EVOLVING SUPERVISION THROUGH LEARNING FEEDBACK FOR GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation models commonly learn target-item distributions through next-token prediction (NTP) on historical interactions. Yet observed interactions are incomplete proxies for users' latent preferences, rather than ground-truth descriptions of their preferences. Supervision can be enriched by extending click-based targets to purchases or incorporating behavioral signals from other contexts. The usefulness of additional supervision depends both on what information it adds and on what the current model has already learned. This motivates us to jointly consider supervision construction and feedback-guided learning. We propose a feedback-driven supervision evolution framework with two nested cycles. The outer Supervision Evolution Cycle (SEC) uses large language models (LLMs) to analyze updated performance feedback, diagnose supervision needs, and guide subsequent supervision construction. Within each round, the inner Feedback-Guided Learning Cycle (FGLC) updates the policy for incorporating constructed supervision based on its observed learning effects. We apply the framework to expand supervision in a recommendation retrieval system. Multi-round offline experiments on industrial recommendation retrieval show that the framework improves retrieval performance under matched total added supervision. Online A/B testing also shows improvements in ad clicks and revenue. The system has been fully deployed on a large-scale industrial e-commerce platform serving hundreds of millions of active users.

1 INTRODUCTION

Generative recommendation models formulate item retrieval as sequence generation, learning to generate target-item identifiers conditioned on users' behavioral histories (Rajput et al., 2023; Deng et al., 2025; Zheng et al., 2024). These models are commonly trained with next-token prediction (NTP), where each behavioral context is paired with an observed interaction as its target (Wang et al., 2024). Observed interactions, however, provide only a partial account of users' interests: exposure and feedback shape which items users encounter and interact with (Schnabel et al., 2016; Joachims et al., 2017; Saito et al., 2020). Interests that are not expressed in recorded interactions receive no direct target supervision under NTP. This motivates exploring supervision beyond the observed interactions.

The usefulness of additional supervision depends on what information it adds and on what the model has already learned. New targets may reinforce preference relationships the model already captures while providing little evidence about those it still misses. We use current model performance and existing supervision coverage to guide where to construct further targets. Feedback-driven construction has been explored in LLM training: S3 and LLM2LLM generate additional examples from a student's errors (Wang et al., 2023; Lee et al., 2024), and data-generation agents in DataEnvGym plan generation from student feedback (Khan et al., 2025). After the model learns from the added targets, renewed evaluation shows which difficulties remain and informs the next round of construction.

Model feedback also bears on how the constructed supervision is learned. New and existing targets train the same model, so learning from one group changes the starting point for learning from others. A training plan fixed in advance need not stay matched to the model as learning proceeds. Teacher–Student Curriculum Learning adjusts task sampling using estimated learning

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.

progress (Matiisen et al., 2019), and Aioli adjusts data-mixture proportions by re-estimating how training on each group affects validation losses (Chen et al., 2025). These methods adapt learning over given task families or data groups. When the supervision is itself expanding, learning outcomes also indicate whether the added targets addressed the model's difficulties, which informs the next round of construction. We therefore use model feedback to guide both what supervision to construct and how to learn from it.

We organize supervision construction and learning into a feedback-driven supervision evolution framework with two nested cycles. In the outer Supervision Evolution Cycle (SEC), LLMs decide where to expand supervision, using current performance, changes across rounds, and existing supervision coverage over task-specific diagnosis units. The constructed batches enter the inner Feedback-Guided Learning Cycle (FGLC), which uses short training probes to assess their effects on model performance and adapt the learning policy as the model changes. After all batches have been incorporated, updated performance and supervision coverage inform the next SEC diagnosis.

We instantiate the framework in an industrial e-commerce generative retrieval system, using product categories as diagnosis units to reflect differences in how users' needs translate into recorded behavior. An LLM-based constructor infers possible purchase motivations, matches them to users' behavioral histories, and generates intent-guided target-product profiles. Catalog items matching these profiles yield additional targets for the same behavioral contexts. We evaluate the complete loop through multi-round offline experiments in industrial recommendation retrieval. With matched total added supervision, the framework achieves better retrieval performance than a fixed initial-diagnosis plan (§5.2). Controlled comparisons further support the effectiveness of feedback-guided progressive learning (§5.4). Online A/B testing also shows improvements in ad clicks and revenue (§5.5). The system has been fully deployed on a large-scale industrial e-commerce platform.

Our main contributions are as follows:

- We identify a key limitation of supervision expansion in generative recommendation: adding preference evidence alone does not determine its usefulness, which also depends on what the model has already learned.
- We propose a feedback-driven supervision evolution framework that jointly addresses what supervision to construct and how to learn from it. The outer SEC guides successive construction using updated model feedback, while the inner FGLC adapts the learning policy for constructed supervision according to its observed effects (§4).
- We validate the framework in industrial recommendation retrieval through multi-round offline experiments, and online A/B testing, demonstrating improvements in retrieval performance and business outcomes, with full deployment on a large-scale industrial e-commerce platform serving hundreds of millions of active users (§5).

2 Related Work

Supervision expansion in recommendation. LLMRec augments interaction supervision with LLM-selected positive and negative items, alongside semantic attributes (Wei et al., 2024). DiffuASR generates pseudo item sequences and prepends them to observed histories for sequential-recommender training (Liu et al., 2023). Intent-based generation also provides training signals: Think-to-Personalize constructs teacher-generated intent-enhanced queries for joint generation and retrieval training (Jiang et al., 2026), while RecGPT iteratively trains intent-related modules on self-generated samples filtered by human–LLM judging (Yi et al., 2025). Our recommendation instantiation constructs additional context–item targets from inferred intents, focusing on how learning feedback guides both subsequent supervision construction and the learning of constructed targets.

