with RAG-Fusion (Rackauckas, 2024; Medrano et al., 2026), which uses Gemini Embedding 2 to retrieve candidates for multiple generated queries and combines their rankings using reciprocal rank fusion; TourRank (Chen et al., 2025), which aggregates results from repeated LLM-based tournaments to rank candidates; and LongLLMLingua (Jiang et al., 2024), which performs query-aware context compression.

We additionally compare GRPO with two alternative post-training objectives in Section 4.4: DPO (Rafailov et al., 2024) and IRPO (Wu et al., 2025), which extends preference optimization to ranking tasks. All three use the same synthetic catalog-grounded training data and backbone. We train GRPO for one epoch and DPO and IRPO for three epochs.

Setups. Finally, we detail our inference and training infrastructure. For proprietary models (Gemini, including Gemini Embedding 2; Claude; Grok) we call their respective APIs, while `kimi-k2-thinking` and `glm-5.2` are evaluated via their Model-as-a-Service (MaaS) endpoints on Google Cloud Vertex AI Model Garden. For custom and open-weights models, we utilize H100 GPUs: `gemma-4-E4B-it` is trained on two 80GB H100 GPUs using Unsloth (Han et al., 2023) and evaluated on a single 80GB H100 GPU. Similarly, we evaluate Jina Reranker 3.5, and train and evaluate the RPTUNE curator, on a single 80GB H100 GPU.

A.5 FEATURE REWARD DEFINITIONS

Let $\hat{v}$ be the variant selected by the model and $v^\star$ the ground-truth variant, with parent products $\hat{p}$ and $p^\star$. The feature reward (FR) is

$$\mathrm{FR}(\hat{v}, v^\star) = \begin{cases} 0 & \text{if } \hat{v} \notin \mathcal{V} \text{ (invalid ID, parse error, or failed call)}, \\ 1 & \text{if } \hat{v} = v^\star, \\ \dfrac{1}{Z} \sum_{f \in \mathcal{F}} w_f \, r_f(\hat{v}, v^\star) & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathcal{V}$ is the set of catalog variants and $Z = \sum_f w_f = 4.8$. The eight feature scores $r_f \in [0, 1]$ and their weights are:

Categorical attributes. Each is an indicator of equality: $r_{\text{type}} = \mathbb{1}[\hat{p}.\texttt{product_type} = p^\star.\texttt{product_type}]$ ($w = 0.5$); r_{ship}, r_{tax}, r_{avail} are the same indicator on the variant fields `requires_shipping`, `taxable`, and `available` ($w = 0.1$ each).

Numeric attributes. For $x \in \{\texttt{price}, \texttt{grams}\}$ ($w = 1.0$ each), with relative difference $\delta_x = |\hat{v}.x - v^\star.x|/v^\star.x$,

$$r_x = \begin{cases} 1 & \delta_x \le 0.1, \\ \max(0,\, 1 - \delta_x) & \text{otherwise}, \end{cases} \tag{3}$$

and if $v^\star.x = 0$, then $r_x = \mathbb{1}[\hat{v}.x = 0]$.

Tags. With tag sets $T(\cdot)$ ($w = 1.0$), $r_{\text{tag}} = |T(\hat{p}) \cap T(p^\star)| \,/\, |T(p^\star)|$, and if $T(p^\star) = \emptyset$, then $r_{\text{tag}} = \mathbb{1}[T(\hat{p}) = \emptyset]$.

Description. With $B(\cdot)$ the set of lower-cased word bigrams of `body_html` ($w = 1.0$), $r_{\text{body}} = |B(\hat{p}) \cap B(p^\star)| \,/\, |B(p^\star)|$. If the target has no bigrams, the same ratio is computed over unigrams; if it has no words at all, then $r_{\text{body}} = \mathbb{1}[\hat{p}$ has no words$]$.

All overlap scores are recall-oriented: they are normalized by the target's features, so the selected product is not penalized for having extra tags or text.

