Generative Retrieval via Diffusion Transformer with Metric-Ordered Sequence Training and Hybrid-Policy Preference Optimization

Anonymous authors
Paper under double-blind review

Abstract

Embedding-based retrieval ranks items by their similarity to a query in a shared vector space and usually aims to return the highest-scoring items. In many production settings this is not what is wanted: given a seed set that expresses a fine-grained pattern, one needs more items that both satisfy a target attribute and stay within that pattern. We formalize this as *pattern-preserving attribute retrieval*. The two goals pull against each other: averaging the seeds preserves the pattern but stays in a low-attribute region, while global attribute retrieval drifts to unrelated patterns. We approach the task with continuous generative retrieval, where a model reads a sequence of item embeddings and generates one or more query embeddings for nearest-neighbor search. We propose MO-DiT+HPPO (Metric-Ordered Diffusion Transformer with Hybrid-Policy Preference Optimization, HPPO), a staged framework with large-scale raw-sequence pretraining, multi-domain metric-ordered continuation pretraining, domain-specific tail-centroid supervised fine-tuning, and HPPO. The metric-ordered stages convert sparse online retrieval labels into in-pattern trajectories ordered from low to high predicted attribute density, so a single continuation-pretrained model learns the metric-improvement direction across all domains. HPPO then aligns the final query distribution with the true online objective: in each round it forms a hybrid-policy candidate pool—deterministic high-metric constructions together with the current policy's own guidance-scale fan—labels every candidate with the real online intersection metric `Joint@K`, and applies iterated, reference-anchored preference optimization with held-out-validation early stopping; a simple tradeoff-aware Pareto pair filter then keeps only winner pairs that do not lower same-pattern purity, so updates raise the attribute metric without sacrificing the pattern. Across four large-scale attribute domains under strict item- and pattern-holdout protocols, metric-ordered DiT training improves the primary intersection metric over a strong pretrained generative retriever, and HPPO improves it further, with gains that are large and paired-bootstrap significant on seven of the eight domain-split cells and a marginal tie on the hardest domain's pattern split. The Pareto filter is the key ingredient: on most domain–split cells it raises `Joint@K` while keeping the same-pattern share higher than an unconstrained preference policy, pushing the attribute–pattern frontier outward rather than merely sliding along it. Metric-predictor validation, matched order ablations, CPT/SFT comparisons, and a candidate-policy ablation show where the gains come from and that static and policy-generated candidates are complementary.

1 Introduction

Large retrieval systems increasingly serve attribute-seeking objectives. A platform may need to find items that satisfy a quality, safety, policy, originality, or integrity criterion. In many such tasks, the desired result is not an arbitrary high-scoring item. Retrieved items should also match the fine-grained pattern represented by a seed set: the same topic, visual style, failure mode, or retrieval intent. A typical operational case illustrates the tension: reviewers discover a small batch of items that express one new variant of a policy

