

MetaStrategy: Generative Ranking with Executable LLM Strategies

Chengyu Lai
Jiuning Lin
Zhibo Xiao
Taobao & Tmall Group of Alibaba
Hangzhou, China
lcy477544@alibaba-inc.com
linjiuning.ljn@alibaba-inc.com
xiaozhibo.xzb@alibaba-inc.com

Xiaodong Zhu
Wuhan University
Wuhan, China
xiaodongzhu@whu.edu.cn

Ruiquan Lan
Taobao & Tmall Group of Alibaba
Hangzhou, China
lanruiquan.lrq@alibaba-inc.com

Bin Zhang
Zihong Huang
Wendong Zhang
Taobao & Tmall Group of Alibaba
Hangzhou, China
tianji.zb@alibaba-inc.com
huangzihong.hzh@alibaba-inc.com
xianzhi.zwd@alibaba-inc.com

Chuxin Chen
Yinjiang Cai
Taobao & Tmall Group of Alibaba
Hangzhou, China
chenchuxin.ccx@alibaba-inc.com
caiyinjiang.cyj@alibaba-inc.com

Shuai Zhong
The University of Hong Kong
Hong Kong, Hong Kong SAR
chris_zsa@connect.hku.hk

Lingqing Zhang
University of Cambridge
Cambridge, United Kingdom
lingqing.zhang@mail.mcgill.ca

Dimin Wang
Jialin Zhu
Han Zhu
Taobao & Tmall Group of Alibaba
Hangzhou, China
dimin.wdm@alibaba-inc.com
xiafei.zjl@alibaba-inc.com
zhuhan.zh@alibaba-inc.com

Abstract

Industrial recommender systems rank heterogeneous content under coupled user, business, commercial, and experience objectives. Existing generative ranking methods typically construct item sequences directly, making them difficult to integrate with mature predictive models, operational rules, and field-level guardrails. We present MetaStrategy, a framework that instead generates a structured, executable ranking strategy. Conditioned on request context, a large language model (LLM) policy emits a typed JSON bundle controlling objective weights, content and category preferences, experience constraints, and position policies. A deterministic validator and compiler instantiate an isolated Generator that competes atomically with incumbents under the list-level Evaluator of the Generator-Evaluator (GE) architecture. We train the policy in a production-path replay environment that re-executes logged requests through the current re-ranking stack without user exposure. The method combines selection, relative-rank, and baseline-lift rewards, a self-competitive curriculum that feeds frequent strategies back as competitors, and Evaluator-routed reward-augmented on-policy distillation that transfers complementary 4B-parameter Teachers into a compact 0.8B-parameter Student. We deploy MetaStrategy in Taobao Homepage Guess You Like through diff-triggered nearline generation; LLM inference remains outside synchronous ranking, with no observable increase in response time (RT). In a seven-day user-randomized online A/B test, MetaStrategy wins 27.93% of treatment-side GE calls and significantly improves click page views (click PV) by 2.11%, item-detail page views (IPV) by 3.12%, and transaction amount by 2.83%.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Reinforcement learning*.

Keywords

industrial recommendation, generative ranking, large language models, multi-objective ranking, reinforcement learning, on-policy distillation

ACM Reference Format:

Chengyu Lai, Jiuning Lin, Zhibo Xiao, Xiaodong Zhu, Ruiquan Lan, Bin Zhang, Zihong Huang, Wendong Zhang, Chuxin Chen, Yinjiang Cai, Shuai Zhong, Lingqing Zhang, Dimin Wang, Jialin Zhu, and Han Zhu. 2027. MetaStrategy: Generative Ranking with Executable LLM Strategies. In *Proceedings of the 33rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '27)*. ACM, New York, NY, USA, 10 pages.

1 Introduction

Industrial recommendation lists are rarely determined by a single relevance score. An e-commerce homepage may mix products, content posts, short videos, live streams, and sponsored items [25]. These content types have different prediction targets and operational constraints, yet they compete for the same exposure positions. Production systems therefore place a strategy layer above their predictive models to coordinate objective weights, content composition, category preferences, exposure filters, density constraints, and position-specific rules [24].

This strategy layer is reliable but difficult to personalize. Global multi-objective formulas cannot express every request-specific trade-off, and tuning rules independently ignores their interactions [15]. For example, a click-oriented weight change may be overridden by a top-position policy, while a content-type boost may violate a density constraint after re-ranking. The decision must therefore be made jointly, conditioned on user context, current intent, and available candidates.

Generative recommendation provides a natural way to model such joint decisions [21, 26]. Existing work formulates recommendation as text generation, semantic-identifier generation, or autoregressive slate construction [5, 7, 20]. Directly generating item sequences, however, creates a difficult integration boundary in a

mature high-traffic system. The generated sequence must reproduce established filters, calibrated predictors, quotas, and safety controls, while offering no simple way to inspect or correct an individual business decision.

We instead generate the ranking *strategy* and let the production ranker execute it. The strategy policy consumes a real-time multi-intent representation produced by an upstream intent model, historical user statistics, and an ordered sequence of recent interactions. Given this context, a large language model (LLM) produces one schema-constrained JSON bundle that jointly adjusts several ranking modules. A deterministic tool process validates the bundle and translates its semantic actions into existing production parameters. The LLM does not emit item identifiers or an item permutation. This separation retains the expressive joint policy of an autoregressive model while preserving the control surface of the deployed ranker.

MetaStrategy operates within a Generator-Evaluator (GE) architecture [6]. Each Generator transforms a shared candidate set containing hundreds of items into a list containing tens of items. A discriminative list-level Evaluator scores the complete lists with user, item, position, and business context, then selects the list to expose. The LLM controls one isolated Generator and competes against approximately ten incumbent production Generators. It can therefore improve the final decision without bypassing the existing production boundary.

Reinforcement learning (RL) in this setting presents two challenges. The Evaluator is useful but imperfect, so direct reward optimization can collapse to a few extreme bundles that exploit stable proxy-model preferences rather than adapt to each request. Moreover, large LLMs are valuable teachers during offline exploration but are too expensive for the intended serving budget. We address collapse with a self-competitive curriculum that freezes frequent compiled strategies as new Generators in the next training round. We address serving cost with Evaluator-routed reward-augmented on-policy distillation (OPD): multiple teachers are compared atomically on each replayed request, and the best valid teacher supplies an on-policy token signal to a compact student without replacing the student's GE reward.

