MOGR: MULTI-OBJECTIVE GENERATIVE RECOMMENDATION VIA OBJECTIVE-CONDITIONED DISTILLATION AND LISTWISE GENERATIVE POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation has emerged as a promising paradigm that generates ranked lists in a single autoregressive pass. However, user behaviors such as reads, clicks, and deeper engagement reflect distinct dimensions of user value, and this single-pass design typically forces all objectives into one merged training signal, which exposes inter-objective conflicts: improving one objective often degrades another. To address this, we propose MOGR (Multi-Objective Generative Recommendation), a single-decoder generator that emits, for each list position, a task token declaring which objective the item serves, then generates its Semantic IDs (SIDs). We train MOGR with Objective-Conditioned Distillation (OCD) and Listwise Generative Policy Optimization (LGPO). OCD distills per-objective specialists into one shared decoder under matching objective labels, so each objective keeps its own supervision. LGPO optimizes complete lists under a concave reward, tuning which objective each ranking position serves—a decision OCD alone cannot make. We further introduce OBRQ, an objective-aware codebook variant for industrial logs. On Kwai26, MOGR achieves the best Recall@512 on all three objectives, with 4.3% minimum relative gain (Min-Gain) over MD-CBS and 20.6% over TIGER, and LGPO is compared with a reward-model proxy. Online, OCD+LGPO improves the online Min-Gain over three live objectives relative to frozen OCD by 0.218% under matched exploration ($p < 0.05$), and by 0.258% under deployed exploration; a separate A/B test gives a 0.524% Min-Gain over the production cascade.

1 INTRODUCTION

Generative recommendation represents items as *Semantic IDs* (SIDs) and trains autoregressive models to directly generate a ranked recommendation list Jiang et al. (2026). These models capture richer inter-item dependencies at the list level and benefit from the scaling properties of large sequence models Zhai et al. (2024); Kong et al. (2025), while semantic-token representations themselves unlock scaling in ranking architectures Zhao et al. (2026). Generating the whole list as a single sequence, however, creates a tension when multiple user objectives must be optimized simultaneously: in the standard single-pass formulation, all objectives are forced into one merged training signal, so improving one objective often degrades another. This tension arises in any generative recommendation model that must satisfy multiple non-exclusive objectives.

We study this problem on *large-scale content recommendation platforms*, where one exposure can trigger several non-exclusive behaviors at once. On the public Kwai26 benchmark Wang et al. (2026b), we study the $K = 3$ binary feedback signals, instantiated here as *Long-View*, *Like*, and *Forward* (share). Online, the live industrial objectives are a *read* (*Read*), a *POI click* to the location page (*Poi*), and a deeper *interaction* such as comment or share (*Int*). Both the public Kwai26 signals and the live industrial objectives span consumption and conversion, the two core dimensions of user value we target. Serving all of them falls to a single generated list. User intents can shift within a session Zhang et al. (2025), so the desirable objective mixture varies across a user's behavior chain;

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.

a fixed mixture cannot capture this shift, motivating position-level allocation learned from online feedback. A merged training signal instead gives every position the same blend.

The single-pass design fails this task in three ways. In training, one generation objective aggregates every user objective, so the decoder cannot condition on a chosen objective. In composition, which objective each list position serves is neither modeled nor optimized. Content-only quantization maps items with distinct behaviors to the same or adjacent SIDs, so the token space itself is objective-agnostic. Multi-task rankers and current generative methods leave this mixture outside a single deployed decoder (Section 2). Item-level generation for a chosen objective is scored offline. LGPO is evaluated with a Kwai26 reward-model proxy and a live A/B (Section 4.1).

We propose **MOGR** (**M**ulti-**O**bjective **G**enerative **R**ecommendation). MOGR makes the objective an explicit part of generation: a *single-decoder generator* emits for each item in the list a task token declaring which objective that item serves, then generates its SIDs. With the objective explicit, the two failures decouple, and the generator is trained through two complementary stages. *Objective-Conditioned Distillation* (OCD) answers the training failure: we train one specialist per objective on logged exposures, and then distill these specialists into the generator under their corresponding objective labels. As a result, a single shared decoder serves all objectives while each objective retains its own supervised signal. **Listwise Generative Policy Optimization** (LGPO) answers the composition failure: it optimizes complete lists under a concave reward, deciding which objective each ranking position should serve, a decision that distillation alone cannot make. **Objective-Block Residual Quantization** (OBRQ) further separates objectives in the SIDs.

Our contributions are summarized as follows:

- **Objective specialization and transfer (OCD).** We train a decoder and K objective specialists on logged exposures, then distill them into a single-decoder generator: OCD transfers each specialist under its own objective condition (Section 3.3). This retains objective-specific supervision while keeping serving to a single decoder instead of $K + 1$ decoders. We evaluate the resulting item coverage on Kwai26 Recall@512 for Long-View, Like, and Forward.
- **List-level objective composition (LGPO).** Given the objective-conditioned generator, LGPO optimizes the placement of objective conditions across positions under a concave list reward, instead of fixing their mixture through decoding quotas (Section 3.4).
- **Objective-block SID construction (OBRQ).** We observe that content-only quantization can leave objectives aliased in the codebook, and propose Objective-Block Residual Quantization (OBRQ) to residual-quantize equal-energy content and per-objective collaborative filtering blocks with parent-cluster energy gates, mitigating aliasing in the codebook (Section 3.5).
- **Offline, online, and industrial validation.** We evaluate OCD on Kwai26 with Recall@512, LGPO on a live A/B and on a Kwai26 reward-model proxy, and OBRQ as an industrial codebook swap. OCD attains a Min-Gain of 4.3% over MD-CBS and 20.6% over TIGER; LGPO attains a Min-Gain of 0.218% over frozen OCD under matched exploration and 0.524% over the production cascade; OBRQ improves over content-only RQ.

2 Related Work

2.1 Generative Recommendation

TIGER Rajput et al. (2023) pioneered SID-based generative recommendation, encoding items via residual vector quantization and generating recommendations autoregressively. Subsequent efforts scaled the paradigm to large-scale deployments Jiang et al. (2026); Kong et al. (2025), enriched the embeddings underlying SID quantization with collaborative signals Zhao et al. (2026), and diversified training signals via generative flow networks Wang et al. (2026a). RL-based alignment has also been applied: OneRec Deng et al. (2025) and MiniOneRec Kong et al. (2025) adopts GRPO-style group advantages Shao et al. (2024) to spare the value network. OneRec-V2 Zhou et al. (2025) replaces reward-model alignment with real user feedback and gradient-bounded GRPO training. UniVA Zhang et al. (2026) introduces value-aware tokenization for advertising.

These methods still return one scalar per list. LGPO instead optimizes which objective each position serves, and forms user-window groups from consecutive requests rather than samples that share only a query.

Closest to our setting, Multi-Decoder OneRec with Constrained Beam Search(MD-CBS) Wang et al. (2026b) serves K LoRA experts with constrained beam search at inference, whereas MOGR distills their objective-conditioned SID policies into a single-decoder generator. Task- and behavior-conditioned generation appears at the single-item level in P5 Geng et al. (2022) and MBGen Liu et al. (2024); MOGR uses per-slot task tokens to transfer objective specialists into one decoder and optimize their placement from complete-list online feedback. On the tokenization side, TRM Zhao et al. (2026) and UniVA Zhang et al. (2026) encode one collaborative or value signal, whereas OBRQ separates objectives in the vocabulary itself.