Example. On Beauty Bakerie, suppose the target is *SophistiCAKED Deluxe Powder Brush* ($14.00, 62 g) and the model selects *Milk & Honey Highlighting Brush* ($18.00, 30 g). Both are *Makeup Essentials* and match on shipping, taxable, and available, so the categorical terms contribute $0.5 + 0.1 + 0.1 + 0.1 = 0.8$. Price: $\delta = |18 - 14|/14 = 0.286 > 0.1$, so $r_{\text{price}} = 0.714$. Grams: $\delta = |30 - 62|/62 = 0.516$, so $r_{\text{grams}} = 0.484$. All 3 target tags (*back-in-stock*, *essentials*, *face product*) appear in the selected product, so $r_{\text{tag}} = 1$. Only 2 of the target's 43 description bigrams overlap, so $r_{\text{body}} = 0.047$. Hence

$$\text{FR} = \frac{0.8 + 0.714 + 0.484 + 1 + 0.047}{4.8} = \frac{3.045}{4.8} = 0.634,$$

whereas exact match (EM) gives 0 for this prediction.

A.6 Full Merchant-wise Results

Table 2 reports per-merchant exact match together with averaged EM, feature reward and latency. Table 3 gives the underlying breakdown: all three metrics for every merchant, in the same row order. Averaging a block across the seven merchants reproduces the corresponding Average column of Table 2. The only exception is the dashed row in the `gemma-4-E4B-it` block, the supervised-reorganizer ablation of Section 4.4 (Appendix A.10), which has no counterpart in Table 2.

A.7 Failure Analysis

The case study in Section 4.2 follows one query that RPTune answers correctly after post-training. This appendix takes the complement: of the queries RPTune still answers incorrectly, what does it return instead, and what separates that answer from the reference? We analyse the per-trial logs behind the + RPTune *[curator + post-train]* row of Table 2: 700 queries (7 merchants $\times$ 100), 10 trials each, recording the selected variant, the model's stated reason for selecting it, and its feature reward on every trial.

The answers returned instead are close substitutes. The mean feature reward of an error trial is 0.63, so a typical alternative already matches most of what the query asked for. Invalid or out-of-catalog identifiers account for 0.1% of error trials. The cases below are the recurring kinds, and each one turns on a distinction between candidates that the catalog record describes almost identically.

Variants of the same listing. 8.1% of error trials across all merchants, and 27.2% on PF, return the reference's parent listing under a different variant. PF sells most products in several pack sizes, which is the setting this arises in. Consider:

Table 3: Merchant-wise breakdown of Table 2: exact variant match (EM, %, ↑), feature reward (FR, %, ↑) and end-to-end latency (Lat., s, ↓) per merchant, in the same row order as Table 2. The EM columns are exactly the per-merchant values of Table 2; averaging a row across the seven merchants reproduces the corresponding Average column. The dashed row in the `gemma-4-E4B-it` block is an ablation without a counterpart in Table 2: the curator's reorganizer is trained with the supervised objective of Appendix A.10 instead of LLM-in-the-loop RL (Section 4.4).