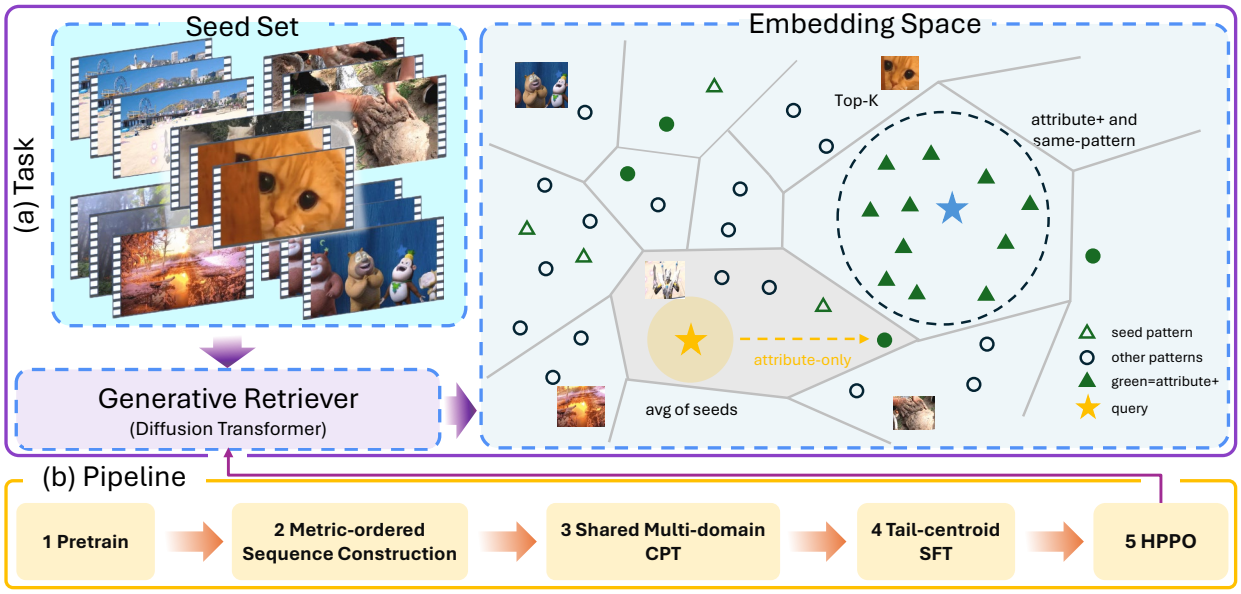

Figure 1: **Pattern-preserving attribute retrieval and the MO-DiT+HPPO pipeline.** Given a seed set from one fine-grained pattern, the objective is to retrieve top-K neighbors that are both attribute-positive and of the seed pattern. Average pooling preserves the pattern but remains in a low-attribute-density region, whereas attribute-only retrieval drifts toward unrelated patterns. The generative retriever instead synthesizes a continuous query embedding near a high-density region of the same pattern. MO-DiT+HPPO follows a staged pipeline (bottom): large-scale raw-sequence pretraining; metric-ordered sequence construction; shared multi-domain continuation pretraining and domain-specific tail-centroid SFT, both trained on the constructed metric-ordered sequences; and online-reward Hybrid-Policy Preference Optimization. The item thumbnails are randomly sampled from a public dataset (Ni et al., 2023) and shown for illustration only.

violation—a specific format, template, or tactic—and need to retrieve many more items of exactly that variant for measurement and treatment. A global attribute classifier returns positives of every variant and floods the review queue with unrelated patterns, while nearest-neighbor expansion of the seeds returns the same pattern at mostly ordinary, low-positive-rate items.

The task therefore sits between two standard tools. Averaging the seed embeddings stays close to the seed pattern but retrieves mostly ordinary examples; optimizing only for the target attribute retrieves many positives from the wrong patterns. The desired query moves from the seed pattern toward a nearby high-density region of the same pattern.

We study this problem in continuous generative retrieval. Each item has a frozen multimodal embedding. A generative retriever receives a sequence of item embeddings and produces a query embedding; the query then retrieves the top-K items whose embeddings are most similar to it, by approximate nearest-neighbor (ANN) search over a large vector index. Continuous generation is attractive because the output query is not restricted to an existing item or to a fixed average of inputs. However, it raises a central supervision question: what target teaches the model to move toward higher-density examples while preserving the pattern?

We propose MO-DiT+HPPO (Metric-Ordered Diffusion Transformer with Hybrid-Policy Preference Optimization, HPPO), a staged training framework for attribute-seeking and pattern-preserving generative retrieval; Figure 1 illustrates the task and the staged pipeline. The first stage is large-scale raw-sequence pretraining, which gives the diffusion transformer a general continuous retrieval prior. The second stage is metric-ordered continuation pretraining, shared across domains. We use online recall density as an accessible proxy metric: for an anchor embedding, attribute density is the fraction of top-K retrieved neighbors that satisfy the target attribute. Because dense online labeling is expensive, we label a sparse subset online, train a lightweight metric predictor over frozen embeddings, and use the predictor to order items inside each latent

pattern cluster. The resulting training examples are sequences that move from lower-metric to higher-metric items within the same pattern; mixing the ordered sequences of all domains lets one continuation-pretrained model internalize the metric-improvement direction as a general capability rather than a per-domain memorization.

The third stage is domain-specific tail-centroid supervised fine-tuning. It maps the input prefix of a metric-ordered sequence to the centroid of its high-metric tail. The centroid target aggregates multiple high-metric examples, reducing sensitivity to a single noisy top item. The fourth stage is HPPO, which labels candidate query embeddings with the final online intersection metric and applies iterated reference-anchored preference optimization. Its key ingredient is a tradeoff-aware Pareto pair filter that prevents preference updates from buying attribute gains at the cost of pattern drift, so alignment improves the primary metric without sacrificing pattern purity. This mirrors the role of post-training alignment: CPT and SFT provide a stable pattern-preserving generator, while HPPO aligns the final query distribution with the true online objective.