Feedback-driven supervision construction. S3 and LLM2LLM use student errors on validation and seed training examples, respectively, to guide iterative data synthesis (Wang et al., 2023; Lee et al., 2024). DataEnvGym provides teacher environments in which data-generation plans are updated from student evaluation feedback, including skill-level performance (Khan et al., 2025). Montessori-Instruct measures local data influence through reference-loss changes after one-step student updates, then uses the resulting preferences to optimize the teacher with DPO for subsequent

synthesis (Li et al., 2025). Our framework couples feedback-driven supervision construction across rounds with adaptive learning within each round.

Adaptive curricula and data mixing. Learning-progress curricula adapt task sampling using changes in student performance (Graves et al., 2017; Matiisen et al., 2019). Skill-It adjusts training mixtures using validation losses and a learned graph of cross-skill influence (Chen et al., 2023), while Aioli re-estimates cross-group interactions during training to update mixture proportions (Chen et al., 2025). GLISTER instead selects training subsets according to a validation objective, refreshing selection as model parameters change (Killamsetty et al., 2021a). Our FGLC uses learning feedback to adapt the order of all newly constructed batches, rather than select a final subset.

3 PRELIMINARIES

Let π_θ be a generative retriever, x a user's behavioral context, and $y = (y_1, \ldots, y_{|y|})$ a target item-identifier sequence. Historical context–target pairs form the initial supervision $\mathcal{D}_0$. For a training set $\mathcal{D}$, the retriever is trained with the next-token prediction (NTP) objective:

$$\ell_{\mathrm{NTP}}(\theta; \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \sum_{j=1}^{|y|} \log \pi_\theta(y_j \mid x, y_{<j}). \tag{1}$$

4 FEEDBACK-DRIVEN SUPERVISION EVOLUTION

We first formulate the interaction between supervision construction and adaptive learning through two nested cycles: the Supervision Evolution Cycle (SEC) and its embedded Feedback-Guided Learning Cycle (FGLC), as illustrated in Figure 1. We then describe its initial implementation in recommendation retrieval, where intent mining serves as the supervision-construction component.

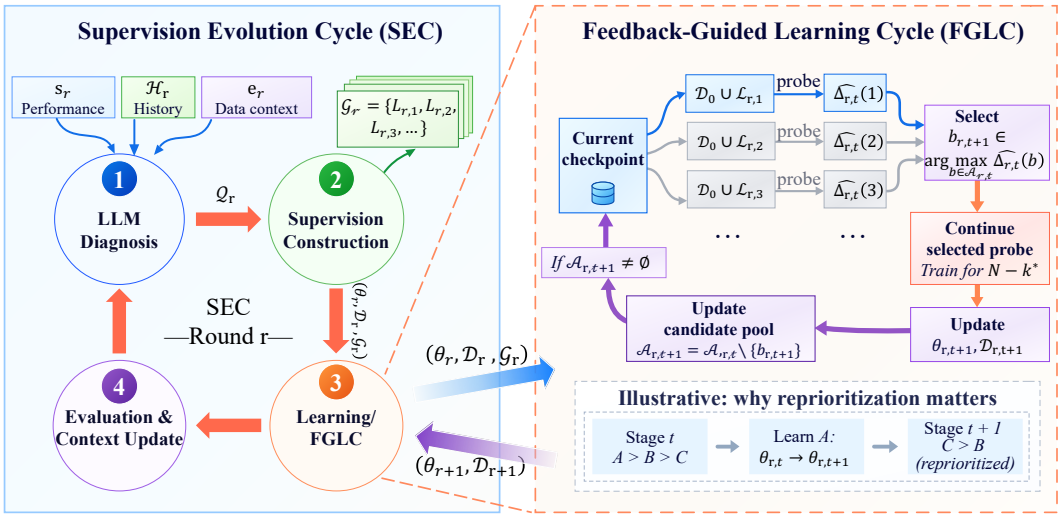

Figure 1: Overview of feedback-driven supervision evolution. The SEC diagnoses supervision needs and issues requests to a supervision constructor. The FGLC learns the resulting batches through probe-guided continuation. Updated performance and context inform the next SEC diagnosis.

4.1 SUPERVISION EVOLUTION CYCLE

The SEC diagnoses supervision needs from model feedback and issues requests to a separate supervision-construction component. At SEC round r, the system maintains a model θ_r and a cumulative record $\mathcal{D}_r$ of supervision learned so far. Diagnosis uses three kinds of information: the current system state s_r, a history of state changes across rounds $\mathcal{H}_r$, and environment information e_r. The

state summarizes performance along dimensions relevant to the task and supervision expansion. The history records how this state has changed following earlier updates, together with the corresponding expansion decisions. Environment information describes the training-data distribution, its composition, and the data available for further expansion.

The LLM diagnoses the current supervision needs and produces construction requests for the round:

$$\mathcal{Q}_r = \mathrm{Diagnose}_{\mathrm{LLM}}(s_r, \mathcal{H}_r, e_r) = \{q_{r,b} : b \in \mathcal{P}_r\}. \tag{2}$$

Here $\mathcal{P}_r$ indexes the round's construction requests, and each $q_{r,b}$ specifies the supervision to be produced by the constructor. The diagnosis prompt (Appendix A) guides the LLM to produce executable requests with concise justifications. This state-dependent view follows iterative machine teaching (Liu et al., 2017) and feedback-driven data construction (Wang et al., 2023; Lee et al., 2024; Khan et al., 2025).