We instantiate the framework in Taobao Homepage Guess You Like. Strategy generation runs nearline and is refreshed only when decision-relevant context changes; the online ranker reads the latest validated strategy and applies it to the LLM-controlled Generator. The deployed policy is the final 0.8B Student trained with Evaluator-routed reward-augmented OPD. The online request never waits for LLM inference, and the lightweight lookup and deterministic resolution add no observable response-time (RT) overhead. Integration is additive: incumbent Generators, the list-level Evaluator, and production fallback behavior remain unchanged. In an online A/B test, MetaStrategy increases exposure page views (exposure PV) by 1.49%, click page views (click PV) by 2.11%, item-detail page views (IPV) by 3.12%, and transaction amount by 2.83% relative to the production control.

Our contributions are:

- We formulate generative ranking as request-conditioned generation of an executable joint strategy. A typed action

schema and deterministic compiler let the LLM control multiple production ranking modules while preserving validation, isolation, and fallback behavior.

- We build a production-path replay environment that compares incumbent and LLM-controlled Generators in one call to the current re-ranking stack. Selection, relative-rank, and baseline-lift signals provide request-level feedback without exposing replay traffic to users.
- We introduce a self-competitive curriculum that turns frequent compiled strategies into explicit competitors, and an Evaluator-routed, reward-augmented OPD objective that compresses complementary teachers while retaining direct GE optimization.
- We deploy MetaStrategy in Taobao Homepage Guess You Like through a non-intrusive serving architecture: nearline LLM inference adds no observable online RT overhead, while an isolated Generator preserves the incumbent ranking and fallback paths. We report significant online gains in click PV, IPV, and transaction amount.

2 Related Work

Industrial multi-objective ranking. Large-scale recommenders commonly separate candidate generation and ranking [4], while multi-task architectures such as the Entire Space Multi-Task Model (ESMM) and Multi-gate Mixture-of-Experts (MMoE) improve estimates for coupled engagement and transaction objectives [16, 17]. Industrial mixed-content feeds must additionally coordinate cross-channel exposure constraints and perception-aware diversification [24, 25]. Personalized re-ranking also studies popularity and diversity trade-offs [1, 12]. These approaches improve predictive signals or a particular re-ranking objective. MetaStrategy instead learns a request-level policy that jointly configures multiple objectives and operational rules while leaving the predictive stack intact.

Listwise re-ranking and slate optimization. Seq2Slate autoregressively constructs a slate, and SLATEQ develops tractable reinforcement learning over slate actions [3, 11]. Personalized re-rankers model interactions among candidates in the final list [14, 19]. Generative re-rankers further model complete-list construction through context-wise, non-autoregressive, or permutation-level objectives [6, 21, 23]. Their principal action is still an item order or permutation. Our action is a bounded executable strategy; list construction remains delegated to the production ranker, and the resulting list is selected under the incumbent list-level Evaluator.

Generative and LLM-based recommendation. P5 and M6-Rec formulate recommendation tasks as language modeling [5, 7]; LLMRank applies an LLM as a zero-shot ranker [9]; and TIGER and OneRec generate semantic identifiers or recommendation sequences [20, 26]. Surveys summarize the rapidly expanding design space of LLM-based and generative recommendation [13, 22]. In contrast, MetaStrategy does not ask the LLM to reproduce retrieval, scoring, filtering, or list construction. It uses the LLM as a typed strategy policy over existing production mechanisms, preserving their validation, guardrails, and fallback paths.

Reinforcement learning and distillation. Recommendation and ranking have been optimized with reinforcement learning over item or slate actions [10, 11]. Conventional knowledge distillation

transfers a fixed Teacher to a smaller Student [8], while on-policy distillation evaluates Teacher feedback on Student-generated trajectories to reduce distribution mismatch [2]. Our learning signal instead evaluates an executed strategy against production Generators. Frequent compiled modes become new competitors in the curriculum, and the distillation Teacher is routed per request through a shared list-level evaluation before its token signal is combined with the Student’s GE reward.

3 Setting and Problem Formulation

3.1 Industrial Strategy Composition

Let x denote the context of one recommendation request and $I = \{i_1, \dots, i_n\}$ its shared candidate set. The user-side context contains three complementary views: (1) a real-time multi-intent representation from an upstream intent model; (2) historical statistics that summarize longer-term preferences and engagement; and (3) an ordered real-time behavior sequence, including clicks, add-to-cart events, purchases, and other recent interactions. It is combined with request state and the candidate-supply summary required for strategy decisions. In our deployment, n is on the order of hundreds, while the exposed list contains tens of items.

A production ranking strategy is a composition of decisions rather than one scalar score. A simplified objective fusion can be written as

$$q(i | x) = \alpha(x) \widehat{v}_{\text{eng}}(i, x) + \beta(x) \widehat{v}_{\text{txn}}(i, x) + \gamma(x) \widehat{v}_{\text{com}}(i, x), \quad (1)$$

but the final list also depends on filtering, supply mixing, diversity, and position rules. Existing multi-task models improve the underlying predictions [16, 17]; our focus is the request-level policy that coordinates these predictions and rules. We use CTR, CVR, and GMV for click-through rate, conversion rate, and gross merchandise value, respectively; the prefix p denotes a model prediction.

3.2 Generator-Evaluator Architecture

For strategy s_j , Generator G_j maps the shared candidates to an ordered list:

$$L_j = G_j(I, x; s_j), \quad |L_j| \ll |I|. \quad (2)$$

The Generators may represent existing ranking recipes, manually designed strategies, or the LLM-generated strategy introduced below. A list-level Evaluator assigns

$$u_j = E(L_j, x), \quad j^* = \arg \max_{j \in C} u_j, \quad L^* = L_{j^*}, \quad (3)$$

where C is the active Generator pool. Unlike a pointwise ranker, E consumes the full list with user, item, position, and business context. Its supervised heads cover point-level click and transaction behavior, down-scroll behavior, list-level click outcomes, and effective cost per mille (eCPM). Their calibrated combination represents the deployment utility used to compare complete lists.

The Evaluator is intentionally separate from the LLM. It represents the platform-side decision boundary, including user value, commercial value, and experience constraints. The LLM policy searches for a personalized strategy within this boundary.