Our contributions are:

- We formalize *pattern-preserving attribute retrieval* for continuous generative retrieval, and define an intersection metric (`Joint@K`) that penalizes both attribute failure and pattern drift.
- We introduce *metric-ordered sequence construction*, which turns sparse online retrieval labels into large-scale in-pattern training trajectories that move from low to high attribute density within a single pattern.
- We show that a shared multi-domain metric-ordered continuation pretraining, followed by domain-specific tail-centroid supervised fine-tuning, improves same-pattern attribute retrieval across four large-scale domains.
- We introduce HPPO, which adapts reference-anchored pairwise preference optimization to a continuous diffusion retriever—using the flow-matching loss as an implicit score, with no token or trajectory log-probabilities—and labels candidate queries by the true online intersection metric over a hybrid candidate pool of static high-metric constructions and a guidance-scale fan of policy samples (which we show are complementary).
- We propose a tradeoff-aware *Pareto pair filter* that keeps only purity-non-decreasing winner pairs, so preference optimization raises the intersection metric while, on most domains and splits, also keeping pattern purity above an unconstrained policy—pushing the attribute–pattern frontier outward rather than merely sliding along it—with gains confirmed by paired-bootstrap significance.
- We provide a practical evaluation protocol with strict item- and pattern-holdout splits and paired-bootstrap significance, together with a comprehensive ablation suite: metric-predictor validation, matched order ablations, CPT/SFT baselines, and preference-variant and candidate-policy ablations.

2 Related Work

Generative retrieval. Generative retrieval reframes retrieval as a generation problem, producing document identifiers, semantic identifiers, or query representations rather than directly scoring all items; representative examples include autoregressive entity retrieval, differentiable search indices, and semantic-ID recommenders (De Cao et al., 2021; Tay et al., 2022; Rajput et al., 2023). These methods decode discrete identifiers, which is inherently serial and can hallucinate code sequences that match no real item, since most code combinations are invalid. Our setting differs in that the model generates a continuous query embedding evaluated by downstream nearest-neighbor retrieval under a pattern-preserving attribute objective, sidestepping discrete decoding. Closest to this generation-for-retrieval view are methods that synthesize the retrieval vector. In text retrieval, HyDE generates a hypothetical document and re-encodes it (Gao et al., 2023b); closer to us, diffusion models generate a continuous embedding directly—an oracle next-item embedding from a behavior history in sequential recommendation (Yang et al., 2023), or a query embedding for cross-modal nearest-neighbor search (Gu et al., 2024). We share this continuous-generation mechanism but differ in task

and setting: from a multimodal item sequence we generate a query embedding under a pattern-preserving attribute-density objective—returning items that are both attribute-positive and of the seed pattern—rather than predicting the next item or retrieving within a single modality. Because the conditioning input is an item sequence encoded with self-attention, the encoder side relates to self-attentive sequential recommendation (Kang & McAuley, 2018), but our objective is attribute-preserving retrieval through a generated query rather than next-item prediction. It also connects to classical ideas: unlike dense retrieval (Karpukhin et al., 2020; Johnson et al., 2021) with fixed learned embeddings, we synthesize the query itself; unlike Rocchio relevance feedback (Rocchio, 1971) with its fixed update rule, our tail-centroid target is a learned conditional mapping from the seed sequence; and the metric-ordered construction relates to curriculum learning (Bengio et al., 2009), except the order encodes a retrieval direction—low to high attribute density within a pattern—rather than example difficulty.

Continuous generative models. Continuous generation synthesizes a target vector directly rather than decoding discrete tokens, avoiding the quantization and codebook design that discrete generative retrievers require. Diffusion and score-based models (Ho et al., 2020; Song et al., 2021) learn to reverse a noising process, and their flow-matching / rectified-flow formulations (Lipman et al., 2023; Liu et al., 2023) recast this as learning a velocity field that transports noise to data along nearly straight trajectories, so generation reduces to integrating an ordinary differential equation in a few steps. The diffusion transformer replaces the convolutional U-Net backbone with a transformer and injects the diffusion timestep through adaptive layer normalization (Peebles & Xie, 2023), which scales well and fits naturally when the conditioning signal is itself a sequence; classifier-free guidance (Ho & Salimans, 2022) then trades sample fidelity against conditioning strength at inference time. Most of this literature generates perceptual signals—images, video, or audio—but a related line shows that continuous generation can replace discrete token prediction inside autoregressive models by modeling each continuous token with a small diffusion head (Li et al., 2024). We adopt a diffusion transformer with a flow-matching objective but generate in a frozen item-embedding space: the output is a query embedding consumed by nearest-neighbor retrieval rather than a signal to be rendered. Diffusion has also been applied to recommendation by denoising user interaction vectors (Wang et al., 2023); we instead generate a query embedding for downstream retrieval rather than reconstructing interactions.