2.2 Multi-Objective Recommendation

Multi-task discriminative models optimize several objectives through shared representations and objective-specific heads, from shared-bottom and entire-space designs Ma et al. (2018b) to multi-gate routing Ma et al. (2018a), progressive expert separation Tang et al. (2020), and industrial ranking that balances engagement and satisfaction Zhao et al. (2019). Each candidate is still scored independently of its neighbors, so these heads do not allocate objectives across a generated list.

Multi-objective RL learns a Pareto frontier over scalarization weights Fatima Ezzahra et al. (2025); Li & Tuzhilin (2026); Zhou et al. (2026) or optimizes one fixed scalarization, and slate RL optimizes one list reward Ie et al. (2019); Deffayet et al. (2023). LGPO uses a concave reward related to Nash social welfare Caragiannis et al. (2019) and chooses the objective at each position (Section 3.4).

2.3 Knowledge Distillation and Mixture of Experts

Classical distillation Hinton et al. (2015) transfers knowledge from one teacher to one student through soft targets, including intermediate-layer alignment Romero et al. (2015). Multi-teacher distillation extends this to an ensemble of teachers, typically by aggregating their soft targets You et al. (2017). When teachers disagree—as objective specialists do—such aggregation risks blurring the differences the student is meant to inherit. Unlike conditioning-based multi-teacher KD, where a condition selects or weights which teacher to distill from Kim & Choi (2021), OCD makes the objective label part of the generation target, so the student learns to condition its generation on it and specialization becomes intrinsic to the sequence rather than an input-dependent routing decision. MOGR's multi-expert teacher uses a shared encoder with per-objective decoders, relating to Mixture-of-Experts Shazeer et al. (2017); however, MoE routes tokens through a learned gate, whereas MOGR fixes the objective-to-decoder assignment: specialization is structural, not determined by a router.

3 Methodology

3.1 Problem Formulation and Method Overview

Given a user u and interaction history $\mathcal{H}_u$, the task is to generate a ranked list of N items. Each interaction carries a feedback vector $\mathbf{y} \in \{0,1\}^K$ whose k-th bit is one when objective k was triggered; the K objectives are non-exclusive, so one exposure may set several bits. The target is the list's performance on the K objectives, not a single combined score.

MOGR (Figure 1(b)) deploys a **single-decoder generator** that produces the list in one pass (Section 3.2). It makes the objective an explicit token in the sequence. Each item i is represented by an SID $\mathbf{s}_i = (s_i^1, \ldots, s_i^L) \in [C]^L$, obtained by residual quantization of item embeddings (Section 3.5). The generator emits

$$\mathbf{L}_u = (t_1, \mathbf{s}_1, \ldots, t_N, \mathbf{s}_N), \tag{1}$$

where $j \in \{1, \ldots, N\}$ indexes the list position, and each t_j conditions the following L SID tokens of $\mathbf{s}_j$. SFT labels use $t_j \in \{t_{\text{exp}}, t_1, \ldots, t_K\}$. At inference, t_{exp} is masked, so generated lists satisfy $t_j \in \{t_1, \ldots, t_K\}$. The served object is the N items; the generated task tokens are the internal allocation used to produce them. A training-time **teacher** supplies per-objective item generation and

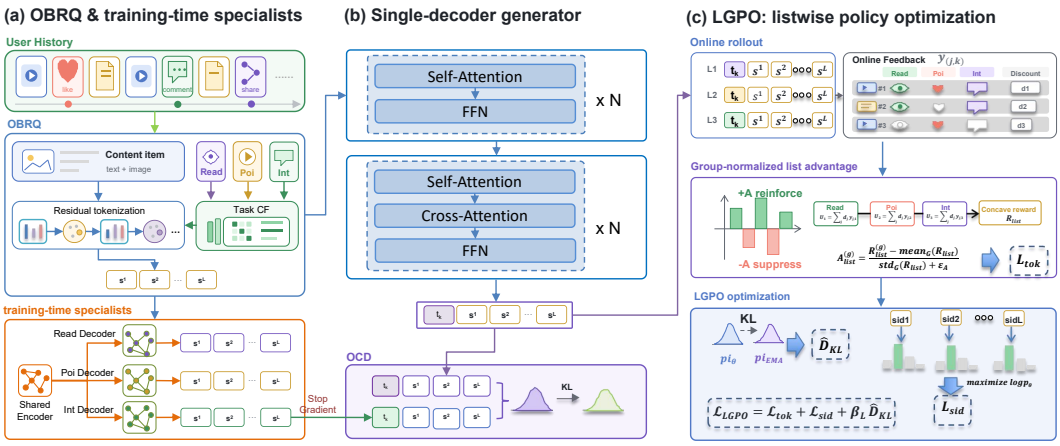

Figure 1: Overview of MOGR. (a) OBRQ and training-time specialists: content and per-objective CF are residual-quantized into SID sequences; objective decoders on a shared encoder learn those policies, and their gradients are stopped before distillation. (b) Single-decoder generator: user history is encoded once by a shared encoder–decoder, and each list slot emits a task token t_k followed by SIDs; OCD distills each specialist into this generator under its own t_k with a KL constraint. (c) LGPO: online feedback on complete lists is converted into a group-normalized list advantage, which updates task-token placement ($\mathcal{L}_{\text{tok}}$) and the subsequent SID tokens ($\mathcal{L}_{\text{sid}}$), regularized by a KL term against an EMA reference.

is distilled into the generator by OCD (Section 3.3). **LGPO** then tunes the placement of task tokens on complete lists using online feedback (Section 3.4). The SID vocabulary is a frozen codebook (Section 3.5). Training follows Algorithm 1.

3.2 The Single-Decoder Generator

The single-decoder generator is the deployed model: an encoder–decoder Transformer with encoder E_S and a single decoder D_S. The encoder encodes $\mathcal{H}_u$ bidirectionally; the decoder generates Eq. (1). Given the memory $\mathbf{M}_u^S = E_S(\mathcal{H}_u)$, the list likelihood is

$$\pi_\theta(\mathbf{L} \mid \mathcal{H}_u) = \prod_{j=1}^{N} \left[p_\theta(t_j \mid \mathbf{L}_{<j}, \mathbf{M}_u^S) \prod_{l=1}^{L} p_\theta(s_j^l \mid t_j, s_j^{<l}, \mathbf{L}_{<j}, \mathbf{M}_u^S) \right], \tag{2}$$

where $\mathbf{L}_{<j} = (t_1, \mathbf{s}_1, \ldots, t_{j-1}, \mathbf{s}_{j-1})$ and $\mathbf{L}_{<1}$ is empty. p_θ denotes the student generator's per-step conditional distribution; $p_{D_{\text{exp}}}$ and $p_{D_T^k}$ denote the corresponding per-step conditionals of the teacher decoders. SFT uses the full task support $\{t_{\text{exp}}, t_1, \ldots, t_K\}$.