3.3 Strategy Generation Objective

Let $\mathcal{A}$ be the schema-constrained strategy space and $\pi_\theta(a | x)$ the LLM policy. The LLM produces one complete bundle $a \in \mathcal{A}$ per

generation. A deterministic tool process $\mathcal{T}$ converts the semantic action into production parameters, and the parameters control an isolated Generator $G_{\mathcal{T}(a)}$. The learning objective is

$$\max_{\theta} \mathbb{E}_{x \sim \mathcal{D}, a \sim \pi_\theta(\cdot | x)} [E(G_{\mathcal{T}(a)}(I, x), x)], \quad (4)$$

subject to schema validity, execution constraints, and platform guardrails. Because the bundle contains multiple dependent modules, the policy factorizes autoregressively:

$$\pi_\theta(a | x) = \prod_{m=1}^M \pi_\theta(a_m | x, a_{<m}). \quad (5)$$

The small domain of each individual field does not make the problem enumerable: the joint space grows exponentially, and later modules can reinforce or override earlier decisions.

4 MetaStrategy

Figure 1 summarizes the execution and learning loop. MetaStrategy compiles request-conditioned strategies into one isolated Generator, obtains list-level feedback from the atomic GE call, and closes the loop through curriculum competition and routed OPD. The upstream intent model supplies a current multi-intent state, while the strategy policy also observes historical aggregates and raw recent behavior. This division avoids relearning intent while retaining temporal evidence absent from a compressed state.

4.1 Executable Strategy Bundle

The policy emits one JSON object enclosed by a fixed answer delimiter. The object contains five ordered executable modules spanning objective weights, content-type and category preferences, experience constraints, and top-CTR policies. The complete module definitions and value domains are detailed in Appendix A.

Bounded, named fields make decisions reviewable, expose cross-module interactions, and allow field-level limits. Sharing the schema across training, replay, nearline generation, and serving reduces training-serving skew.

4.2 Validation, Compilation, and Isolation

An output passes through four deterministic gates. The parser first extracts the answer block and parses JSON. Schema validation then checks module completeness, ordering, field types, enumerations, and numeric ranges. The tool process maps each semantic action to production parameters, including Generator configuration, diversity rules, and top-position settings. Finally, the compiled parameters are attached only to the designated LLM-controlled Generator.

Invalid Student output receives a format penalty and skips Student tool execution. In multi-Teacher training, a malformed Teacher is removed from the routing set, while other valid candidates may still be evaluated. This fail-closed behavior prevents missing fields or empty configurations from becoming a shortcut. Configuration isolation also makes the replay reward attributable to the LLM strategy rather than an accidental change to another Generator.

4.3 Production-Path Replay Environment

Panel (c) of Figure 1 depicts our production-path replay environment, which replaces a locally implemented user simulator. Each

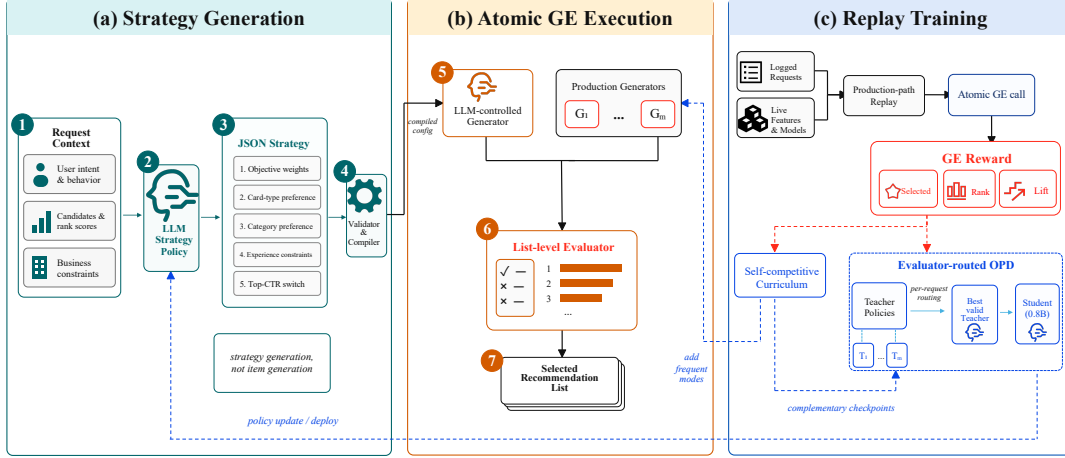


Figure 1: Overview of MetaStrategy. (a) The LLM maps request context to one structured JSON strategy, which is validated and compiled rather than directly generating an item sequence. (b) The compiled configuration controls one Generator that competes with production Generators under a shared list-level Evaluator. (c) Production-path replay supplies the three GE rewards used for self-competitive curriculum learning and per-request evaluator-routed OPD into the deployed 0.8B Student.

episode is initialized from the input log of a real online request. The log preserves the candidate set, its fine-ranking scores, and the request context required by the re-ranking stage. Recall and fine-ranking inference are not replayed; the episode starts from these fixed candidates and scores and executes the production re-mixing and re-ranking path through isolated load-test traffic. At replay time, the request resolves current online features and invokes the current production models, including a fresh list-level Evaluator score for every generated list.

Within one atomic GE invocation, the LLM-controlled Generator competes against approximately ten production Generators, including perturbation-based and context-aware re-estimation variants. All Generators operate on the same request and candidate set, and all valid lists are scored by one Evaluator invocation. A Generator that times out, fails execution, or returns an invalid list is removed from the comparison; the remaining valid Generators are still evaluated. Consequently, reward computation uses the valid Generator set of that invocation.

Replay is isolated from user-facing serving. Its selected list is returned to the training process but is never exposed to a user, and the request does not write impression logs, attribution records, or business counters. An episode is a single-step contextual decision: the environment does not synthesize clicks or transactions and does not simulate a subsequent user-state transition. The Evaluator output is therefore a proxy utility for the resulting list rather than simulated or observed user feedback.

Repeated executions of the same logged request need not produce identical results. Online features are resolved at replay time, and the Evaluator and other production models may change with online releases. We therefore avoid comparing absolute scores across replay invocations. Instead, the learning signal is based on paired, within-request comparisons among Generators that share one atomic GE invocation. This design preserves fidelity to the current production re-ranking stack, while restricting optimization to the support of

the logged candidates and fixed fine-ranking scores. It is not an exact reconstruction of the historical serving state or a long-horizon user simulator.

4.4 Evaluator-Derived GE Reward

For each replay invocation, the environment returns the scores of all valid Generator lists and the Generator selected by the Evaluator. The LLM-controlled Generator is compared with a designated production baseline and the other valid production Generators under this shared evaluation state.