Preference optimization and alignment. Reinforcement learning from human feedback aligns generative models with preference signals through a learned reward model (Ouyang et al., 2022). Direct Preference Optimization removes that reward model and trains directly on preference pairs with a reference-anchored classification loss (Rafailov et al., 2023), and the same objective has been adapted to diffusion models (Wallace et al., 2024). A parallel line tunes generative models with policy-gradient RL: denoising is cast as a multi-step decision process (Black et al., 2024), and GRPO (Shao et al., 2024) has recently reached flow-matching generation by converting the deterministic sampling ODE into an SDE with tractable per-step log-probabilities (Liu et al., 2025; Xue et al., 2025). We take the preference route rather than on-policy GRPO because our reward is real online retrieval through a rate-limited service, where labeling candidates once offline is far cheaper than an on-policy sampling loop. HPPO adapts preference optimization to continuous query generation in three ways: a candidate's implicit score is its negative flow-matching loss rather than a token log-probability, preference labels come from real online retrieval rather than human raters or a learned reward model, and the candidate pool mixes deterministic constructions with policy samples (complementary in our ablation). It further adds a tradeoff-aware Pareto pair filter that keeps preference pairs non-decreasing in a secondary objective (pattern purity), so the policy improves the primary metric without drifting off the attribute–pattern frontier and resists the reward over-optimization an unconstrained proxy invites (Gao et al., 2023a). This relates to multi-objective alignment such as multi-objective DPO, which scalarizes several reward models into one preference objective (Zhou et al., 2024); our filter instead enforces the secondary objective at the pair-construction level, with no additional reward models.

3 Preliminaries

Items and embeddings. Each item is a piece of multimodal content (for example, video, image, text, video–text, or image–text). A frozen pretrained multimodal embedding model maps each item i to an embedding $e_i \in \mathbb{R}^d$; these embeddings are fixed throughout and define the retrieval space. A generative

retriever receives a sequence of item embeddings and produces one or more query embeddings,

$$q = f_\theta(e_{i_1}, \ldots, e_{i_m}), \qquad q \in \mathbb{R}^d, \tag{1}$$

each searched in the same vector index as the item embeddings.

Attributes and patterns. Each domain y exposes two signals over items. A production *attribute scorer* gives a binary label $a_y(i) \in \{0, 1\}$ for whether item i meets a target attribute—a quality, safety, policy, originality, or integrity criterion. The scorer is reliable at its own granularity but coarse: it does not separate the many distinct sub-types that exist within one attribute. To recover finer structure we cluster the attribute-positive items in embedding space and use the cluster index $c_y(i) \in \{1, \ldots, C\}$ as a *pattern* proxy—a fine-grained sub-type within the attribute. Such fine-grained patterns have no closed vocabulary and are extremely numerous, so labeling the pattern of every item at scale—whether by human annotation or by prompting a strong external model with case-specific instructions for each pattern—is prohibitively expensive; the embedding clustering above is a scalable, label-free approximation. On a sampled set of cases we qualitatively compared the cluster grouping against manual pattern judgments and found them consistent. Both signals are used only to define the evaluation metrics and the training reward: the attribute scorer is an online production model and the clustering is computed offline, and the generative retriever invokes neither at inference time.

Task. A seed sequence $X = (e_{i_1}, \ldots, e_{i_m})$ is drawn from a single pattern c, so all seeds share the pattern. The goal is to generate a query whose top-K retrieved neighbors are dense in items that are *both* attribute-positive *and* of the seed pattern. Averaging the seed embeddings stays inside the pattern but lands in a low-attribute region, while optimizing only for the attribute drifts to other patterns; the desired query therefore *seeks* higher attribute density while *preserving* the seed pattern.

Metrics. All metrics are densities over the top-K retrieved set $\mathcal{R}_K(q)$, i.e. properties of the retrieved neighborhood rather than of a single item. The raw attribute density

$$D_y(q) = \frac{1}{K} \sum_{j \in \mathcal{R}_K(q)} a_y(j), \tag{2}$$

reported as `Attr@K`, is insufficient alone: it rewards attribute hits even when the query drifts to a different pattern. The primary metric is the intersection density

$$J_y(q, c) = \frac{1}{K} \sum_{j \in \mathcal{R}_K(q)} a_y(j) \mathbb{I}[c_y(j) = c], \tag{3}$$

reported as `Joint@K`, which requires both attribute correctness and pattern preservation. We also report the conditional same-pattern purity among retrieved attribute positives,

$$P_y(q, c) = \frac{\sum_{j \in \mathcal{R}_K(q)} a_y(j) \mathbb{I}[c_y(j) = c]}{\max(\sum_{j \in \mathcal{R}_K(q)} a_y(j), 1)}, \tag{4}$$

reported as `Cond@K`, and the same-pattern share

$$S_y(q, c) = \frac{1}{K} \sum_{j \in \mathcal{R}_K(q)} \mathbb{I}[c_y(j) = c], \tag{5}$$

reported as `Same@K`, the fraction of retrieved items in the seed pattern regardless of the attribute.