3.2.1 Supervised Initialization

The generator is warm-started on every logged exposure in the session, including slots with no bit set. Each slot receives one exclusive task label because the generator emits a single task token t_j: t_{exp} if no bit is set, otherwise the rarest set bit, ranked by positive rate. The rarest-first rule prevents frequent objectives from dominating the task-token distribution and under-representing rare ones. The task-token softmax in Eq. (2) is trained with this exclusive hard label. Each task token $t \in \{t_{\text{exp}}, t_1, \ldots, t_K\}$ has its own input embedding. Let $t^\star$ be the exclusive label of a slot; for slot j, write $t_j^\star$. Then $I_k = \{j : t_j^\star = t_k\}$ and $K^+ = \{k : I_k \neq \varnothing\}$. The SID loss keeps a coverage floor on all slots and adds an exclusive t_k term plus a token-level KL that anchors the SID policy under

t_k to the expose policy:

$$\mathcal{L}_{\mathrm{SFT}} = \mathcal{L}_{\mathrm{CE}}(\mathbf{s} \mid t_{\mathrm{exp}}) + \lambda_{\mathrm{S}} \frac{1}{|K^+|} \sum_{k \in K^+} \mathcal{L}_{\mathrm{CE}}(\mathbf{s} \mid t_k) + \lambda_{\mathrm{task}} \mathcal{L}_{\mathrm{CE}}(t^\star) + \beta_{\mathrm{S}} \mathcal{L}_{\mathrm{KL}},$$
$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{|K^+|} \sum_{k \in K^+} \frac{1}{L} \sum_{l=1}^{L} D_{\mathrm{KL}}\big(\mathrm{sg}\, p_\theta(\cdot \mid t_{\mathrm{exp}}, s^{<l}) \,\big\|\, p_\theta(\cdot \mid t_k, s^{<l})\big), \tag{3}$$

where each $\mathcal{L}_{\mathrm{CE}}(\mathbf{s} \mid t_k)$ and the inner KL are means over I_k, SID CE averages the L codebook layers, and sg denotes stop-gradient.

3.3 Objective-Conditioned Distillation

3.3.1 Training-Time Specialists.

The teacher exists only in training. It is an encoder–decoder Transformer with one shared encoder E_T, an *expose* decoder D_{exp}, and K independent task decoders $\{D_T^k\}_{k=1}^K$ (Figure 1(a)). E_T encodes the mixed-behavior history $\mathcal{H}_u$ once into $\mathbf{M}_u = E_T(\mathcal{H}_u)$. D_{exp} starts from a shared BOS token, uses a tied SID embedding table, and generates the L-token SID of every logged exposure. Its next-token loss is the only signal that updates E_T and the SID table, so the shared representation remains a coverage encoder. Decoder D_T^k has its own task token t_k and output head. It cross-attends to $\mathrm{sg}(\mathbf{M}_u)$ and reads detached SID embeddings from D_{exp}'s table. The teacher is trained on every logged exposure:

$$\mathcal{L}_{\mathrm{T}} = \mathcal{L}_{\mathrm{CE}}(\mathbf{s} \mid D_{\mathrm{exp}}) + \sum_{k=1}^{K} \big(\lambda_{\mathrm{T}} \mathcal{L}_{\mathrm{CE}}^{\eta_k}(\mathbf{s} \mid t_k) + \beta_{\mathrm{T}} \mathcal{L}_{\mathrm{KL}}^k\big),$$
$$\mathcal{L}_{\mathrm{KL}}^k = \frac{1}{L} \sum_{l=1}^{L} D_{\mathrm{KL}}\big(\mathrm{sg}\, p_{D_{\mathrm{exp}}}(\cdot \mid s^{<l}) \,\|\, p_{D_T^k}(\cdot \mid t_k, s^{<l})\big), \tag{4}$$

where $\eta_{i,k} = 1 + \alpha_k y_{i,k}$ reweights the SID CE of D_T^k. D_{exp} fits the coverage SID distribution with uniform CE, and each D_T^k fits objective k with weighted CE. The KL term anchors $p_{D_T^k}$ to the expose distribution.

3.3.2 Distilling into the Generator.

The specialists are stronger per-objective SID generators when the task is held fixed (Table 6). OCD distills each frozen specialist into the generator from the SFT checkpoint of Section 3.2.1, so the list model inherits those SID policies. Each specialist is aligned under its own task token t_k. Samples are logged positive exposures $(\mathcal{H}_u, \mathbf{s}_i, \mathbf{y})$ with $Y = \{k : y_k = 1\}$; empty Y is skipped.

For each $k \in Y$ both models read the same prefix: t_k is forced and the SID is the logged $\mathbf{s}_i$. The teacher runs E_T then D_T^k; the generator runs E_S then D_S. At each SID position they match codebook softmaxes by KL:

$$\mathcal{L}_{\mathrm{OCD}} = \frac{1}{|Y|} \sum_{k \in Y} \frac{1}{L} \sum_{l=1}^{L} D_{\mathrm{KL}}\Big(p_{D_T^k}(\cdot \mid t_k, s^{<l}, \mathbf{M}_u^T) \,\|\, p_{D_S}(\cdot \mid t_k, s^{<l}, \mathbf{M}_u^S)\Big). \tag{5}$$

A multi-label exposure repeats the SID under every t_k for $k \in Y$ and averages those KLs. OCD updates E_S and D_S on SID tokens only; the task-token head stays the SFT classifier of Section 3.2.1. A second term anchors the four-way task-token distribution to the frozen SFT generator on every logged slot, including slots with $Y = \varnothing$. The softmax is restricted to $\{t_{\mathrm{exp}}, t_1, \ldots, t_K\}$, and the SFT probabilities use that checkpoint's own encoder:

$$\mathcal{L}_{\mathrm{task}} = D_{\mathrm{KL}}(\mathrm{sg}\, p_{\mathrm{SFT}}(t \mid \mathrm{BOS}, \mathcal{H}) \,\|\, p_{D_S}(t \mid \mathrm{BOS}, \mathcal{H})). \tag{6}$$

The mean is over valid logged slots. OCD minimizes $\mathcal{L}_{\mathrm{OCD}} + \lambda_{\mathrm{task}} \mathcal{L}_{\mathrm{task}}$ with λ_{task}=0.1. Task-token embeddings, E_S, and D_S all receive this gradient.

3.4 LISTWISE GENERATIVE POLICY OPTIMIZATION

LGPO composes N items into one list and is the only stage that deploys the generator: it serves complete lists, collects real feedback, and updates on those generations. OCD anchors item-level SID policies (Section 3.3); LGPO's primary target is the placement of generated task tokens.

3.4.1 ONLINE ROLLOUT AND FEEDBACK COLLECTION.

Serving uses a decoding procedure μ_θ, not softmax sampling from Eq. (2). The task-token softmax is restricted to $\{t_1, \ldots, t_K\}$: t_{exp} is masked and does not appear in served lists. SID tokens are greedy under the untempered codebook softmax. One uniformly random slot per list samples its task token from the restricted softmax; all other task tokens are greedy. LGPO updates on lists drawn from μ_θ. The platform records $y_{j,k} \in \{0, 1\}$ at every rank.

A *user-window group* is the lists served to the same user over G_L consecutive requests. The group mean is a user-level baseline. Later requests include newly consumed items, so the histories in the window are not identical. Unlike GRPO, the G_L lists in a user-window group do not share the same prompt, so the group-mean baseline absorbs some state drift. We group by user to remove cross-user heterogeneity, and online serving cannot sample multiple lists for the same request without degrading the user experience; the KL trust region to π_{ema} further limits drift within the window. A group is finalized when all G_L feedbacks have arrived; each list is scored with the snapshot θ_{old} that served it. All-zero groups are skipped.

3.4.2 CONCAVE MULTI-OBJECTIVE LIST REWARD.

With rank discount $d_j = 1/\log_2(j+1)$, the list utility of objective k is $U_k(\mathbf{L}^{(g)}) = \frac{1}{Z_k} \sum_{j=1}^{N} d_j y_{j,k}^{(g)}$, where Z_k is the mean discounted list utility of objective k estimated from historical logs and held fixed throughout LGPO optimization. The list reward is

$$R_{\text{list}}(\mathbf{L}^{(g)}) = \sum_{k=1}^{K} w_k \log\Big(1 + U_k(\mathbf{L}^{(g)})\Big), \tag{7}$$

with $w_k \geq 0$ and $\sum_k w_k = 1$. The logarithm is applied per objective, so extra gains on an already well-served objective have diminishing returns. We optimize the concave log reward rather than max-min because max-min gradients concentrate on the worst objective.