Let u_S be the Student Generator score, u_B the designated baseline score, N the number of valid competing lists, and $\text{rank}_S \in \{0, \dots, N-1\}$ the descending rank of u_S . We use three complementary rewards:

$$r_{\text{sel}} = \mathbb{I}[j^* = S], \quad (6)$$

$$r_{\text{rank}} = \frac{N - 1 - \text{rank}_S}{\max(N - 1, 1)}, \quad (7)$$

$$r_{\text{lift}} = \text{clip}\left(\frac{u_S - u_B}{\max(|u_B|, \epsilon)}, -c, c\right). \quad (8)$$

Selection is aligned with the final GE decision but sparse. Rank gives a dense relative signal across all competitors. Lift measures continuous improvement over the same-call production reference. The aggregate reward is

$$r_{\text{GE}} = w_{\text{sel}} \cdot r_{\text{sel}} + w_{\text{rank}} \cdot r_{\text{rank}} + w_{\text{lift}} \cdot r_{\text{lift}}. \quad (9)$$

For group-based policy optimization, multiple bundles are sampled for the same request and the reward is normalized within the group:

$$A_{q,k}^{\text{RL}} = \frac{r_{q,k} - \mu_q}{\sigma_q + \epsilon}. \quad (10)$$

This advantage is used by a clipped policy objective. The reward depends on the executed list, while the policy action remains the strategy bundle.

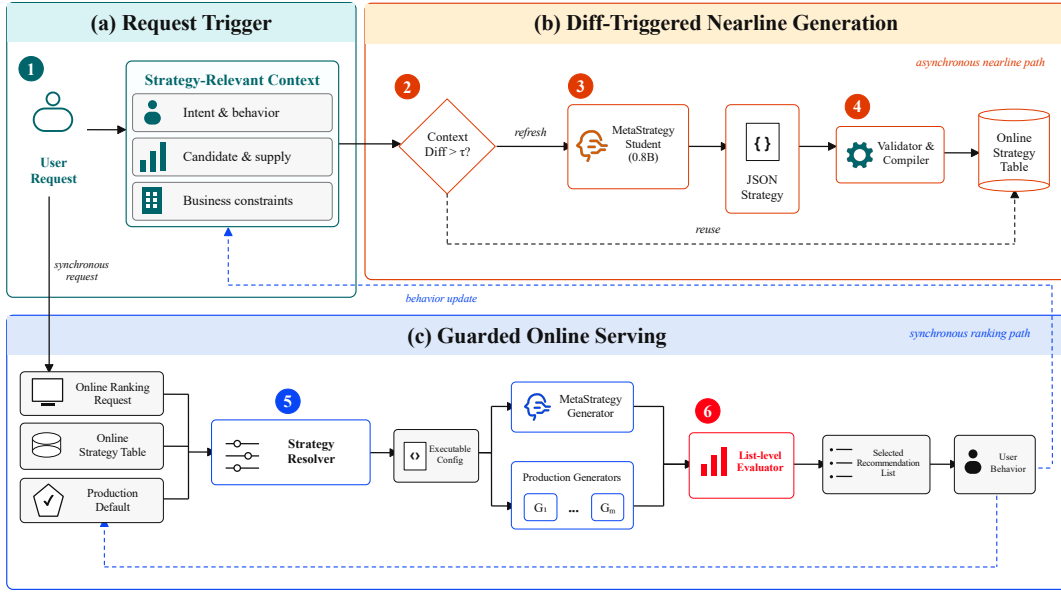


Figure 2: Nearline generation and guarded GE serving. Requests proceed to online ranking while a context-difference gate asynchronously reuses or refreshes the strategy. Valid Student output is compiled and published; the synchronous resolver selects a personalized or default configuration for the isolated MetaStrategy Generator under the incumbent Evaluator.

4.5 Self-Competitive Strategy Curriculum

The Evaluator is a learned proxy and may contain exploitable regularities. In early training, we observe that a policy can repeatedly push bounded fields to extreme values or emit nearly identical bundles for unrelated requests. Such a policy may score well in replay while providing little request-level adaptation.

Our curriculum turns the policy’s dominant behavior into its next competitor, as shown by the frequent-mode feedback path in Figure 1(c). At curriculum round t , the policy is trained against Generator pool C_t :

$$\theta_t = \text{RL_Train}(\mathcal{D}, C_t). \quad (11)$$

We execute valid sampled bundles and compute a canonical signature over the compiled configuration, $z = \text{Sig}(\mathcal{T}(a))$. Using post-compilation signatures merges JSON variants that are operationally equivalent. Let $\mathcal{Z}_t$ contain the k most frequent signatures. The next pool is

$$C_{t+1} = C_t \cup \{G(z) : z \in \mathcal{Z}_t\}. \quad (12)$$

Each $G(z)$ is a frozen Generator that executes the corresponding strategy. The next policy must outperform these now-explicit modes rather than obtain reward by rediscovering them.

The curriculum is an outer loop across training jobs. It does not enumerate the joint strategy space and does not require online execution of multiple LLM outputs. Checkpoints from different rounds, random seeds, model scales, or training preferences can form a complementary Teacher pool. We additionally track compiled-signature diversity and bounded-action saturation; these are diagnostics and optional regularizers, while Eq. 12 is the mechanism that raises competition difficulty.

4.6 Evaluator-Routed Reward-Augmented OPD

The distillation branch in Figure 1(c) compresses a pool of expensive strategy policies into a compact Student. Routing is performed per request, and a valid comparison requires all candidate strategies to see the same replay state. For request x , the Student and M Teachers generate

$$a^S \sim \pi_\theta(\cdot | x), \quad a^i \sim \pi_i(\cdot | x), \quad i = 1, \dots, M. \quad (13)$$

Every valid bundle is compiled into an isolated Generator rule. The Student, Teachers, and production baselines are submitted in one replay call, so their lists share the original request, candidates, and Evaluator state.

Teacher routing excludes the Student. Among Teachers that pass parsing, schema, tool execution, and Evaluator scoring, we choose

$$i^*(x) = \arg \max_{i \in \mathcal{V}(x)} E(G_{\mathcal{T}(a^i)}(I, x), x), \quad (14)$$

where $\mathcal{V}(x)$ is the valid Teacher set. A deterministic identifier breaks ties. If $\mathcal{V}(x)$ is empty, the request is routed to `__none__`: its RL reward is retained and the distillation term is masked.