The supervision-construction component receives each request together with the available data context e_r and returns a batch:

$$\mathcal{L}_{r,b} = \mathrm{Construct}(q_{r,b}, e_r), \qquad \mathcal{G}_r = \{\mathcal{L}_{r,b} : b \in \mathcal{P}_r\}. \tag{3}$$

Each $\mathcal{L}_{r,b}$ contains context–target pairs, and $\mathcal{G}_r$ is passed to the FGLC. Construction requests are indexed within each round, so a later round may request further supervision for a previously expanded target. The requests and available source data determine the content and volume of expansion.

The FGLC progressively incorporates all batches in $\mathcal{G}_r$, adapting its learning policy according to observed learning effects. Upon completion, the SEC obtains the updated model, supervision, and system state:

$$(\theta_{r+1}, \mathcal{D}_{r+1}) = \mathrm{Learn}(\theta_r, \mathcal{D}_r, \mathcal{G}_r), \tag{4}$$

$$s_{r+1} = \mathrm{Evaluate}(\theta_{r+1}). \tag{5}$$

The latest state change is added to the history:

$$\delta_{r+1} = \mathrm{Compare}(s_r, s_{r+1}), \qquad \mathcal{H}_{r+1} = \mathcal{H}_r \parallel (\mathcal{Q}_r, \delta_{r+1}), \tag{6}$$

where $\mathrm{Compare}$ summarizes changes in the task-specific state and $\parallel$ appends a record, starting from an empty $\mathcal{H}_0$. Environment information is also refreshed to e_{r+1} to reflect the updated training data and available sources.

4.2 Feedback-Guided Learning Cycle

Given the constructed batches $\mathcal{G}_r$ for round r, the FGLC adapts their learning policy under the NTP objective (Eq. 1), determining the order of all batches rather than selecting a final subset. Drawing on learning-progress curricula (Graves et al., 2017; Matiisen et al., 2019) and objective-driven data selection (Killamsetty et al., 2021a;b; Xia et al., 2024), we evaluate candidate continuations from the current model state and refresh their scores after each committed update.

We use t for FGLC stages, with $\theta_{r,0} = \theta_r$, $\mathcal{D}_{r,0} = \mathcal{D}_r$, and $S_{r,0} = \varnothing$. The set $S_{r,t}$ records only the batch indices incorporated during SEC round r and is initialized anew in each round; the remaining indices are $\mathcal{A}_{r,t} = \mathcal{P}_r \setminus S_{r,t}$. The sets $\mathcal{D}_r$ and $\mathcal{D}_{r,t}$ record supervision coverage for subsequent diagnosis. Each stage trains on the historical supervision $\mathcal{D}_0$ together with the batch selected for that stage; completed additional batches leave the stage-training data, while the updated model carries their learning effects forward. Let $U(\theta)$ be a higher-is-better utility on a fixed decision benchmark, measuring the model's overall task performance. For a candidate $b \in \mathcal{A}_{r,t}$, its full-stage continuation and learning value are

$$\theta_{r,t+1|b} = \mathrm{Train}_N(\theta_{r,t}, \mathcal{D}_0 \cup \mathcal{L}_{r,b}), \tag{7}$$

$$\Delta_{r,t}(b) = U(\theta_{r,t+1|b}) - U(\theta_{r,t}). \tag{8}$$

Here Train_k denotes k NTP updates under the common stage-training protocol (Appendix B.2). Candidate continuations share the current model as their reference and the stage budget N, so their learning effects are compared using the same overall utility.

Completing every candidate continuation is expensive. We instead run a short probe of $k^* < N$ steps from the common stage checkpoint for each remaining batch, following the prefix of its full-stage

training schedule:

$$\widetilde{\theta}^{(k^*)}_{r,t,b} = \mathrm{Train}_{k^*}(\theta_{r,t}, \mathcal{D}_0 \cup \mathcal{L}_{r,b}), \tag{9}$$

$$\widehat{\Delta}_{r,t}(b) = U(\widetilde{\theta}^{(k^*)}_{r,t,b}) - U(\theta_{r,t}). \tag{10}$$

The stagewise decision selects the next batch in the incorporation order according to its estimated learning value:

$$b_{r,t+1} \in \arg\max_{b \in \mathcal{A}_{r,t}} \widehat{\Delta}_{r,t}(b). \tag{11}$$

Completing the selected stage updates the learned-batch indices and cumulative supervision record:

$$S_{r,t+1} = S_{r,t} \cup \{b_{r,t+1}\}, \qquad \mathcal{D}_{r,t+1} = \mathcal{D}_{r,t} \cup \mathcal{L}_{r,b_{r,t+1}}. \tag{12}$$

We retain the selected probe and continue it for the remaining $N - k^*$ steps, obtaining $\theta_{r,t+1} = \theta_{r,t+1|b_{r,t+1}}$. The next stage reevaluates the remaining batches from this updated model, pairing each candidate with the historical supervision. The FGLC continues until all batches in $\mathcal{P}_r$ have been incorporated, giving $T_r = |\mathcal{P}_r|$ and $S_{r,T_r} = \mathcal{P}_r$. The SEC then receives $\theta_{r+1} = \theta_{r,T_r}$ and $\mathcal{D}_{r+1} = \mathcal{D}_{r,T_r}$.

Short probes provide a ranking proxy; Appendix C.1 reports an independent ranking calibration and local full-stage validation.

4.3 Intent-Based Supervision Expansion

We apply the framework to recommendation retrieval, using product categories as the diagnosis dimensions and intent mining as the supervision-construction component.