3.4.3 TASK-TOKEN CLIPPED SURROGATE

The list advantage standardizes Eq. (7) within the group:

$$A_{\text{list}}^{(g)} = \frac{R_{\text{list}}(\mathbf{L}^{(g)}) - \text{mean}\Big(\{R_{\text{list}}(\mathbf{L}^{(g')})\}_{g'=1}^{G_L}\Big)}{\text{std}\Big(\{R_{\text{list}}(\mathbf{L}^{(g')})\}_{g'=1}^{G_L}\Big) + \varepsilon_{\text{A}}}, \tag{8}$$

where $\varepsilon_{\text{A}} > 0$ is a small constant. Lists are drawn from $\mu_{\theta_{\text{old}}}$.

The update forms its trust region on the *task-token policy* alone. Let $j^\star$ be the exploration slot of μ_θ (Section 3.4.1). At $j^\star$, the sampled task token is $t_{j^\star}$, and the behavior policy is that restricted task softmax, giving the ratio $\rho^{(g)}(\theta) = \frac{p_\theta(t_{j^\star}^{(g)} | \mathbf{L}_{<j^\star}^{(g)}, \mathbf{M}_u^S)}{p_{\theta_{\text{old}}}(t_{j^\star}^{(g)} | \mathbf{L}_{<j^\star}^{(g)}, \mathbf{M}_u^S)}$. The clipped surrogate on this ratio is

$$\mathcal{L}_{\text{tok}}(\theta) = -\frac{1}{G_L} \sum_{g=1}^{G_L} \min\Big(\rho^{(g)}(\theta)\, A_{\text{list}}^{(g)},\ \text{clip}\Big(\rho^{(g)}(\theta), 1-\delta, 1+\delta\Big)\, A_{\text{list}}^{(g)}\Big), \tag{9}$$

where $\delta > 0$ is the clip range. Because the ratio is formed under the restricted task softmax, Eq. (9) is a PPO-style clipped surrogate at the sampled slot.

We additionally train the SID tokens with an auxiliary negative log-likelihood loss:

$$\mathcal{L}_{\text{sid}}(\theta) = -\lambda_{\text{sid}} \frac{1}{G_L} \sum_{g=1}^{G_L} \sum_{j=1}^{N} \sum_{l=1}^{L} \log p_\theta\Big(s_j^{l,(g)} \mid t_j^{(g)}, s_j^{<l,(g)}, \mathbf{L}_{<j}^{(g)}, \mathcal{H}_u\Big), \tag{10}$$

Algorithm 1 MOGR training pipeline

1: Construct and freeze the SID codebook (content residual K-means on Kwai26; OBRQ on the industrial log) from offline logs.
2: Train D_{exp} and $\{D_T^k\}$ on logged exposures (Eq. (4)); only D_{exp} updates E_T and the SID table.
3: Warm-start the generator with exclusive-task SFT on logged exposures ($\mathcal{L}_{\text{SFT}}$, Eq. (3)).
4: Freeze all teachers and the SFT generator; distill SID policies into the generator on positive logged exposures ($\mathcal{L}_{\text{OCD}}$, Eq. (5)) and anchor task-token logits to SFT on every logged slot ($\mathcal{L}_{\text{task}}$, Eq. (6)).
5: Optimize complete lists with LGPO from online rollout and feedback ($\mathcal{L}_{\text{LGPO}}$).

This term is excluded from the probability ratio and clipping. Because LGPO updates only the sampled task token, we keep a small λ_{sid} to anchor the SID policy and limit drift from the OCD initialization. The overall objective is $\mathcal{L}_{\text{LGPO}}(\theta) = \mathcal{L}_{\text{tok}}(\theta) + \mathcal{L}_{\text{sid}}(\theta) + \beta_L \widehat{D}_{\text{KL}}$, where $\widehat{D}_{\text{KL}}$ is the tokenwise, teacher-forced $D_{\text{KL}}(\pi_\theta \| \pi_{\text{ema}})$ on the served sequence (restricted task tokens and untempered SID tokens), π_{ema} is an exponential moving average of the generator, and $\beta_L \geq 0$ controls the KL strength.

3.5 Objective-Block Residual Quantization

The generator and the teacher share the SID vocabulary of Eq. (2). Content-only residual quantization Lee et al. (2022); Rajput et al. (2023) groups items by embedding similarity, so two items with similar content but different objective rates can receive the same SID.

Objective-Block Residual Quantization (OBRQ) puts content and one collaborative embedding per objective as separate blocks of one residual codebook, and reweights later layers by each parent cluster's leftover residual energy (Figure 1(a)). The codebook is built once from logs and frozen.

A frozen content vector $\mathbf{z}_i$ is obtained by contrastively aligning a BERT encoding of the item title and body Devlin et al. (2019) with a Vision Transformer encoding of the item image Dosovitskiy et al. (2021); the aligned vector is stored and reused for quantization. For each objective k, a CF item tower trained on that objective produces $\mathbf{e}_i^k$. Each block is L2-normalized and concatenated, $\mathbf{x}_i = [\text{L2}(\mathbf{z}_i); \text{L2}(\mathbf{e}_i^1); \ldots; \text{L2}(\mathbf{e}_i^K)]$.

Let $\mathbf{r}_i^{(1)} = \mathbf{x}_i$ and write $\mathbf{r}_{i,m}^{(\ell)}$ for block $m \in \{0, \ldots, K\}$ at layer ℓ (m=0 is content). Layer 1 is ordinary residual K-means with C centroids on $\mathbf{r}^{(1)}$. At layer $\ell \geq 2$, items that share the previous-layer code c form a parent $\mathcal{C}_c$. The energy share of block m in that parent is $w_{c,m} = \frac{\sum_{i \in \mathcal{C}_c} \|\mathbf{r}_{i,m}^{(\ell)}\|^2}{\sum_{m'} \sum_{i \in \mathcal{C}_c} \|\mathbf{r}_{i,m'}^{(\ell)}\|^2}$. Assignment uses the scaled residual $\tilde{\mathbf{r}}_{i,m}^{(\ell)} = \sqrt{w_{c,m}}\, \mathbf{r}_{i,m}^{(\ell)}$; centroids are means in that scaled space, then the residual is unscaled back to the original blocks. A parent whose residual energy is dominated by objective k therefore spends the next layer on that objective's block.

The SID of item i is $(c_i^{(1)}, \ldots, c_i^{(L)})$, from L-layer balanced residual K-means with C centroids: each centroid is assigned at most a fixed multiple of n/C items.

4 Experiments

4.1 Experimental Setup

Kwai26 is a public multi-objective generative-retrieval benchmark Wang et al. (2026b) (Appendix Table 3): 50,000 test sessions, 25.03M items with valid SIDs, and the official $3\times8{,}192$ lexicon. A positive is a target-slot item that triggered k. On these sessions, Long-View, Like, and Forward occupy 21.92%, 5.16%, and 0.79% of the 191,335 target slots, so SFT exclusive labels use Forward $\succ$ Like $\succ$ Long-View; 1,349 sessions (2.70%) contain at least one Forward positive. The industrial objectives are Read, Poi, and Int, ordered Int $\succ$ Poi $\succ$ Read by the same positive-rate rule. Kwai26 items carry only an item ID and an official SID, with no content vector, so OBRQ is not built there and Table 1 uses the official lexicon. Codebook variants are fit on a sampled dump and each retrains teacher and generator (Appendix Table 4; Section 4.3).