Method / Model	BB			OON			HK			WCP			WB			FF			PF		
	EM	FR	Lat.	EM	FR	Lat.	EM	FR	Lat.	EM	FR	Lat.	EM	FR	Lat.	EM	FR	Lat.	EM	FR	Lat.
Main method comparison (LLM backbone: gemini-3.7-flash*)*																					
RAG-Fusion	18.8	62.3	6.0	0.1	25.7	6.4	1.8	39.3	7.0	15.0	55.3	5.8	11.3	55.4	6.5	2.0	55.5	6.7	16.0	46.1	6.7
LongLLMLingua	9.1	45.0	26.5	12.3	49.6	25.4	18.4	54.6	63.4	8.6	44.0	65.5	9.1	56.8	83.2	5.3	55.8	143.6	31.4	59.9	42.0
TourRank	6.1	41.2	58.2	13.0	50.8	55.0	16.7	54.8	60.6	9.0	50.5	58.2	10.1	58.3	59.1	14.3	65.2	54.4	26.2	61.5	51.4
RAG [EmbeddingGemma 300M]	28.8	66.9	5.2	23.8	61.0	6.1	17.0	49.4	6.8	16.7	59.4	5.8	26.1	63.9	4.7	7.3	68.5	5.5	45.7	71.3	5.8
RAG [Gemini Embedding 2]	37.3	73.9	3.6	18.4	51.4	3.2	10.9	47.5	3.8	32.3	63.7	3.5	31.8	68.8	3.0	12.9	59.0	3.1	44.6	71.5	3.3
RAG [Jina Reranker 3.5]	42.5	74.4	4.5	23.8	58.9	4.3	16.7	50.5	5.2	21.9	61.5	10.2	36.5	69.0	7.2	11.5	62.9	12.8	41.9	71.4	20.1
gemini-3.7-flash	38.6	69.9	5.6	44.0	69.9	4.2	30.7	62.2	11.2	30.3	64.1	7.2	37.4	72.9	10.6	30.4	74.6	5.9	46.3	69.6	11.5
Context selectors (25% catalog retention)																					
+ EmbeddingGemma 300M	34.5	70.5	5.3	39.3	68.2	5.8	25.5	57.1	9.3	20.9	59.4	7.7	44.9	75.1	7.4	26.0	73.6	7.2	45.7	71.1	6.7
+ Gemini Embedding 2	41.5	74.1	3.9	37.2	67.4	4.3	29.2	59.2	7.5	32.2	66.0	8.6	45.8	76.1	7.7	34.2	77.4	8.1	40.4	70.3	7.5
+ Jina Reranker 3.5	44.7	73.7	7.2	43.5	71.2	7.3	29.5	58.9	9.2	29.8	62.8	13.4	44.8	75.5	10.8	28.6	75.7	15.4	43.8	71.3	23.5
+ RPTUNE	51.2	78.0	3.7	56.3	78.3	3.1	51.1	73.7	7.8	42.4	71.7	5.3	60.0	82.1	5.8	46.6	83.7	4.9	52.2	73.8	5.1
RPTUNE *full pipeline evaluation*																					
gemma-4-E4B-it	15.2	64.5	4.3	7.5	49.9	4.1	2.5	43.4	4.2	6.6	55.7	6.2	7.8	55.6	6.2	11.1	68.9	7.1	24.3	58.8	8.4
+ RPTUNE [encoder]	25.2	67.9	3.6	9.6	51.9	4.1	10.3	55.5	3.7	12.5	57.0	3.9	14.8	57.1	3.7	13.2	70.3	3.7	28.9	64.6	3.9
+ RPTUNE [curator]	38.2	79.4	3.4	11.7	53.9	3.7	12.9	60.1	3.5	13.1	61.2	4.1	17.9	58.7	3.8	14.7	71.7	5.0	36.2	70.6	3.9
+ RPTUNE [curator + post-train]	49.3	86.4	3.1	25.1	70.0	3.3	19.2	64.0	3.6	22.8	71.7	4.2	36.2	72.9	3.8	21.3	79.6	3.8	43.3	76.0	4.0
+ curator w/ supervised reorganizer	32.5	75.0	3.0	6.7	52.1	3.1	6.0	43.6	3.2	8.3	55.2	3.7	16.3	59.3	3.8	12.9	69.9	3.9	31.8	66.6	4.3
RPTUNE *context curation across other frozen LLM backbones*																					
grok-4.1-fast-reasoning	40.6	72.4	36.7	44.0	70.6	34.6	25.7	60.7	40.6	18.1	54.1	31.0	29.8	69.7	27.7	21.6	72.5	38.2	28.1	57.5	27.7
+ RPTUNE	58.9	84.0	29.2	50.6	77.4	25.3	46.8	72.6	33.7	39.2	71.0	25.6	61.1	83.5	23.2	39.6	83.0	29.3	53.6	76.2	19.4
grok-4.1-fast-non-reasoning	14.8	58.8	8.0	21.6	54.7	8.0	12.6	52.4	8.0	18.3	52.4	9.7	11.8	45.1	10.0	3.1	59.6	10.5	7.7	29.2	11.9
+ RPTUNE	46.2	79.7	1.7	39.0	72.2	2.1	22.3	64.7	2.4	31.9	70.2	2.5	29.7	69.9	3.0	22.2	77.8	3.5	26.2	50.3	3.4
kimi-k2-thinking	40.1	72.6	69.4	39.4	70.3	65.7	21.9	55.2	73.8	22.4	59.0	65.7	34.6	69.7	75.2	23.3	74.6	83.5	32.0	59.5	61.5
+ RPTUNE	57.1	82.7	33.1	49.5	76.4	35.1	33.9	66.1	53.0	37.9	70.4	32.9	54.5	79.6	44.2	38.4	83.0	47.1	47.1	72.1	38.2
glm-5.2	32.6	56.2	120.3	28.8	45.3	129.9	15.8	34.6	151.1	31.2	62.7	109.0	42.4	73.1	131.4	33.1	77.1	122.1	47.0	72.6	95.2
+ RPTUNE	54.1	79.8	109.9	43.4	70.2	107.7	30.5	58.6	135.2	42.5	70.3	110.4	56.0	79.5	147.1	43.7	81.4	106.6	59.2	77.6	100.1
claude-sonnet-5	40.8	74.5	22.5	36.3	66.4	27.1	21.6	55.5	40.0	31.6	62.0	31.6	40.5	70.3	35.6	30.7	71.7	47.5	37.7	69.1	19.6
+ RPTUNE	54.1	80.7	21.0	43.5	72.3	22.4	34.1	64.2	33.9	36.0	65.7	28.4	52.4	77.6	31.6	39.8	82.3	33.5	54.2	78.4	20.4
gemini-3.1-pro	57.5	80.2	69.0	54.7	77.8	80.2	48.8	72.4	152.0	49.1	73.7	111.4	51.2	77.2	137.0	51.5	83.3	128.7	52.5	74.7	68.8
+ RPTUNE	69.9	87.3	47.1	70.8	87.7	44.8	68.7	84.3	91.6	55.7	77.9	95.8	52.3	78.4	127.0	55.5	85.6	115.9	65.0	82.0	50.7
claude-opus-5	54.0	79.6	19.1	43.8	71.8	20.3	30.0	57.9	24.3	31.0	61.7	20.8	44.4	73.1	25.4	34.1	78.4	24.6	53.7	77.9	21.7
+ RPTUNE	54.6	80.9	19.9	49.2	76.9	18.7	41.9	67.6	21.7	37.4	66.6	19.9	55.4	78.1	24.6	48.6	83.8	23.6	64.3	82.3	19.5