OPD is evaluated on the Student’s sampled token trajectory, not on a Teacher-forced target sequence. At response token ℓ ,

$$d_\ell(x) = \log \pi_\theta(a_\ell^S | x, a_{<\ell}^S) - \log \pi_{i^*(x)}(a_\ell^S | x, a_{<\ell}^S). \quad (15)$$

Under Student sampling, this log-ratio estimates the reverse direction of the Kullback–Leibler (KL) divergence, $\text{KL}(\pi_\theta \| \pi_{i^*})$. We combine it directly with the reward advantage:

$$A_\ell^{\text{mix}} = A_\ell^{\text{RL}} - \lambda_{i^*(x)} d_\ell(x), \quad (16)$$

and use A_ℓ^{mix} in the standard clipped policy objective. Only the routed Teacher contributes reference log-probabilities. This is not

an additional Teacher-forced cross-entropy loss or a standalone KL loss.

Keeping A^{RL} matters. Pure imitation cannot exceed the routed Teacher and can propagate poor supervision when every Teacher is weak. The mixed objective preserves exploration against the production GE reward, uses the best comparable Teacher when one is available, and degrades to RL-only training when routing is unavailable.

4.7 Two-Stage Training Procedure

Teacher construction iteratively trains policies against curriculum pools, adds frequent compiled modes as competitors, and retains complementary checkpoints. Student compression atomically evaluates one bundle from each policy, routes the best valid Teacher, and updates the Student with Eq. 16. The Teacher pool is offline-only; each nearline refresh invokes only the deployed Student.

5 Production Deployment

MetaStrategy is deployed in the Taobao homepage Guess You Like feed, where products, content posts, short videos, live streams, and sponsored items compete under user, transaction, commercial, and experience objectives. As Figure 2 shows, LLM inference runs in an asynchronous nearline branch; the synchronous path performs only strategy lookup and deterministic resolution. This temporal and structural decoupling adds no observable RT overhead and leaves incumbent Generators, the Evaluator, and fallback behavior unchanged.

Diff-triggered generation. As shown in Figure 2, a request proceeds directly to the synchronous ranker while a nearline branch assembles strategy-relevant context from intent and behavior, candidate and supply summaries, and business constraints. A feature-aware difference gate invokes the deployed 0.8B Student only when no valid entry exists or the context has changed sufficiently; otherwise it reuses the latest strategy. The Student emits the same typed JSON bundle used during training. Validation and compilation precede publication to the online strategy table, so failed refreshes cannot replace the latest valid entry.

Guarded GE serving. The synchronous path performs no LLM inference. A deterministic resolver reads the latest valid entry or the production default and attaches the executable configuration only to the isolated MetaStrategy Generator. Its list competes with unchanged production Generators under the incumbent Evaluator and is exposed only when selected by the atomic GE call. Temporal decoupling, validation-before-publication, deterministic fallback, and structural isolation preserve online RT, support immediate rollback, and introduce no observable latency increase. Appendix B gives the complete refresh and fallback protocol.

6 Experiments

The evaluation addresses four questions:

- RQ1:** Does executable strategy generation improve list quality and add incremental value beyond incumbent production Generators?
- RQ2:** How do model scale and successive training stages affect executability, strategy concentration, and contribution to the GE pool?

RQ3: Can routed reward-augmented OPD compress the 4B policies into a competitive 0.8B Student?

RQ4: What user and platform outcomes does the method produce under online deployment?

6.1 Experimental Setup

Industrial replay dataset. We collect online request logs from Taobao Homepage Guess You Like over eight consecutive days, randomly sampling 8,192 requests per day for 65,536 requests in total. Days 1–6 form the training set (49,152 requests), Day 7 the validation set (8,192), and Day 8 the test set (8,192). Products, content posts, short videos, live streams, and sponsored items share this recommendation surface. Each log preserves the strategy-policy context, candidate set, fine-ranking scores, and constraints required by the re-ranking stage. During replay, these candidates and scores remain fixed, while current online features, production Generators, and the list-level Evaluator execute through the isolated production path. Test-period feedback is excluded from RL, curriculum construction, OPD, prompt selection, and hyperparameter tuning.

Replay protocol. The production rows are measured in the unmodified incumbent Generator pool $\mathcal{P}$, and each LLM benchmark Generator G_j is added individually to the same pool. All methods use exactly the same logged requests and pinned production-model and Evaluator versions. Their calls are interleaved within the same replay window to balance temporal variation in online feature lookups. Within an atomic call, all Generators share the logged request, candidate set, fixed fine-ranking scores, online feature state, and one Evaluator invocation. We compute utility relative to BaseG or the incumbent pool within that invocation. A strategy is valid when it passes parsing and compilation and returns an executable list. Only valid lists enter the GE comparison; validity is measured over all test requests, and a request without a valid benchmark list contributes zero to that method’s selection rate and incremental GE lift.

Production Generator baselines.

- **Base G** greedily fills each position with the remaining card having the highest production rank score while satisfying the same hard filtering, position, density, and diversity constraints used by all methods. It is the reference method for relative improvements.
- **GNR G** [18] is a sequential, context-aware production Generator. At each position, it conditions on the cards already placed, re-estimates click propensity for the remaining candidates, and selects the next card under the production constraints.
- **NAR G** [23] predicts position-specific scores in parallel and uses a non-autoregressive list constructor. It captures position preference without sequential inference.
- **Single-objective perturbation G** starts from the BaseG scoring formula and perturbs one business objective while retaining the same rule stack. We instantiate the production perturbation configurations before evaluating on the test period; individual target variants are reported separately when multiple configurations are used.

LLM Generator variants.

Table 1: Production-path replay on Taobao Homepage Guess You Like. Pointwise and Evaluator columns are relative changes from BaseG. Production rows use the incumbent GE call; each LLM Generator is added individually to the same pinned incumbent pool. All rows use the same request set and are interleaved within one replay window. Best and second-best comparable values are bolded and underlined, respectively. “–” denotes a metric that is not applicable.