On Kwai26, all methods are compared on one mixed pool $C(u; 512)$, scored with micro-averaged Recall@512. TIGER Rajput et al. (2023) and OneRec-V2 Zhou et al. (2025) are single-decoder SID models trained with next-token prediction on the official target slots—one merged exposure signal—and each decodes on this pool. MD-CBS Wang et al. (2026b) trains per-objective experts and merges them with the quota schedule from the original paper (Appendix A). DSSM Huang et al. (2013) is a two-tower retriever trained with exposure InfoNCE on the same official targets and returns Top-512 from the item catalog using only the item ID. MOGR emits a task token then an SID per slot in one constrained beam. For specialist diagnostics (Appendix Table 6), we decode a dedicated pool of 512 per objective: teacher specialists use D_T^k, while the MOGR generator forces the corresponding task token t_k. Online, the displayed feed has $N{=}20$ items. Frozen OCD is greedy; OCD+LGPO samples one task-token slot per list (Section 3.4). TIGER, OneRec-V2, and the MOGR generator share $d_{\text{model}}{=}256$, $4{+}4$ layers, and 8 heads; MD-CBS adds per-objective adapters; DSSM uses a 128-d two-tower. Remaining hyperparameters are in Appendix A. Min-Gain is the conservative relative lift versus a named baseline, $\min_k \frac{R_k - R_k^{\text{base}}}{R_k^{\text{base}}}$. On Kwai26 the baseline is TIGER. OBRQ uses the same hit rule on Read / Poi / Int.

The Kwai26 log has no feedback on lists the generator emits and no propensity, so Appendix Table 9 compares LGPO with a reward-model proxy rather than IPS or a doubly robust estimator. SFT and OCD for this proxy use only the first 55,000,000 training sessions. Reward models A and B train from scratch on two later slices, each with its own encoder and SID embedding table. Those slices do not overlap each other, the OCD sessions, or the LGPO sessions. LGPO updates the OCD checkpoint on a disjoint slice of 5,000,000 sessions under model A's reward (Section 3.4); both policies are scored by model B, with each binary $y_{j,k}$ replaced by B's probability. Training samples one task-token slot per list; evaluation is fully greedy. The remaining protocol is in Appendix B.

4.2 Main Results

Table 1 reports Recall@512 after SFT and OCD on one merged pool. MOGR (SFT+OCD) attains the highest Recall on every objective (0.193 / 0.139 / 0.145), a Min-Gain of 20.6% versus TIGER. OneRec-V2 reaches Min-Gain 8.8%. MD-CBS is the closest mixed-pool baseline (0.177 / 0.129 / 0.139, Min-Gain 10.6%); the Forward margin is 0.006, with larger gaps on Like and Long-View. DSSM retrieves Top-512 over IDs. SID methods decode a 512-item pool from the lexicon, which is the richer item representation, and count a hit if the gold item or its SID appears.

Table 1: Kwai26 Recall@512 on one merged pool $C(u; 512)$, official lexicon. Min-R is $\min_k R_k$. Min-G is the minimum relative lift versus TIGER (%). MD-CBS uses paper quotas. *w/o* OCD is generator SFT.

Method	Long-View	Like	Forward	Min-R ↑	Min-G (%) ↑
DSSM	0.050	0.034	0.031	0.031	–
TIGER	0.160	0.113	0.116	0.113	0.0
OneRec-V2	0.175	0.123	0.134	0.123	8.8
MD-CBS	0.177	0.129	0.139	0.129	10.6
MOGR (SFT+OCD)	**0.193**	**0.139**	**0.145**	**0.139**	**20.6**
w/o OCD	0.175	0.127	0.140	0.127	9.4

4.3 Ablation Studies

Dropping OCD cuts Min-Gain from 20.6% to 9.4% on the same mixed pool, with Recall falling on all three objectives (0.193 → 0.175, 0.139 → 0.127, 0.145 → 0.140; Table 1). Under forced t_k, OCD moves the generator toward the teacher specialists (Appendix Table 6: Long-View 0.182 → 0.197, Like 0.133 → 0.141, Forward 0.143 → 0.147). Width $d{=}384$ is strongest on all three objectives (0.209/0.151/0.152; Appendix Figure 2). Table 1 uses the shared width $d{=}256$. Appendix Figure 3 varies each specialist's α_k independently.

Each industrial codebook retrains teacher and generator (Appendix Table 5). Content RQ residual-quantizes the frozen 128-d content embedding. Concat RQ concatenates L2-normalized content and three CF towers and runs ordinary residual K-means; OBRQ adds the parent-cluster energy

reweighting of Section 3.5. Adding the CF blocks raises Min-Gain from 0 to 5.1%; the energy reweighting raises it further to 9.6% (Int $0.165 \rightarrow 0.172$). Lower $\bar{V}^k$ in Appendix Table 8 means items that share a prefix SID have more homogeneous objective rates. Appendix Table 5 forces t_k and decodes a dedicated pool of 512, asking whether that lexicon can retrieve the named objective. Appendix Table 5 reports both merged-pool and forced-t_k Recall@512 under each codebook. $\bar{V}^k$ and both Recall settings improve from Content RQ to Concat RQ to OBRQ.

4.4 Online A/B Evaluation

LGPO is trained on the OCD+LGPO arm's own user bucket. Live objectives are Read, Poi, and Int. Randomization is user-level over tens of millions of users. The A/B runs for 7 days with all arms concurrent. The production cascade receives 60% of users; OCD, OCD+LGPO, R_{lin}, and $\lambda_{\text{sid}}{=}1$ each receive 10%. Min-Gain versus the concurrent OCD arm uses that greedy arm as the baseline; OCD+LGPO uses one sampled task-token slot per list (Section 3.4). The default user-window size is $G_L{=}8$. $G_L{=}4/12$ are ablated in a later period that also runs $G_L{=}8$ and frozen OCD (Appendix Figure 5). Table 2 reports lifts versus the production cascade. Slot-level allocation is in Appendix Figure 4. Versus production, OCD+LGPO lifts Read, Poi, and Int by 0.656%, 0.524%, and 0.704% (Min-Gain 0.524%; user-level p $0.015/0.023/0.032$); lifts versus frozen OCD are in Appendix Table 7. A linear sum cuts Min-Gain to 0.097%; increasing the SID auxiliary weight to $\lambda_{\text{sid}} = 1$ reduces Min-Gain to 0.003%. LGPO shifts slot share from Read ($81.19 \rightarrow 73.35$) toward Poi ($17.61 \rightarrow 22.26$) and Int ($1.20 \rightarrow 4.39$). In this main A/B (7 days; four 10% buckets), OCD+LGPO improves Read, Poi, and Int by 0.302%, 0.258%, and 0.299%, respectively, over greedy frozen OCD. An additional concurrent A/B test compares OCD+LGPO with frozen OCD under identical exploration; Table 7 reports their direct relative lifts.

Table 2: Relative lifts (%) versus the production cascade. Each cell is lift / user-level p. Min-G is $\min_k$ lift versus production.

Arm	Read	Poi	Int	Min-G ↑
Production	0.000	0.000	0.000	0.000
OCD	0.352 / 0.042	0.265 / 0.055	0.403 / 0.073	0.265
OCD+LGPO	**0.656** / 0.015	**0.524** / 0.023	**0.704** / 0.032	**0.524**

5 Conclusion

We presented MOGR, a multi-objective generative recommendation framework that addresses inter-objective conflicts. A single-decoder generator emits a task-token/SID list, trained by OCD and LGPO. On Kwai26, OCD reaches a Min-Gain of 20.6% versus TIGER. Online, OCD+LGPO attains a Min-Gain of 0.258% versus concurrent OCD and 0.524% versus the production cascade.