Category-level diagnosis. The state s_r summarizes category-level performance, and the history $\mathcal{H}_r$ records its changes across rounds alongside earlier expansion decisions. Environment information e_r includes the amount and composition of existing supervision, prior category-level expansions, the available behavioral data, and existing category-specific purchase-logic libraries. Together, these inputs allow the SEC to identify categories for further supervision expansion. Each construction request specifies a target category, $q_{r,b} = c_{r,b}$.

Intent-based supervision construction. The intent-mining component executes each request by building or revising the purchase-logic library for category $c_{r,b}$ and using it to construct additional context–target pairs from the current data represented in e_r. When a category c is expanded for the first time, general-purpose LLMs analyze user behavioral data to infer possible purchase motivations; similar inferred logics are deduplicated and merged into an initial purchase-logic library. In the deployed system, learning from the resulting supervision changes the model, which in turn changes the data produced by the system. If the SEC diagnoses the same category as requiring further expansion, the constructor uses the currently available data to revise the existing library. Denote the newly constructed or revised library used for category c in round r by $\mathcal{B}_{r,c}$.

Following library construction or revision, the constructor selects behavioral contexts from the current source data for the requested category. Given a selected context x, an in-house LLM trained for the industrial application matches the history to the most compatible purchase logic in $\mathcal{B}_{r,c}$. The matched purchase logic guides the generation of a target-product profile, which is matched to catalog products to obtain candidate items, each identified by a SKU (product ID). Each candidate's identifier sequence y is paired with the original context x to form an additional context–item pair (x, y) for training under the same NTP objective. This pipeline implements $\mathrm{Construct}(q_{r,b}, e_r)$ and yields batch $\mathcal{L}_{r,b}$ in Eq. 3.

Learning feedback and deployment. The FGLC uses macro-averaged retrieval recall on the decision set as its feedback utility U. Let $R_c(\theta)$ denote mean recall@1500 for evaluation category c, expressed on a 0–100 percentage scale, and let $\mathcal{C}_{\mathrm{eval}}$ contain all evaluation categories:

$$U(\theta) = \frac{1}{|\mathcal{C}_{\mathrm{eval}}|} \sum_{c \in \mathcal{C}_{\mathrm{eval}}} R_c(\theta). \tag{13}$$

Learning from one category can affect others through the shared retriever, so this utility evaluates overall effects rather than only the added category's gain.

Offline SEC diagnosis uses category-level retrieval recall. In the deployed system, an initial offline diagnosis identifies the first categories for supervision expansion; subsequent SEC diagnoses use online business feedback, including ad clicks and revenue. The FGLC retains the retrieval utility above in both settings. Final offline performance is measured on a separate test set, with the evaluation protocol specified in Appendix B.

5 EXPERIMENTS

We evaluate the complete loop through multi-round retrieval experiments. The SEC studies compare constructed supervision at fixed learner states, while the FGLC study compares learning policies on identical supervision. Online A/B testing evaluates business impact.

5.1 EXPERIMENTAL SETUP

Dataset. We evaluate on a proprietary industrial dataset from an industrial e-commerce recommendation system. Evaluating supervision evolution in a live system requires both behavioral logs for constructing supervision and online feedback on successive model updates. Historical supervision comprises approximately 40 million context–target pairs. All offline methods use behavioral logs from the same fixed 10-day window for supervision construction throughout all rounds. The decision set is drawn from the calendar day immediately following this 10-day window, and the test set from the next calendar day.

Backbone. We use Qwen3-0.6B (Team, 2025) as the generative retrieval backbone, trained with the NTP objective. The 0.6B model size is fixed across all methods to satisfy the serving cost and latency budgets of the production system.

LLM configuration. We use DeepSeek-V4-Flash (DeepSeek-AI, 2026) for SEC diagnosis. Qwen3.5-397B-A17B (Qwen Team, 2026) constructs and revises purchase-logic libraries, while an in-house model fine-tuned from Qwen3.5-9B matches behavioral contexts to library entries. These open-weight models support self-hosted deployment.

Evaluation metrics. Our primary offline metric is category-macro recall@1500 over approximately 300 product categories, including categories not expanded in a given round (Eq. 13). Decision-set performance guides SEC diagnosis and FGLC probe scoring; U_{test} denotes the same metric on a separate, fixed test set. We report recall on a 0–100 percentage scale and gains in percentage points (pp). Controlled comparisons report gains from their shared starting checkpoints and paired differences between strategies. The main offline experiments use three independent random seeds. We report means with standard deviations ($\pm$) where available; entries without $\pm$ report means only. The online A/B evaluation reports relative changes in ad clicks and revenue (§5.5).

5.2 OVERALL EFFECTIVENESS OF SUPERVISION EVOLUTION

Compared strategies. We cross two construction-request policies with two learning policies, using the same intent-based construction procedure across all four combinations. All four arms share the first-round category requests, with only the learning policy varying in that round. **Fixed requests** uses the initial state and environment to formulate a category-expansion schedule shared by arms A and C before training. **Updated requests** uses each arm's current state, history, and environment to update both target categories and request counts from the second SEC round onward. The fixed plan is determined independently of the updated policy's eventual category choices. This factor changes whether request decisions are refreshed; either policy may revisit a category.

For learning, **Initial-order** probes every batch at the start of each round, fixes the batch order using these scores, and learns all batches while carrying the updated model forward between batches. **Feedback** reevaluates the remaining batches after each stage and learns all batches using the FGLC. Both score probes by overall decision utility. The four combinations are listed in Table 1; arm D is the complete framework.