Method	$\Delta pCTR \uparrow$	$\Delta pCVR \uparrow$	$\Delta pIPV \uparrow$	$\Delta pGMV \uparrow$	$\Delta \text{Evaluator} \uparrow$	Valid $\uparrow$	Selected $\uparrow$	$\Delta GE \uparrow$	Beat Base $\uparrow$
<i>Production ranking Generators</i>									
Base G	–	–	–	–	–	–	6.97%	–	–
GNR G	-2.43%	+2.08%	-2.75%	+14.04%	+1.68%	–	6.09%	–	37.4%
NAR (non-autoregressive) G	<u>+1.64%</u>	+15.24%	<u>+6.46%</u>	+17.03%	+2.75%	–	<u>13.7%</u>	–	63.4%
Single-objective perturbation G	+13.82%	-4.01%	+11.81%	+1.05%	+1.21%	–	4.94%	–	50.1%
<i>LLM strategy Generators</i>									
Prompt-only (0.8B)	-2.82%	+8.17%	+0.13%	+12.65%	-0.29%	4.39%	0.39%	+0.02%	1.8%
Prompt-only (4B)	-3.41%	+13.67%	-0.41%	+13.81%	+0.42%	80.86%	7.87%	+0.34%	35.69%
RL (0.8B)	-0.48%	+10.10%	+2.51%	+13.58%	+1.05%	97.12%	8.56%	+0.38%	46.88%
RL (4B)	-1.07%	+12.54%	+0.96%	+14.23%	+1.58%	98.85%	7.93%	+0.45%	45.86%
Curriculum RL (4B)	-5.70%	<u>+18.47%</u>	-2.95%	<u>+18.88%</u>	+0.56%	98.89%	11.27%	<u>+0.57%</u>	45.65%
Routed reward-augmented OPD (0.8B)	-0.68%	+22.69%	+1.65%	+37.76%	<u>+2.53%</u>	98.03%	16.24%	+0.73%	<u>62.60%</u>

- **Prompt-only** applies the common prompt, JSON action schema, validation, compiler, and execution engine without parameter updates from replay feedback. We evaluate 0.8B and 4B policies.
- **RL** optimizes the three-component reward in Eq. 9 against the original production pool. We train both 0.8B and 4B policies without feeding frequent strategies back as competitors.
- **Curriculum RL** applies the self-competitive strategy curriculum to the 4B policy, progressively adding frequent compiled strategies to the competing Generator pool.
- **Routed reward-augmented OPD** distills two 4B checkpoints into the deployable 0.8B Student. For each request, the Student learns from the valid Teacher selected by the shared Evaluator call while retaining its own GE reward.

6.2 Formal Evaluation Metrics

Let L_q^j be the length- K_q list produced by Generator G_j on request q , and let $\mathcal{V}_j$ be the requests on which it returns a valid list. For objective $m \in \{CTR, CVR, IPV, GMV\}$, the pointwise list diagnostic averages the request-preserved fine-ranking score s_m first within each list and then over valid requests:

$$S_m(G_j) = \frac{1}{|\mathcal{V}_j|} \sum_{q \in \mathcal{V}_j} \frac{1}{K_q} \sum_{i \in L_q^j} s_m(i, x_q). \quad (17)$$

Let $C_{q,j}$ be the valid Generator set in the atomic call for row j , and let $\mathcal{P}$ be the incumbent pool. Using the production tie-breaking rule, define

$$j_q^* = \text{TieBreakArgMax}_{g \in C_{q,j}} E(L_q^g, x_q). \quad (18)$$

Within the same call, define

$$u_{q,j} = E(L_q^j, x_q), \quad u_q^{\mathcal{P}} = \max_{g \in \mathcal{P}} E(L_q^g, x_q). \quad (19)$$

With $u_{q,B}$ denoting the BaseG score from the same call, Evaluator lift is

$$\Delta_{\text{Eval}}(G_j) = \frac{1}{|\mathcal{V}_j|} \sum_{q \in \mathcal{V}_j} \frac{u_{q,j} - u_{q,B}}{\max(|u_{q,B}|, \epsilon)}. \quad (20)$$

We report validity $|\mathcal{V}_j|/N$ and the selection rate

$$\text{SR}(G_j) = \frac{1}{N} \sum_{q=1}^N \mathbb{I}[q \in \mathcal{V}_j] \mathbb{I}[j_q^* = j]. \quad (21)$$

The beat-Base rate replaces the incumbent reference with $u_{q,B}$. For the primary offline endpoint, set $\tilde{u}_{q,j} = -\infty$ for invalid outputs and $\tilde{u}_{q,j} = u_{q,j}$ otherwise. Incremental GE lift is

$$\Delta_{\text{GE}}(G_j) = \frac{1}{N} \sum_{q=1}^N \frac{\max(u_q^{\mathcal{P}}, \tilde{u}_{q,j}) - u_q^{\mathcal{P}}}{\max(|u_q^{\mathcal{P}}|, \epsilon)}. \quad (22)$$

Finally, for any strategy module h , let $z_{q,j}^{(h)}$ be its compiled decision from G_j on request q . Its concentration is

$$F_{\text{top1}}(G_j; h) = \max_z \frac{1}{|\mathcal{V}_j|} \sum_{q \in \mathcal{V}_j} \mathbb{I}[z_{q,j}^{(h)} = z]. \quad (23)$$

Lower values indicate broader use of module h 's action space. All comparisons are paired by request; an invalid output contributes zero to selection and incremental lift.

6.3 Offline Evaluation

Table 1 answers RQ1–RQ3. We prioritize Δ_{GE} , the incremental utility of adding one Generator to the incumbent pool. Evaluator lift and beat-Base measure standalone list quality, selection measures competition within an atomic call, and the pointwise columns diagnose which objectives move.

Production baselines. NAR is the strongest incumbent, with +2.75% Evaluator lift, 13.70% selection, and 63.40% beat-Base. GNR obtains +1.68% Evaluator lift but is selected on only 6.09% of calls and beats BaseG on 37.40% of requests. The single-objective perturbation produces the largest pCTR and pIPV gains (13.82% and 11.81%), yet yields only +1.21% Evaluator lift and 4.94% selection. These contrasts show that improving an isolated predictor does not ensure a better heterogeneous list under the platform objective.