Limitations

Kwai26 Recall@512 is scored on one merged pool; Forward uses the Stage 1 share flag aligned onto official slots. The public split has no on-policy feedback on lists the model itself serves. Section 4.1 therefore trains and scores LGPO with two reward models trained from scratch on disjoint sessions. The reported utility is an expected probability under model B, not live y, and each group is several lists from one history rather than consecutive requests from one user. We do not report an off-policy estimator or a user simulator. Industrial logs and the live A/B cannot be released. A request displays one list, so live y is recorded for that list only. LGPO groups consecutive requests from the same user rather than several samples of one request. Later requests in the window include newly consumed items. Generator SFT keeps one exclusive task label per slot, so a multi-label exposure keeps only one t_k. OCD then distills each objective under its own token. The sampled-slot ratio does not represent the full serving-policy ratio. The reward weights w_k and the teacher up-weights α_k are set by hand. A new objective adds one task token; teacher training uses $K{+}1$ decoders.

REPRODUCIBILITY STATEMENT

Public tables use the official Kwai26 split, lexicon, and 512-candidate protocol Wang et al. (2026b). Hyperparameters, MD-CBS quotas, and OBRQ codebook details are in Appendix A. The Kwai26 LGPO proxy is specified in Appendix B. Code for the public pipelines will be released upon acceptance.

AI USE STATEMENT

We used generative AI tools to edit the manuscript for clarity, compress text to the page limit, and check LaTeX layout, and to assist with implementation (training and evaluation scripts). We did not use generative AI tools to generate synthetic data, develop theoretical models, formulate mathematical claims, design the method or experiments, or interpret results; the remaining required disclosure tasks are not applicable. AI-assisted wording was reviewed by the authors. AI-assisted code was run and checked by the authors against the reported metrics. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses anonymized user interaction logs from a large-scale content recommendation platform, in compliance with the platform's data governance policies and privacy regulations. No personally identifiable information is used or disclosed.

REFERENCES

Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019.

Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten De Rijke. Generative slate recommendation with reinforcement learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 580–588, 2023.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Zaizi Fatima Ezzahra, Abakarim Sana, Qassimi Sara, and Rakrak Said. Multi-objective reinforcement learning for recommender systems: a comprehensive survey of methods, challenges, and future directions: Z. fatima ezzahra et al. *International Journal of Multimedia Information Retrieval*, 14(4):33, 2025.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, pp. 299–315, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392785. doi: 10.1145/3523227.3546767. URL https://doi.org/10.1145/3523227.3546767.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 2333–2338, 2013.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In *IJCAI*, volume 19, pp. 2592–2599, 2019.

Jie Jiang, Jun Zhang, Yi Li, Yue Liu, Changping Wang, Yuan Wang, Yuling Xiong, Xun Liu, Haiyang Wu, Qian Li, et al. GPR: Towards a generative pre-trained one-model paradigm for large-scale advertising recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7477–7488, 2026.

Hakbin Kim and Dong-Wan Choi. Conditional knowledge distillation for model specialization. *Journal of KIISE, JOK*, 48(4):369–376, 2021. doi: 10.5626/JOK.2021.48.4.369.

Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. MiniOneRec: An open-source framework for scaling generative recommendation. *arXiv preprint arXiv:2510.24431*, 2025.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11513–11522. IEEE, 2022.

Pan Li and Alexander Tuzhilin. Deep Pareto reinforcement learning for multi-objective recommender systems. *MIS Quarterly*, 50(3):1087–1116, 2026.

Zihan Liu, Yupeng Hou, and Julian McAuley. Multi-behavior generative recommendation. In *CIKM*, 2024.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 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*, pp. 1930–1939, 2018a.

Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 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*, pp. 1137–1140, 2018b.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets, 2015. URL `https://arxiv.org/abs/1412.6550`.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*, pp. 269–278, 2020.

Yejing Wang, Shengyu Zhou, Jinyu Lu, Qidong Liu, Xinhang Li, Wenlin Zhang, Feng Li, Pengjie Wang, Chuan Yu, Jian Xu, et al. GFlowGR: Fine-tuning generative recommendation frameworks with generative flow networks. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1925–1936, 2026a.

You Wang, Zhao Liu, Guoping Tang, Yiqing Yang, Shuo Su, Jing Liu, Naifu Zhou, Xiaoyou Zhou, Wei Jiang, Jian Liang, et al. Multi-decoder OneRec: Controllable generative retrieval for multi-objective industrial recommendation. *arXiv preprint arXiv:2607.26500*, 2026b.

Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1285–1294, 2017.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Qingbo Zhang, Xiaochun Yang, Hao Chen, Bin Wang, and Xiangmin Zhou. Intention-sensitive preference learning network for personalized session-based recommendation. *ACM Transactions on Recommender Systems*, 4(1):1–23, 2025. doi: 10.1145/3727647.

Xinxun Zhang, Yuling Xiong, Jiale Zhou, Zhengkai Guo, Zhennan Pang, Junbang Huo, Jingwen Wang, Xuyang Sun, Enming Zhang, Jiaguang Jin, et al. Unified value alignment for generative recommendation in industrial advertising. *arXiv preprint arXiv:2605.05803*, 2026.

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 43–51, 2019.

Zhen Zhao, Tong Zhang, Jie Xu, Qingliang Cai, Qile Zhang, Leyuan Yang, Daorui Xiao, and Xiaojia Chang. Farewell to item ids: Unlocking the scaling potential of large ranking models via semantic tokens. *arXiv preprint arXiv:2601.22694*, 2026.

Guorui Zhou, Hengrui Hu, Hongtao Cheng, Huanjie Wang, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Lu Ren, Liao Yu, Pengfei Zheng, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Ruiming Tang, Shiyao Wang, Shujie Yang, Tao Wu, Wuchao Li, et al. OneRec-V2 technical report, 2025. URL `https://arxiv.org/abs/2508.20900`.

Yichu Zhou, Mehdi Ben Ayed, Lin Yang, Jiacong He, Andreanne Lemay, Jiaye Wang, Jaewon Yang, Josie Zeng, Dhruvil Deven Badani, Yijie Dylan Wang, et al. A production-ready RL framework for personalized utility tuning with Pareto sweeping in Pinterest recommender systems. *arXiv preprint arXiv:2605.16344*, 2026.

A Implementation and codebook variants

Training budgets. On Kwai26, TIGER, OneRec-V2, MD-CBS, and MOGR use one chronological pass over the official training sessions and the released $50{,}000$-user test split. Main-table MOGR uses the official SID lexicon. Teacher training and generator OCD share that budget. MD-CBS is Multi-Decoder OneRec with the paper quotas. Routes run Long-View, Like, Forward, then general, with quotas $86/85/85/256$ and beam widths $86/171/256/512$. Collision masks use the full SID; the routes merge item-wise into $C(u; 512)$. On the industrial dump of Table 4, codebook variants train the teacher and the generator for one pass each (AdamW; teacher learning rate 5×10^{-5}, batch 64; λ_{T}=1, β_{T}=0.3; generator exclusive-task SFT at 10^{-4}, then OCD at 2×10^{-5}; Eq. (3)).

Table 3: Kwai26 official split Wang et al. (2026b). Forward is aligned from the Stage 1 share flag. Positive rates are on the 191,335 official target slots of the 50,000 test sessions.