Table 1: Multi-round strategies and final test utility.

Arm	Requests	Learning	U_{test} (%)
A	Fixed	Initial-order	72.41 ± 0.05
B	Updated	Initial-order	74.80 ± 0.09
C	Fixed	Feedback	74.22 ± 0.06
D (full)	Updated	Feedback	76.73 ± 0.08

Table 2: Test utility after each completed evolution round (%).

Completed rounds	A	B	C	D (full)
0	70.26 ± 0.05	70.26 ± 0.05	70.26 ± 0.05	70.26 ± 0.05
1	70.47 ± 0.03	70.47 ± 0.03	70.78 ± 0.02	70.78 ± 0.02
2	70.83 ± 0.04	71.18 ± 0.05	71.07 ± 0.03	71.96 ± 0.06
3	71.51 ± 0.05	71.98 ± 0.04	71.76 ± 0.06	73.37 ± 0.07
4	71.73 ± 0.04	73.28 ± 0.07	72.45 ± 0.05	74.53 ± 0.08
5	72.22 ± 0.06	74.23 ± 0.08	73.39 ± 0.07	75.68 ± 0.05
6	72.41 ± 0.05	74.80 ± 0.09	74.22 ± 0.06	76.73 ± 0.08

Experimental protocol. We run $R = 6$ offline rounds, indexed by $r = 0, \ldots, R-1$, from a common initial checkpoint within each independent run. In round r, each arm requests supervision for m_r distinct categories, with $n_r = 100{,}000$ new pairs per request and $N_r = 4000$ committed updates per batch. Total added supervision is matched across arms at the end of all R rounds; per-round request counts and target categories may differ. An arm requesting an already constructed expansion of a category reuses the corresponding batch. We report final test utility and the trajectory after each completed round. Appendix B details the shared construction and training protocol; Appendix C.2 reports computation accounting, including discarded probes.

Updating construction requests improves final utility under both learning policies (B over A and D over C). Under the shared fixed construction schedule, C exceeds A by 1.81 pp on identical supervision batches, supporting the benefit of refreshing learning decisions as the model evolves. With updated requests, D exceeds B by 1.93 pp, with subsequent construction requests responding to each arm's evolving model state. The complete framework achieves the highest final utility, exceeding C by 2.51 pp.

Table 2 follows test utility from the shared initial checkpoint through six completed rounds; its final row gives the endpoints reported in Table 1. Arms C and D tie for the highest first-round utility. The complete framework achieves the highest utility from round two onward and continues to improve through round six. Appendix B.3 reports arm D's first-time and revisited category counts and the training pairs added at each round.

5.3 Constructing Supervision from Updated Feedback

Common-state construction fork. We use arm A's checkpoint after its first completed round as the common starting state, with $U_{\text{test}} = 70.47\%$ (Table 2). We clone this model and optimizer checkpoint together with its diagnosis history, supervision coverage, source data, and purchase-logic libraries. Three request policies then branch from this state to construct supervision for the second round. **Original-plan** follows the second-round requests in the initial schedule. **Low-recall** selects eligible categories with the lowest current decision-set recall. **Updated-diagnosis** issues requests from the current $(s_r, \mathcal{H}_r, e_r)$ using the SEC. The policies share the eligible category universe, request count, pair quota, and constructor. Each branch obtains supervision for its requests using the same available-data snapshot and constructor, including library construction or revision when required.

To compare the usefulness of the resulting supervision without adaptive learning-order differences, each branch performs one Joint continuation from the shared checkpoint, training on historical supervision plus all batches obtained for its second-round requests under the same committed-update

Table 3: Second-round construction from a common checkpoint.

Request policy	Test gain (pp)	Gain over Original-plan (pp)
Original-plan	0.39 ± 0.02	0.00 ± 0.00
Low-recall	0.55 ± 0.03	0.16 ± 0.04
Updated-diagnosis	1.47 ± 0.07	1.08 ± 0.08

Table 4: Learning gains from fixed supervision at two model states (pp).

Constructed supervision	Earlier state	Later state
Original-plan	0.14 ± 0.01	0.19 ± 0.02
Updated-diagnosis	0.25 ± 0.02	0.48 ± 0.04
Updated-diagnosis minus Original-plan	0.11 ± 0.02	0.29 ± 0.04

budget. The outcome is the gain in overall test utility after this continuation (Table 3). Updated-diagnosis versus Original-plan tests renewed construction decisions beyond simply adding new examples and training further. Low-recall supplies a dynamic reference based on current performance alone.

With the starting state and learning protocol fixed, Updated-diagnosis yields greater test gains than both alternatives, exceeding Low-recall by 0.92 pp. The comparison supports using updated feedback together with supervision context to guide construction requests, rather than targeting low-recall categories alone.

Dependence on the learner state. We use arm A's checkpoints after rounds one and five as the earlier and later states, with $U_{\text{test}} = 70.47\%$ and 72.22%, respectively (Table 2). We freeze the Original-plan and Updated-diagnosis supervision sets constructed for round six and train each set from both checkpoints. The source data and constructed pairs remain unchanged across these continuations, and all branches use the same joint-incorporation protocol and update budget. Table 4 reports test gains relative to each starting state. This retrospective comparison examines whether the relative advantage of updated construction changes with the learner state while holding the two supervision sets fixed.

Updated-diagnosis yields greater gains at both checkpoints, with a larger advantage at the later state. Since each supervision set is unchanged across states, this pattern is consistent with the relative usefulness of constructed supervision depending on the learner's current state.