> **Q2.** "I'm looking for a completely plastic-free and PVOH-free laundry detergent for my newborn twins who have severe eczema. I strongly prefer the convenience of pre-measured drop-in pods [...] However, because we go through so many clothes, I absolutely need a package that provides at least 100 loads. My budget is strictly under $20."

For Q2, the reference is the 140-load pack of *The Simplest Baby Laundry Detergent* ($19.99); in 8 of 10 trials RPTUNE returns the 100-load pack of the same product ($16.99), with the stated reason that "the 100 Load variant meets the minimum load requirement and is under budget." Both packs satisfy both hard constraints, and the cue that favours the larger one, going through many clothes, is carried by a subordinate clause. The two packs are 0.09 FR apart, so the feature reward also treats them as close.

Sibling products that differ in one unscored field. The same structure recurs one level up, between products that are identical in every scored field. Consider:

> **Q3.** "I'm attending a super long outdoor wedding and need a completely transfer-proof, vegan liquid lipstick that won't budge when I eat cake or drink champagne. I'm looking for a striking warm red or a deep berry shade with a matte finish, though I'd consider a metallic one if it's just as smudge-proof. My budget is pretty strict at under 25 dollars [...]"

For Q3, RPTUNE returns the reference, *Cranberry Stiletto Matte Lip Whip*, in 6 of 10 trials, and in the other 4 returns *Versailles* or *Mon Chéri Matte Lip Whip*, which carry the same product type, the same $22.00 price and the same 23 g weight, and are separated from the reference only by shade name and description text ("a smudge-proof, vegan, matte liquid lipstick in a berry-sienna hue,

matching the user's request for a deep berry shade"). The case study in Section 4.2 follows this same discrimination on a numeric field, where post-training moves the model from 23 g substitutes to the 15 g reference; Q3 is the same query family with a colour word in place of the number.

A.8 CATALOG-INCREASE PROTOCOL

This appendix details the evaluation protocol behind Figure 3a, in particular how the evaluation set is composed at each injection level.

Catalog construction. The experiment uses a single merchant, Beauty Bakerie (BB, 96 products / 115 variants; Table 1). We form the reduced initial catalog by randomly removing half of the products *within each product type*, so that every product type survives in the reduced catalog and the type distribution is preserved. The full RPTUNE pipeline—encoder, curator, and post-training—is trained once on this reduced catalog and is never retrained thereafter. The withheld products are then reintroduced in ten cumulative steps: the set injected at level ℓ is a strict superset of the set injected at level $\ell - 10$, so a product that has returned never leaves again. At 0% the model sees only the reduced catalog; at 100% the entire BB catalog has been restored, i.e. the catalog has doubled relative to the catalog the model was trained on.