	Kwai26
Users / test sessions	50,000 / 50,000
Items with valid SIDs	25,028,687
Training sessions	125,261,311
SID levels / codes	3 / 8,192
History	recent 20 + Long-View 256
Test target slots	191,335
Max targets per session	8
Objectives	Long-View / Like / Forward
Positives (slots)	41,944 / 9,866 / 1,514
Slot rate	21.92% / 5.16% / 0.79%
Sessions with ≥ 1 positive	24,031 / 5,935 / 1,349

Table 4: Sampled 19-day industrial dump for OBRQ codebook swaps (desensitized).

	Industrial
Users / items	683,707 / 21,007,902
Requests / impressions	1.09×10^8 / 2.84×10^8
Time span	1–19 Jul. 2026
Train / test days	15 / 4
History T / SID (L, C)	200 / $(4, 256)$
Objectives	Read (dwell>3s) / Poi / Int

Public protocol details. Kwai26 follows the released session construction, histories, and 50,000-user test split Wang et al. (2026b). Forward flags are joined from the released Stage 1 share flag onto official target slots and are not written back into Stage 2. The backbone is d=256, 8 heads, 4+4 layers, d_{ff}=1024. The user encoder is the OneRec five-field encoder (item ID, author ID, tag, time gap, and watch time) Deng et al. (2025). The teacher uses λ_{T}=1, β_{T}=0.3, and positive upweight $w_{i,k} = 1 + \alpha_k y_{i,k}$ with $\alpha = (2, 1, 0.5)$ on (Long-View, Like, Forward). SFT exclusive labels pick one token; α_k only upweights positives on the full-exposure CE, so the frequent head can take a larger α. D_{exp} and each D_T^k have the same depth as the SID-baseline decoder; only D_{exp} updates E_T and the SID table. Generator SFT uses exclusive task labels (Kwai26: Forward $\succ$ Like $\succ$ Long-View; industrial: Int $\succ$ Poi $\succ$ Read; else t_{exp}), a separate embedding per task token, and Eq. (3) with λ_{S}=0.3, $\varepsilon_{\text{task}}$=0.1, β_{S}=0.1. OCD uses learning rate 2×10^{-5}, matches specialist SID policies under forced t_k, and anchors the four-way task softmax to the frozen SFT generator with λ_{task}=0.1 (Eq. (6)). Online LGPO uses user-window groups of G_L=8 consecutive requests from the same user, list-reward weights $w = (1/3, 1/3, 1/3)$ on Read / Poi / Int, λ_{sid}=0.1, clip δ=0.2, KL coefficient β_L=0.01, and EMA decay 0.999 for π_{ema}. One uniformly random slot per list samples its task token from the restricted, untempered softmax; all other task tokens and all SID tokens are greedy. Serving masks t_{exp}, and LGPO scores the same task softmax. Frozen OCD is fully greedy in the original experiment. An additional concurrent, user-randomized test starts frozen OCD and OCD+LGPO from the same OCD checkpoint with identical decoding settings; only OCD+LGPO updates. It runs for seven days with 120,000 users per arm. Industrial Content RQ uses the frozen 128-d content embedding of Section 3.5 and the balanced residual K-means defined there, with centroid cap $1.25\,n/C$, L=4, C=256, fit on a 200,000-item impression-weighted subset of the industrial training split.

Collaborative embeddings. For each objective an in-batch InfoNCE (Oord et al., 2018) DSSM (Huang et al., 2013) is trained on y_k=1 positives. The user tower takes a user identifier plus static profile; the item tower takes coarse sparse features, an item cluster, and the frozen 128-d content vector (no learned item-id table). The 128-d item-tower output is frozen as $\mathbf{e}_i^k$. These vectors are used only when building Concat RQ or OBRQ.

Table 5: Industrial Recall@512 on the codebook swap. Each row retrains teacher and generator. The merged-pool generator optimizes the aggregated signal, while the forced-t_k generator decodes a dedicated pool of 512 under a per-slot task token. Min-G is the minimum relative lift versus Content RQ (%).

	Merged pool				Forced t_k			
Codebook	Read	Poi	Int	Min-G↑	Read	Poi	Int	Min-G↑
Content RQ	0.121	0.113	0.157	0.0	0.123	0.116	0.159	0.0
Concat RQ	0.134	0.128	0.165	5.1	0.135	0.132	0.167	5.0
OBRQ	**0.138**	**0.131**	**0.172**	**9.6**	**0.140**	**0.134**	**0.173**	**8.8**

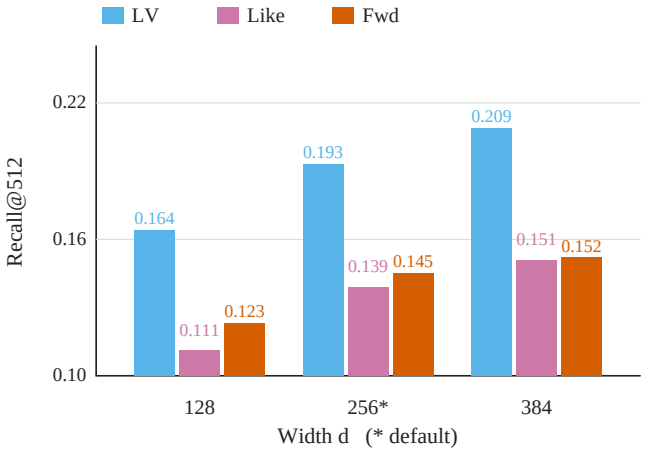

Figure 2: Kwai26 Recall@512 versus decoder width. d=256 is the default in Table 1.

Frozen-code diagnostics. Let $p_i^k = n_k(i)/n_{\text{impr}}(i)$ on the training window, restricted to items with $n_{\text{impr}} \geq 20$. $V^k(d)$ is the unweighted mean of $\text{Var}\{p_i^k\}$ over prefix SIDs of depth d that contain at least two kept items. Table 8 reports $\bar{V}^k = \frac{1}{3}\sum_{d=1}^{3} V^k(d)$ ($\times 10^{-2}$): full 4-layer SIDs are 93% singletons on Content RQ, so d=4 is omitted from the average. Ent. is the first-layer codebook entropy over $\log(C-1)$: one index is reserved as unknown, so occupancy is over $C-1$ live centroids. Used / Coll. are full-SID occupancy (Used/256^4). Collision is catalog size over occupied SIDs.

Table 6: Different configurations of Teacher and MOGR on Recall@512. Teacher: dedicated pool of 512 from D_{exp} or D_T^k. MOGR: t_k forced.

Variant	Long-View	Like	Forward
Teacher D_{exp} only	0.180	0.127	0.134
Teacher *w/o* KL	**0.215**	0.153	0.151
Teacher Full	0.214	**0.154**	**0.166**
MOGR *w/o* OCD	0.182	0.133	0.143
MOGR	0.197	0.141	0.147

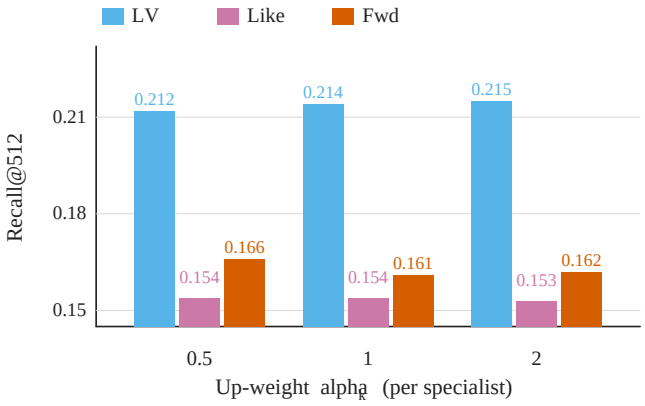

Figure 3: Teacher positive up-weight $w_{i,k} = 1 + \alpha_k y_{i,k}$. Each D_T^k uses its own α_k. Dedicated specialist pool of 512. Default $\alpha = (2, 1, 0.5)$ on Long-View / Like / Forward.