5.4 Learning Constructed Supervision with Updated Feedback

Within each seed, we use arm A's initial checkpoint, with zero completed evolution rounds (Table 2), as the common starting state. We collect and freeze all supervision batches incorporated along arm A's trajectory from zero to six completed rounds. Three learning policies then start from the same initial model and optimizer state with this complete collection. **Joint** trains on historical supervision and all collected batches together. **Initial-order** probes every batch in the collection at the initial checkpoint and learns them in the resulting fixed order. **Feedback** refreshes probes over the remaining batches after each stage and selects the next batch using the FGLC. Both progressive policies score probes by overall decision utility. All three learn the entire collection under the same total committed-update budget; the progressive policies additionally share per-batch update budgets, historical-data mixing, and stage-training settings.

Table 5 reports final test utility and gains computed relative to the corresponding initial checkpoint within each seed before aggregation across seeds.

Initial-order modestly improves over Joint training, whereas refreshing the learning order in Feedback yields a larger additional gain. With all policies learning the same supervision, these results support adapting learning decisions to the model's evolving state.

Table 5: Learning all supervision from arm A's six-round trajectory.

Learning policy	Final U_{test} (%)	Gain from start (pp)
Joint	72.03 ± 0.13	1.77 ± 0.08
Initial-order	72.66 ± 0.16	2.40 ± 0.11
Feedback	74.36 ± 0.19	4.10 ± 0.14

Table 6: Online supervision evolution.

Round	Expansion categories	New training pairs (M)	Clicks lift (%)	Revenue lift (%)
1	5	0.50	+0.12	+0.21
2	13	1.30	+0.32	+0.45
3	8	0.80	+0.44	+0.62
4	9	0.90	+0.59	+0.78
5	10	1.00	+0.70	+0.92

Appendix C reports independent probe-ranking evidence and specifies computation accounting for the learning policies.

5.5 Online Supervision Evolution and Business Impact

The first online round expands supervision for five categories identified through offline evaluation. For rounds two through five, the LLM uses category-level online clicks and revenue, together with their changes after model updates, to guide further supervision expansion.

Table 6 reports five online evolution rounds covering 45 distinct categories and adding 4.5 million training pairs. All category expansions in this trajectory are first-time additions, and new training-pair counts denote additions within each round. Each row reports online performance observed after completing that round's SEC, including its embedded FGLC. The initial system before the first round is retained as the concurrent A/B control throughout the experiment; all click and revenue lifts are relative to this control. Observed click and revenue lifts increase across the five online rounds, reaching $+0.70\%$ and $+0.92\%$, respectively.

6 Limitations

Our current application focuses on intent-based supervision expansion, where we observe performance gains. Future work should extend the framework to other dimensions of supervision expansion to evaluate its effectiveness in those settings.

7 Conclusion

In generative recommendation, the usefulness of added supervision depends on both the preference evidence it provides and what the model has already learned. We presented a feedback-driven supervision evolution framework that jointly adapts what supervision to construct and how to learn from it. Offline experiments show improved retrieval across multiple rounds, and online A/B testing shows gains in ad clicks and revenue. These findings support the view that supervision should evolve with the model's learning state.

AI use statement

In this work, we used LLMs as components of the proposed framework to analyze model feedback and generate additional training supervision. We also used AI assistance for language polishing, improving readability, and enhancing grammatical correctness. The research ideas, method design,

and experimental protocol were developed by the authors. The authors have reviewed all AI-assisted content and take responsibility for the final text, claims, and associated artifacts.

Ethics statement

This work evaluates recommendation retrieval-model training through offline experiments and an online A/B test. We report aggregate results without user-level identifiers or external data release. The supervision-selection objective is retrieval recall; the online evaluation also measures advertising outcomes.

Reproducibility statement

The training objective and supervision-evolution procedure are specified in §3 and §4. Appendix A provides the SEC diagnosis prompt. Appendix B describes the shared evaluation and training conditions and the controlled-comparison protocols. Probe-ranking evidence and computation accounting are described in Appendix C. Section 5.5 describes the online evaluation and its comparison reference.

References

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36:36000–36040, 2023.

Mayee Chen, Michael Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. In *International Conference on Learning Representations*, volume 2025, pp. 46089–46132, 2025.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

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

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. Pmlr, 2017.

Angqing Jiang, Gaoming Zhang, Jianchun Song, Kena Qi, Dayao Chen, Wei Lin, and Defu Lian. Think-to-personalize: Unifying reasoning and retrieval for user-centric personalized dense retrieval. *arXiv preprint arXiv:2608.18855*, 2026.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 781–789, 2017.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. Dataenvgym: Data generation agents in teacher environments with student feedback. In *International Conference on Learning Representations*, volume 2025, pp. 53480–53507, 2025.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8110–8118, 2021a.

Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in neural information processing systems*, 34:14488–14501, 2021b.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6498–6526, 2024.

Xiaochuan Li, Zichun Yu, and Chenyan Xiong. Montessori-instruct: Generate influential training data tailored for student learning. In *International Conference on Learning Representations*, volume 2025, pp. 74809–74834, 2025.

Qidong Liu, Fan Yan, Xiangyu Zhao, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Feng Tian. Diffusion augmentation for sequential recommendation. In *Proceedings of the 32nd ACM International conference on information and knowledge management*, pp. 1576–1586, 2023.

Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B Smith, James M Rehg, and Le Song. Iterative machine teaching. In *International Conference on Machine Learning*, pp. 2149–2158. PMLR, 2017.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

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

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

Yuta Saito, Suguru Yaginuma, Yuta Nishino, Hayato Sakata, and Kazuhide Nakata. Unbiased recommender learning from missing-not-at-random implicit feedback. In *Proceedings of the 13th international conference on web search and data mining*, pp. 501–509, 2020.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pp. 1670–1679. PMLR, 2016.