Evaluation set and reference labels. The same 100 evaluation queries, each run for 10 trials, are used at every injection level; no query is added or dropped as the catalog grows. Reference labels, however, are not constant. For the 56 queries whose original ground-truth product falls in the withheld half, no correct answer exists in the reduced catalog, so we re-establish a substitute reference by re-running the self-consistency filtering and human verification stages of our data pipeline (Section 4.1, Appendix A.3) against the reduced catalog. When such a query's original product is reintroduced, its reference label reverts to that product. A query is counted as *New-product* at a given level once its reference label has been replaced by a reintroduced product, and as *Original-product* while its reference label is still the one it carried at 0%. Group membership therefore shifts as products return, and Table 4 reports the resulting composition at every level.

Table 4: Composition of the BB evaluation set at each injection level. The query set is fixed at 100 queries $\times$ 10 trials throughout, so the counts below are also percentages of the evaluation set. *New-product* counts queries whose reference label has been replaced by a reintroduced product; *Original-product* counts the remainder.

Products reintroduced (%)	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
# queries, Total	100	100	100	100	100	100	100	100	100	100	100
# queries, Original-product	100	96	96	93	93	87	86	79	62	58	44
# queries, New-product	0	4	4	7	7	13	14	21	38	42	56

Reading the New-product panel. Because products are reintroduced gradually, the New-product group is very small at low injection levels: it contains 4 queries at 10% and 20% and 7 queries at 30% and 40%. The 100.0 EM values at 10% and 20% are therefore $4/4$, and the 57.1 at 30% is exactly $4/7$. These leftmost points should be read as indicative only; the New-product trend is meaningful from roughly 50% onward, where the group reaches 13 queries or more. The Total column is the one comparison whose denominator is constant across the sweep, and the relative-decline figures quoted in Section 4.3 are computed on it.

A.9 FEATURE-REWARD VIEWS OF THE ABLATIONS

Figures 9 and 10 give the feature reward (FR) view of the ablations in Section 4.4, over the same runs and aggregation as the main-text figures they mirror; FR tracks EM throughout. Tags in Figure 10 pair the post-training catalog with the evaluation catalog (U untrained, R raw, C curated), and its arrows give the FR gain from curated training at a fixed evaluation context.

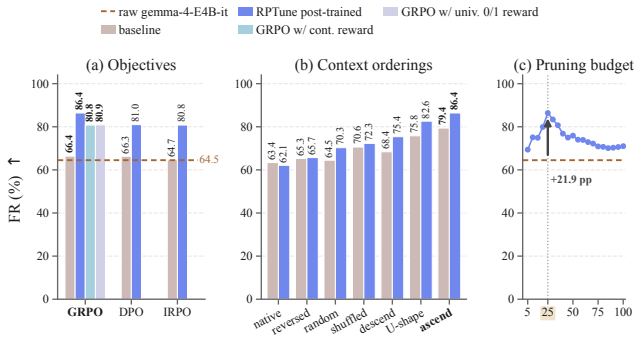


Figure 9: FR for (a) post-training objectives (Figure 4), (b) context layouts on Beauty Bakerie (Figure 7) and (c) pruning budget sweep (Figure 6).

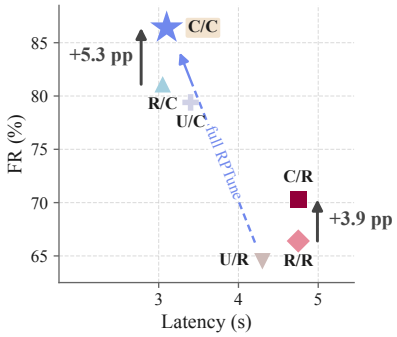


Figure 10: FR vs. latency for the curation quadrants (Figure 5).

A.10 Supervised Reorganizer Objective

This appendix details the supervised control used in the reorganizer ablation of Section 4.4. The ablation asks whether RPTune's LLM-in-the-loop objective (Eq. 1) is necessary, or whether ordinary supervised ranking with the same capacity and the same relevance supervision suffices.