Table 7: Online relative lifts (%). The first three rows use greedy frozen OCD as the reference. The last row reports the direct comparison between OCD+LGPO and concurrent frozen OCD under matched exploration. Min-G is the minimum relative lift across objectives.

Arm	Read	Poi	Int	Min-G ↑
OCD+LGPO	**0.302**	**0.258**	**0.299**	**0.258**
R_{lin}	0.151	0.097	0.109	0.097
λ_{sid}=1	0.019	0.135	0.003	0.003
Matched exploration (OCD+LGPO vs. frozen OCD)				
OCD+LGPO	0.254	0.218	0.269	0.218

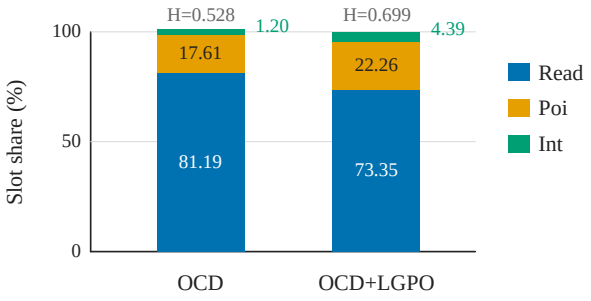

Figure 4: Slot share of emitted task tokens (%) in the original experiment. Entropy H (nats) is annotated above each stack.

Table 8: Frozen SID diagnostics on industrial training items with $n_{\text{impr}} \geq 20$ (C^L=256^4). $\bar{V}^k = \text{mean}_{d=1}^3 V^k(d)$ ($\times 10^{-2}$, lower better). Ent. is first-layer entropy over $\log(C-1)$. Used / Coll. are full-SID occupancy (Used/256^4).

Codebook	$\bar{V}^{\text{Read}}$	$\bar{V}^{\text{Poi}}$	$\bar{V}^{\text{Int}}$	Ent.	Used / Coll.
Content RQ	1.194	0.343	0.0143	0.972	18.10M / 1.16
Concat RQ	1.113	0.341	0.0139	0.970	17.26M / 1.21
OBRQ	**1.024**	**0.335**	**0.0132**	**0.975**	17.87M / 1.17

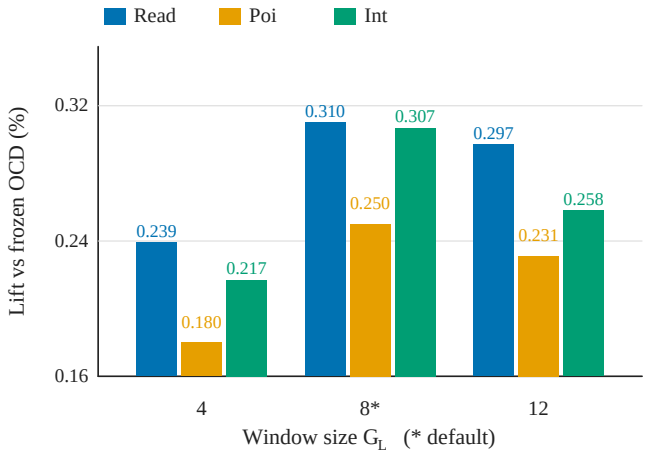

Figure 5: Relative lift versus frozen OCD (%) by user-window size. G_L=4/12 are a later period that also runs G_L=8 and frozen OCD. Min-G is the lowest of the three series (Poi here).

B KWAI26 OFFLINE LGPO PROXY

The proxy asks whether LGPO changes objective allocation on lists the OCD generator emits.

Table 9: Kwai26 LGPO proxy under the held-out reward model. Entries are proxy utilities. Min-G is the minimum relative change versus frozen OCD (%).

Policy	Long-View	Like	Forward	Min-G (%)
Frozen OCD	1.328	1.306	1.226	–
LGPO	**1.364**	**1.350**	**1.281**	**2.711**

Split. The 125,261,311 training sessions are split into six contiguous slices that do not overlap. SFT and OCD use the first 55,000,000. Reward model A trains on the next 30,630,655 and holds out the following 2,000,000 for its Z_k and AUC. LGPO runs for one epoch on the next 5,000,000 sessions from this OCD checkpoint. The reward is model A's probability, and A's parameters stay fixed during the update. Reward model B trains on the next 30,630,656 and holds out the last 2,000,000 training sessions for its own Z_k and AUC. The official test sessions are used for the comparison in Table 9; all 50,000 are scored by model B.

Reward model. Models A and B are trained separately from scratch. Each has its own encoder and its own SID embedding table, and neither is initialized from the OCD checkpoint. The user vector is the masked mean of that model's encoder memory over the five history fields. The item vector is the mean of that model's three SID embeddings. A two-layer MLP maps the pair to three logits and is trained with binary cross-entropy on logged slate labels. Positive weights are the slot-frequency weights on that model's own training sessions. They stay fixed when the holdout and the test comparison are scored. The score uses the history and the item SID. Holdout AUC on Long-View, Like, and Forward is $0.752/0.743/0.820$ for model A and $0.761/0.762/0.822$ for model B. On each model's own holdout, Z_k is the mean discounted list sum of that model's probabilities and is then held fixed. Training uses model A's probabilities and Z_k; Table 9 uses model B's.

List score. With the same rank discount $d_j = 1/\log_2(j+1)$ as Section 3.4,

$$\hat{U}_k = \frac{1}{Z_k} \sum_{j=1}^{N} d_j \, p(y_k{=}1 \mid \mathcal{H}, i_j). \tag{11}$$

The sum uses the probability. $N{=}8$. The list reward remains Eq. (7) with $w_k{=}1/3$. Advantage is the within-group standardization of Eq. (8).

Update. Each history yields 4 lists. Decoding follows Section 3.4: t_{exp} is masked, one uniform slot samples its task token, and every other task token and every SID token is greedy under the codebook trie, with items already emitted removed. The loss is $\mathcal{L}_{\text{LGPO}}$, with the online coefficients λ_{sid}, δ, β_L, and the EMA decay. One epoch is run from the OCD checkpoint above. The learning rate is $2{\times}10^{-5}$.

Evaluation. Frozen OCD and LGPO both decode fully greedy lists. Figure 6 reports task-token slot share under that decode. Min-G in Table 9 is the minimum relative change in proxy utility versus frozen OCD.

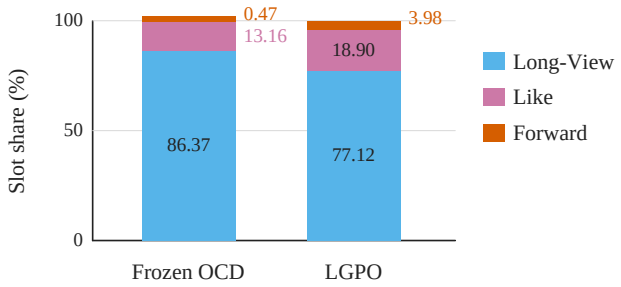

Figure 6: Task-token slot share (%) on the Kwai26 LGPO proxy, fully greedy lists.

Protocol gaps. A group shares one history. The live user window instead stacks consecutive requests, and later requests contain newly consumed items. List length is $N{=}8$. The industrial list has length 20. Both reward models emit probabilities, so neither $\hat{U}_k$ nor the training reward is live feedback. Generated SIDs and the logged slate barely overlap, which is why models A and B score the generated lists.