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

Ruida Wang, Wangchunshu Zhou, and Mrinmaya Sachan. Let's synthesize step by step: Iterative dataset synthesis with large language models by extrapolating errors from small models. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 11817–11831, 2023.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2400–2409, 2024.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM international conference on web search and data mining*, pp. 806–815, 2024.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Chao Yi, Dian Chen, Gaoyang Guo, Jiakai Tang, Jian Wu, Jing Yu, Mao Zhang, Sunhao Dai, Wen Chen, Wenjun Yang, et al. Recgpt technical report. *arXiv preprint arXiv:2507.22879*, 2025.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 2024.

A SEC DIAGNOSIS PROMPT

Each independent call receives the complete inputs $(s_r, \mathcal{H}_r, e_r)$ through the braced placeholders (Eq. 2). In the recommendation retrieval implementation, each request is a target category $q_{r,b} = c_{r,b}$ with a separate justification. Supervision construction, training, and evaluation are executed outside the prompt.

```
Recommend supervision-construction requests for the next round
using updated model feedback and the available data context.

INPUTS
- Current state {s_r}: task objectives, metric definitions, and
  current performance summaries.
- History {H_r}: previous construction requests and the state
  changes observed after learning from the resulting
  supervision; empty in the initial round.
- Environment {e_r}: existing supervision coverage and
  composition, available source data, the construction interface
  and request schema, and applicable feasibility and budget
  constraints.

DIAGNOSIS GUIDELINES
1. Examine current performance together with existing
   supervision to identify opportunities for further supervision
   expansion.
2. Use historical feedback to update construction priorities,
   considering which limitations persist and how performance
   changed following earlier expansions. If history is empty,
   rely on the current state and available data context.
3. Formulate requests that the supplied construction interface
   can execute under the available data and resource
   constraints.
4. Allow previously expanded targets to be revisited when
   updated evidence supports further construction.

DECISION BOUNDARIES
- Ground recommendations in the supplied evidence. Do not invent
  performance results, data availability, or construction
  capabilities.
- Treat observed changes as decision feedback rather than
  definitive evidence of the cause of a performance limitation.
- Specify what supervision to construct; do not generate
  training examples or determine their learning order.
- Follow the supplied request schema and constraints. Identify
  material missing information rather than assuming unsupported
  values.

OUTPUT
Return the construction requests, each accompanied by a concise
justification referencing the current state, history, or data
context. Keep each justification separate from the executable
request fields. Report any missing information that prevents a
supported request.
```

B OFFLINE EVALUATION PROTOCOL

Table 7 summarizes the shared configuration and starting checkpoints; their test utilities are given in Table 2.

Table 7: Shared offline configuration.

Field	Setting
Backbone and objective	Qwen3-0.6B; NTP
Historical supervision	Approximately 40 million pairs
Evaluation categories	Approximately 300; fixed across models
Source-data window	Fixed 10-day window across all offline rounds
Decision set	First calendar day after the source window
Test set	Calendar day immediately after the decision day
Evolution rounds R	6
Total added supervision	Matched across arms over all R rounds
New pairs per request n_r	100,000, including revisits
Committed updates per batch N_r	4000
Probe length k^*	1250
Independent runs	3
Construction-fork checkpoint	Arm A after round 1
Construction-fork scope	Round 2 supervision; one Joint continuation
State-crossed checkpoints	Arm A after rounds 1 and 5
State-crossed supervision	Original-plan and Updated-diagnosis sets constructed for round 6
Same-supervision checkpoint	Arm A at 0 completed rounds
Same-supervision batches	All batches from arm A's six-round trajectory

B.1 Evaluation and Shared Construction Conditions

For each evaluation instance x, let $\mathcal{Y}(x)$ be its logged clicked targets and $\mathcal{P}_\theta(x)$ the model's top-1500 retrieved items. Define $\mathcal{Y}_c(x) = \{y \in \mathcal{Y}(x) : \text{category}(y) = c\}$ and let $\mathcal{E}_c$ contain the evaluation instances with $\mathcal{Y}_c(x) \neq \varnothing$. Category recall averages the fraction of retrieved targets within category c over these instances:

$$R_c(\theta) = \frac{100}{|\mathcal{E}_c|} \sum_{x \in \mathcal{E}_c} \frac{|\mathcal{P}_\theta(x) \cap \mathcal{Y}_c(x)|}{|\mathcal{Y}_c(x)|}. \tag{14}$$

With one target per instance, this groups instances by their target's category and measures the fraction whose target appears in the top-1500 results. With multiple targets, both the numerator and denominator use only targets in category c; an instance contributes to each category represented among its targets. Overall utility is the unweighted mean of $R_c(\theta)$ over the fixed evaluation categories (Eq. 13). Decision and test benchmarks share this definition; the former guides diagnosis and selection, while the latter is held out for evaluation. All arms use the same candidate inventory and evaluation population, regardless of which categories they expand.

All offline rounds and arms use the same source data drawn from a fixed 10-day window. Each arm maintains its own model, diagnostic history, and record of supervision already learned, while constructed batches are shared across arms. A request for an already constructed category expansion reuses its batch without regeneration; a further expansion of the same category is a distinct batch. Reused batches retain their contents, source data, and library version. First-round category requests are shared across all arms; request policies can diverge from round two onward. The common constructor produces any additional batches required by their requests. The common eligible category universe is fixed before the experiment by source-data availability and pair-quota feasibility; previous expansion does not remove eligibility. Low-recall orders these categories by current decision recall, resolving ties by a fixed category order.