Scale and replay RL. Scaling the prompt-only policy from 0.8B to 4B raises validity from 4.39% to 80.86%, selection from 0.39% to 7.87%, and Δ_{GE} from 0.02% to 0.34%. Thus, scale primarily

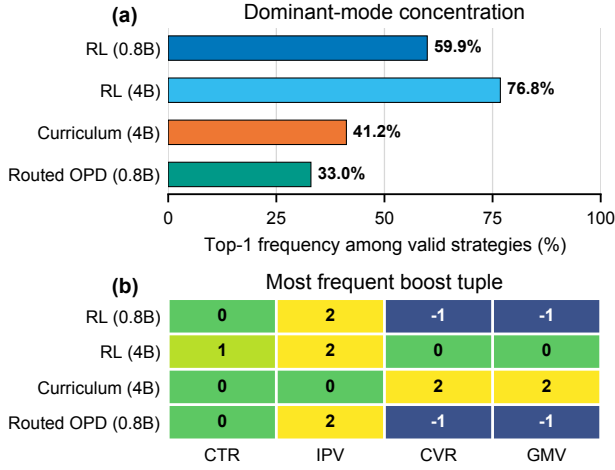


Figure 3: Collapse mitigation for ranking_weight_boost: (a) Top-1 complete-tuple frequency and (b) dominant tuple in (CTR, IPV, CVR, GMV) order.

improves the ability to emit executable strategies. Replay RL makes the 0.8B policy 97.12% valid and raises its selection and Δ_{GE} to 8.56% and 0.38%, slightly exceeding the prompt-only 4B policy on both GE metrics. At 4B, RL further increases Evaluator lift to 1.58% and Δ_{GE} to 0.45%, although selection remains 7.93%. Model scale therefore helps feasibility, whereas executed-list reward supplies the alignment needed for GE contribution.

Curriculum and routed OPD. Relative to direct RL at matched 4B scale, Curriculum RL keeps validity unchanged (98.85% versus 98.89%) while raising selection from 7.93% to 11.27% and Δ_{GE} from 0.45% to 0.57%. Its pCTR and pIPV decrease while pCVR and pGMV increase, illustrating that the curriculum optimizes joint list utility rather than simultaneous pointwise dominance.

The routed 0.8B Student is 98.03% valid and obtains the highest selection (16.24%) and incremental lift (0.73%), while ranking second in Evaluator lift (2.53%) and beat-Base (62.60%). It exceeds both 4B training variants on the two GE contribution metrics despite its smaller serving size. NAR has slightly higher standalone Evaluator lift, but the Student’s leading Δ_{GE} shows greater marginal complementarity to the incumbent pool. Together, the results show that executable strategy generation adds incremental list value (RQ1), scale and replay learning play distinct roles (RQ2), and routed OPD transfers complementary Teacher behavior into the deployable Student (RQ3).

6.4 Ablation of Strategy-Collapse Mitigation

We use ranking_weight_boost as a representative strategy. It assigns one value in $\{-2, -1, 0, 1, 2\}$ to each of CTR, IPV, CVR, and GMV, giving 625 complete tuples. Figure 3 applies Eq. 23 to measure concentration among valid outputs. We interpret it with validity and GE utility in Table 1, so a lower frequency cannot be attributed to failed execution.

Scale control. Direct RL concentrates 59.9% of 0.8B outputs on (0, 2, -1, -1) and 76.8% of 4B outputs on (1, 2, 0, 0). Thus, added

Table 2: Seven-day online A/B test. Bold values are statistically significant.

Exposure PV ↑	Click PV ↑	IPV ↑	Transaction Count ↑	Transaction Amount ↑	Ad Cost ↑
+1.49%	+2.11%	+3.12%	-0.24%	+2.83%	+0.87%

capacity alone does not prevent collapse under the same Evaluator-derived reward.

Curriculum ablation. At 4B, the self-competitive curriculum reduces Top-1 frequency from 76.8% to 41.2% while validity remains unchanged (98.85% versus 98.89%). Selection rises from 7.93% to 11.27% and Δ_{GE} from 0.45% to 0.57%, indicating broader request-conditioned choices without sacrificing GE contribution.

Routed-OPD ablation. At 0.8B, routed reward-augmented OPD lowers Top-1 frequency from 59.9% to 33.0%. Relative to direct RL, validity rises from 97.12% to 98.03%, selection from 8.56% to 16.24%, and Δ_{GE} from 0.38% to 0.73%. The matched-scale ablations therefore show that curriculum competition and routed OPD both mitigate collapse while improving list-level utility.

6.5 Online A/B Test

We conduct a seven-day online A/B test in July 2026. Users are assigned to stable control and treatment buckets via a deterministic user-level hash. Each bucket represents 1% of traffic from a large-scale production service and contains millions of users. A preceding A/A test finds no material imbalance. Outcomes are normalized per user. Ad cost denotes advertising spend and is therefore a positive platform commercial-value metric.

The control retains the incumbent GE pool, while the treatment adds the final routed reward-augmented OPD 0.8B Generator with all other ranking components and eligibility rules fixed. MetaStrategy wins 27.93% of treatment-side atomic GE calls. Table 2 shows gains of 1.49% in exposure PV, 2.11% in click PV, 3.12% in IPV, 2.83% in transaction amount, and 0.87% in ad cost; transaction count changes by -0.24%. The platform’s standard procedure finds the gains in exposure PV, click PV, IPV, and transaction amount significant. Transaction count and ad cost are not statistically significant, although the ad-cost direction is positive for platform commercial value. Deployed on this recommendation surface, MetaStrategy serves hundreds of millions of users daily, demonstrating its effectiveness in real-world production.

7 Conclusion

MetaStrategy reframes generative ranking as request-conditioned, schema-bounded executable strategy generation. Its typed policy and compiler instantiate an isolated Generator that competes under the incumbent Evaluator. Production-path replay, self-competitive curriculum, and Evaluator-routed reward-augmented OPD yield a deployable 0.8B Student with +0.73% incremental GE lift offline. In a seven-day online A/B test, click PV, IPV, and transaction amount increase by 2.11%, 3.12%, and 2.83%, respectively, with no added ranking RT. These results establish a practical LLM strategy layer for personalized ranking. Future work will study robustness to Evaluator updates, broader typed action spaces, and long-term user feedback beyond fixed-candidate replay.