What is held fixed. The supervised reorganizer differs from RPTune's reorganizer only in its training objective. It uses the same scoring function s_i (Section 3.1) on top of the same frozen encoder, updates only the MLP parameters θ, and starts from the same zero-initialized adjustment, so training begins from the encoder ranking. It is trained on the same synthetic queries and relevance scores $\mathcal{D}_{\text{rel}}$ (Section 3.2) and parameterizes the same catalog-wide softmax $w_\theta(c \mid q)$ of Eq. 1, here with $\beta = 1$. At inference time, the curated context is built identically (top 25%, ascending order) and passed to the same frozen `gemma-4-E4B-it`, evaluated under the protocol of Section 4.1.

Objective. We turn each query's graded relevance scores into a target distribution over the catalog and minimize its cross-entropy with w_θ:

$$p_T(c \mid q) = \frac{\exp\big(y(q,c)/T\big)}{\sum_{c' \in \mathcal{C}} \exp\big(y(q,c')/T\big)}, \qquad \mathcal{L}_{\text{Sup}}(\theta) = -\frac{1}{B} \sum_{q \in \mathcal{B}} \sum_{c \in \mathcal{C}} p_T(c \mid q) \log w_\theta(c \mid q). \tag{4}$$

We choose this listwise form so that $\mathcal{L}_{\text{Sup}}$ and $\mathcal{L}_{\text{Reorganizer}}$ are both cross-entropies against the same w_θ and differ only in their target; the ablation therefore isolates the training signal rather than the model or the loss family. The target temperature T controls how sharply p_T concentrates on the most relevant products: as $T \to 0$, all mass moves to $\arg\max_c y(q,c)$, and as $T \to \infty$, p_T becomes uniform. Since $y \in [0,1]$, $T = 0.1$ turns every 0.1 gap in relevance into an e-fold ratio in target mass, so the best matches dominate while near-substitutes retain partial credit.

Supervised versus LLM-in-the-loop gradients. The two objectives update the scores in different ways. For a query q,

$$\frac{\partial \mathcal{L}_{\text{Sup}}}{\partial s_i} \propto \frac{1}{\beta}\Big(w_\theta(c_i \mid q) - p_T(c_i \mid q)\Big), \qquad \frac{\partial \mathcal{L}_{\text{Reorganizer}}}{\partial s_i} \propto -\frac{1}{\beta G} \sum_{g \in \mathcal{V}_q} A_g \Big(\mathbb{1}[a_g = c_i] - w_\theta(c_i \mid q)\Big).$$

The supervised gradient is dense and fixed: every product moves toward its label mass at every step, independently of what the downstream LLM does with the resulting context. In the LLM-in-the-loop gradient, when all G rollouts are valid, the standardized advantages sum to zero, the w_θ terms cancel, and only products that the frozen LLM actually selected from the current context receive gradient: selections that beat their group average are promoted and the others are demoted. The RL signal thus concentrates on the LLM's observed behavior on the contexts the reorganizer currently builds, for example demoting a plausible but less relevant product that the LLM tends to select, whereas the supervised target does not depend on the LLM at all.

Training details. We minimize $\mathcal{L}_{\text{Sup}}$ with AdamW (learning rate 10^{-3}, linear warmup over the first 10% of steps followed by linear decay) on minibatches of $B = 128$ queries, as in RL training

(Appendix A.2), for 20 epochs over each merchant's training queries. We use $T = 0.1$ and the final (epoch-20) checkpoint for every merchant; both were fixed in advance rather than tuned on the evaluation queries. Supervised training requires no LLM calls and is therefore considerably cheaper than RL training.

Results. The dashed row of the `gemma-4-E4B-it` block in Table 3 reports the per-merchant results. Averaged over the seven merchants, the supervised reorganizer reaches 16.4% EM and 60.2% FR, essentially matching the encoder-only curator (16.4% EM, 60.6% FR) and falling 4.3 EM and 4.9 FR points short of RPTUNE's LLM-in-the-loop reorganizer (20.7% EM, 65.1% FR). It improves over the encoder on only three merchants (BB, WB, and PF) and does not improve on the other four (OON, HK, WCP, and FF), whereas RPTUNE's reorganizer improves over the encoder on all seven merchants and achieves higher EM than the supervised reorganizer on each of them. Relevance supervision alone therefore does not yield consistently better contexts; the gains of RPTUNE's reorganizer come from optimizing against the downstream LLM's selections.