B.2 Training and Controlled Comparisons

Each progressive stage trains on $\mathcal{D}_0$ plus its current batch for $N_r = 4000$ updates; completed additional batches are not replayed. The progressive policies share per-batch budgets, historical/new-data mixing, and schedule settings. Probes branch from the same model, optimizer, and schedule state and follow the first $k^* = 1250$ updates of the full stage. Feedback retains its selected probe and completes the remaining 2750 updates without restarting the schedule. Initial-order retains its first

selected probe, then trains subsequent batches from the current committed model rather than saved initial probes. Exact probe-score ties follow the supplied candidate order.

Joint trains on $\mathcal{D}_0$ and all compared batches under the same total committed-update budget: $m_r N_r$ for a single round, or the sum of the assigned stage budgets over the full collection in Table 5. Its total historical/new-data allocation matches the progressive policies, with new-data sampling balanced across batches. All construction-fork and state-crossed continuations use this joint protocol to hold the learning policy fixed.

Controlled comparisons clone the model, optimizer, and schedule state. Construction forks also clone the source snapshot, supervision coverage, diagnostic history, and category libraries. Training randomness is paired across branches where applicable. State-crossed continuations reuse frozen supervision without regeneration. In Table 5, batches retain their contents and construction-instance identities. Paired contrasts are computed within each run before averaging across runs.

B.3 Expansion Records

Table 8 summarizes arm D's expansion counts and training-pair additions by round. First-time and revisited counts refer to D's own expansion history; a batch previously used by another arm can still be a first-time expansion for D. Each first-time or revisit expansion adds $100{,}000$ pairs, with a revisit adding a distinct batch rather than replaying an earlier one. Shared batches count toward D's training allocation without being regenerated. Arm D incorporates 48 batches totaling 4.8 million pairs, covering 45 distinct categories with three revisit events. The underlying records retain category and construction-instance IDs, source-data provenance, and library versions.

Table 8: Per-round expansion records for arm D.

Arm	Round	First-time	Revisited	New pairs (M)
D (full)	1	5	0	0.50
D (full)	2	12	0	1.20
D (full)	3	9	1	1.00
D (full)	4	6	2	0.80
D (full)	5	6	0	0.60
D (full)	6	7	0	0.70

C Probe Calibration and Computational Cost

C.1 Probe Ranking Fidelity

We examine probe ranking in an independent component study with a fixed pool of eight preconstructed category batches and a Qwen3-0.6B retriever. Each batch contains approximately $100{,}000$–$200{,}000$ pairs. Candidate continuations train on historical supervision plus the candidate batch and are scored by overall decision-set utility (Eq. 13). Full stages use 4000 updates, and the operating probe length is 1250, as in the main experiments. The component study's decision and test sets each contain 1.1 million instances.

Figure 2 compares probe rankings with full-stage rankings at this study's initial checkpoint. Across the eight candidates, the adopted probe length of 1250 steps yields Kendall $\tau_a \approx 0.86$.

For a local check after learning, we compare men's casual shoes and infant formula from a common checkpoint reached after two stages incorporating laptops and dresses. Table 9 reports decision-set gains averaged across three seeds. Initial probes are measured relative to this study's initial model; updated probes and full-stage gains use the common post-stage-two checkpoint.

Men's casual shoes rank above infant formula initially, but updated probes reverse this ordering, in agreement with the full-stage decision gains. From the same checkpoint, the infant-formula branch also exceeds the men's-casual-shoes branch on the test set by 0.10 pp. This local comparison supports refreshing probe rankings as the model changes; probe-gain magnitudes need not match full-stage gains.

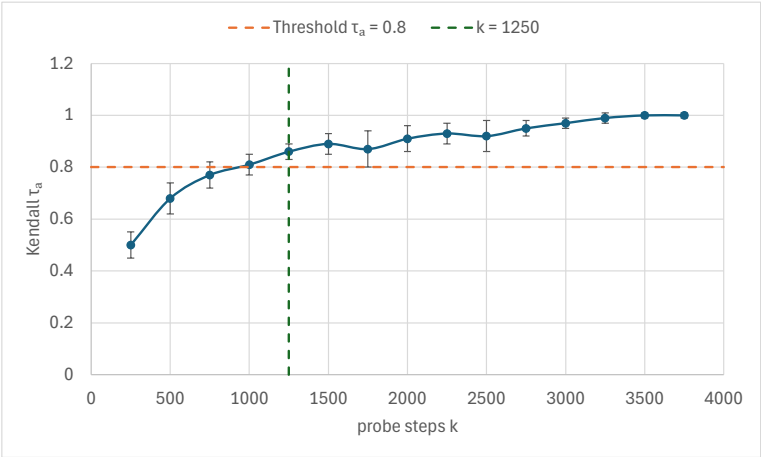

Figure 2: Probe ranking fidelity at the component study's initial checkpoint.

Table 9: Local probe and full-stage decision gains (pp).

Candidate	Initial probe	Updated probe	Updated full stage
Men's casual shoes	0.067	0.083	0.14
Infant formula	0.052	0.104	0.24

C.2 Computation Accounting

We account for learning-policy computation over the complete six-round batch collection in Table 5. Committed updates include retained probe steps exactly once; discarded probes are counted separately. Joint has no ranking probes, Initial-order scores the full collection once, and Feedback probes the remaining batches after each stage. Equal committed budgets therefore do not imply equal total computation. Calibration and controlled-study continuations are evaluation expenses rather than training-policy costs.