References

- [1] Himan Abdollahpour, Robin Burke, and Bamshad Mobasher. 2019. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555* (2019).
- [2] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=3zKtaqxLhW>
- [3] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2Slate: Re-ranking and slate optimization with RNNs. *arXiv preprint arXiv:1810.02019* (2018).
- [4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.
- [5] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084* (2022).
- [6] Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. 2021. GRN: Generative Rerank Network for Context-wise Recommendation. *arXiv preprint arXiv:2104.00860* (2021).
- [7] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*. 299–315.
- [8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [9] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European conference on information retrieval*. Springer, 364–381.
- [10] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 368–377.
- [11] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets.. In *IJCAI*, Vol. 19. 2592–2599.
- [12] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems—A survey. *Knowledge-based systems* 123 (2017), 154–162.
- [13] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2024. Large language models for generative recommendation: A survey and visionary discussions. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*. 10146–10159.
- [14] Yi Li, Jieming Zhu, Weiwen Liu, Liangcai Su, Guohao Cai, Qi Zhang, Ruiming Tang, Xi Xiao, and Xiuqiang He. 2022. Pear: Personalized re-ranking with contextualized transformer for recommendation. In *Companion Proceedings of the Web Conference 2022*. 62–66.
- [15] Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural re-ranking in multi-stage recommender systems: A review. *arXiv preprint arXiv:2202.06602* (2022).
- [16] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.
- [17] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.
- [18] Yue Meng, Cheng Guo, Yi Cao, Tong Liu, and Bo Zheng. 2025. A generative re-ranking model for list-level multi-objective optimization at taobao. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 4213–4218.
- [19] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.
- [20] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- [21] Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. 2024. Non-autoregressive generative models for reranking recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5625–5634.
- [22] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860* (2023).
- [23] Junwei Xu, Zhibo Xiao, Chuxin Chen, Chengyu Lai, Qijie Shen, Jiuning Lin, Dimin Wang, Jialin Zhu, and Xiao-Ping Zhang. 2026. OMGRec: One-time Matching-based Generative Rerank with Permutation-level Modeling in E-commerce. In *Proceedings of the ACM Web Conference 2026*. 8421–8424.
- [24] Yue Xu, Hao Chen, Zefan Wang, Jianwen Yin, Qijie Shen, Dimin Wang, Feiran Huang, Lixiang Lai, Tao Zhuang, Junfeng Ge, et al. 2023. Multi-factor sequential re-ranking with perception-aware diversification. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 5327–5337.
- [25] Yue Xu, Qijie Shen, Jianwen Yin, Zengde Deng, Dimin Wang, Hao Chen, Lixiang Lai, Tao Zhuang, and Junfeng Ge. 2023. Multi-channel Integrated Recommendation with Exposure Constraints. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5338–5349.
- [26] Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. 2025. Onerec technical report. *arXiv preprint arXiv:2506.13695* (2025).

A Complete Strategy Action Schema

The action schema is the contract between the language policy and the production re-ranking system. It intentionally excludes item identifiers, free-form code, and direct permutations. Instead, the policy must emit a complete bundle of typed, bounded decisions whose semantics are stable across offline training, production-path replay, nearline generation, and online execution. The ordinal values in Table 3 express relative preferences or adjustment strengths; they do not replace the incumbent fine-ranking scores or bypass hard business and experience constraints.

The five modules expose complementary controls. Weight adjustments alter the balance among engagement and transaction targets; card-type and category preferences control supply composition; experience constraints regulate exposure, purchase, and density patterns; and top-CTR switches activate audited page-specific templates. Modules are interpreted jointly because a valid strategy may combine several adjustments for one request. Every output must include the required modules in the prescribed order and satisfy field types, enumerations, and value ranges. Unknown, missing, or out-of-range fields cause the bundle to be rejected.

After validation, a versioned compiler maps the semantic actions to executable production parameters. The downstream resolver then applies eligibility checks and hard constraints before list construction. This separation keeps the LLM action auditable and permits the concrete fields or bounds to evolve through a matched schema-and-compiler update without changing the learning formulation. Table 3 reports the schema used in the current deployment.

Table 3: Structured strategy space in the current deployment. The method requires typed executable modules; the particular fields are configurable.

Module	Decision and domain
ranking_weight_boost	CTR, IPV, CVR, and GMV adjustments; $\{-2, -1, 0, 1, 2\}$ per field.
cardtype_preference	Preferences for auction, advertising, promotion, video, live, and professional content; $\{-2, -1, 0, 1, 2\}$ per field.
category_preference	Preference strength for eligible categories; $\{-2, -1, 0, 1, 2\}$ per key.
experience_constraints	Exposure, purchase, and density constraints; $\{-2, -1, 0, 1, 2\}$ per field.
top_ctr_strategy	Page-specific top-CTR and reverse-order switches; $\{0, 1\}$ per page group.

B Detailed Production Deployment

B.1 Request Trigger and Context Fork

Every ranking request creates two logically decoupled paths. The synchronous branch proceeds directly to the online ranker. In parallel, the nearline branch assembles a strategy-relevant context h_t from intent and behavior, candidate and supply summaries, and active business constraints. The behavior signals combine the upstream intent model’s real-time output, historical user statistics, and the ordered sequence of recent interactions. The synchronous branch uses the latest available entry, while a refreshed strategy becomes visible only after publication to the online table.

B.2 Diff-Triggered Generation

Let $\bar{h}_t$ denote the context associated with the latest valid table entry. The nearline service invokes the deployed Student when no valid entry exists or when

$$\Delta(h_t, \bar{h}_t) > \tau, \quad (24)$$

where Δ measures feature-aware changes in strategy inputs. Otherwise, the service reuses the current entry. On refresh, the validator and compiler enforce schema and field-level bounds before translating the JSON bundle into executable ranking parameters. Only a valid configuration is published. A failed refresh does not replace the latest valid entry; when no usable entry is available, synchronous serving selects the production default.

B.3 Strategy Resolution and Isolation

For each ranking request, a deterministic resolver reads the latest table entry and the production default, checks validity, and returns an executable configuration. A valid personalized entry takes precedence; an invalid, missing, or expired entry selects the default. The resolved configuration is attached only to the MetaStrategy Generator. No incumbent Generator, Evaluator, or fallback branch is modified, and the MetaStrategy list is exposed only when it wins the atomic GE comparison. The system therefore supports gradual traffic ramp-up, request-level attribution, and immediate rollback without on-request model updates.

C Limitations and Responsible Deployment

MetaStrategy optimizes a learned Evaluator rather than online value directly. Self-competition raises the cost of exploiting frequent strategy modes, but it cannot remove Evaluator misspecification. The Evaluator and other production models can also change across online releases. Atomic replay makes Generator scores comparable within a request, not across replay invocations or model versions.

Replay fixes the logged candidate set and its fine-ranking scores, then executes the current production re-ranking path. It therefore cannot evaluate gains that require different retrieval candidates or upstream fine-ranking predictions. Nor does it simulate clicks, transactions, or later user-state transitions. Replay results measure list-level proxy utility within the support of the logged request and must be validated by randomized online experiments.

The typed action schema further limits the policy to mechanisms exposed by the production ranker. This restriction improves auditability and rollback, but it may exclude useful strategies. Generated configurations can alter commercial exposure and content composition, so deployment requires field-level bounds, fallback behavior, traffic ramp-up, segment analysis, and continuous user experience monitoring.