4 Method

MO-DiT+HPPO is a staged framework (Figure 1): (i) large-scale raw-sequence pretraining for a general continuous-retrieval prior, (ii) shared multi-domain metric-ordered continuation pretraining, (iii) domain-specific tail-centroid supervised fine-tuning, and (iv) Hybrid-Policy Preference Optimization (HPPO). All

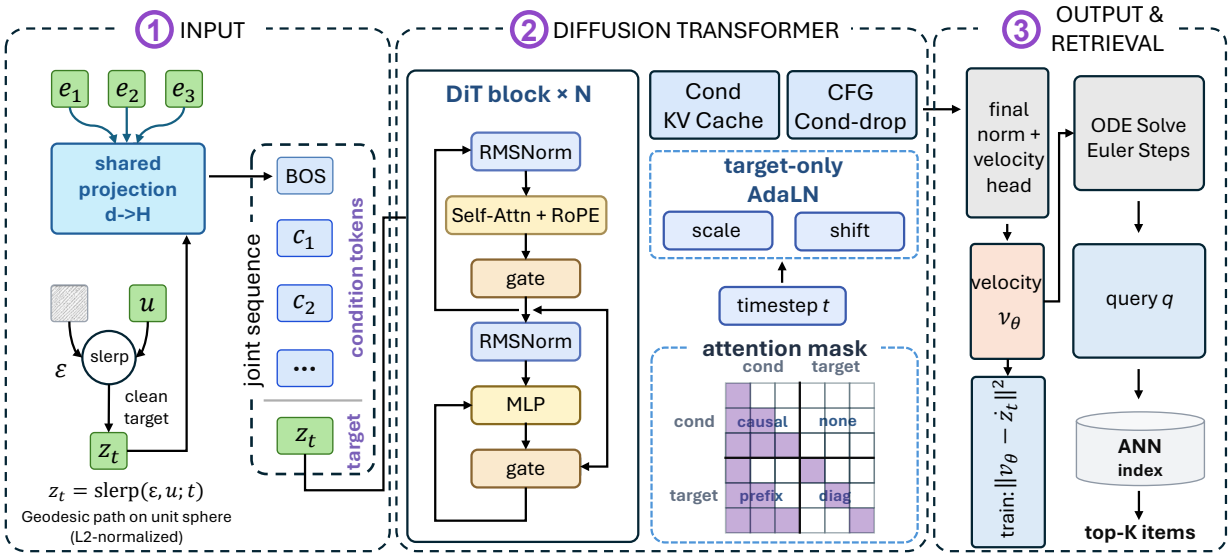

Figure 2: **Continuous generative retriever architecture.** Frozen item embeddings are projected into condition tokens and concatenated with a noised target token on a spherical flow-matching path. A single diffusion transformer jointly processes condition and target tokens under a structured mask: condition tokens encode the prefix, each target attends to its condition prefix but not to other targets, and condition positions do not attend to targets. The target timestep is injected through AdaLN, the velocity head predicts the flow field, and ODE integration produces a d-dimensional query embedding that is searched directly in the ANN index.

four training stages reuse one diffusion-transformer backbone and the same flow-matching objective, differing only in their training data and supervision target. This section describes the backbone, the construction of metric-ordered sequences, the two metric-ordered training objectives, the preference-alignment stage, and inference in turn.

4.1 Continuous Generative Retriever

Figure 2 sketches the architecture; the retriever operates entirely in the frozen item-embedding space defined in Section 3.

The model is a single diffusion-transformer backbone (Vaswani et al., 2017; Peebles & Xie, 2023) trained with a flow-matching objective (defined in the Metric-Ordered Training Objectives subsection below): the architecture is the diffusion transformer, and the generative objective is flow matching, as in recent continuous generators. It processes the condition tokens and the noised target tokens jointly, rather than compressing the condition into one vector and feeding it to a separate generation head. Each condition position projects its item embedding to the transformer width and adds a positional encoding and a learned role marker that separates condition positions from target positions; a binary condition mask marks valid positions so batched sequences can use different effective prompt lengths.

Each target position holds a noised target token, a point on the flow-matching path between Gaussian noise and the clean target at a sampled time t (the objective is defined in that subsection). The condition tokens and the noised target token(s) are concatenated into one sequence and passed through the transformer under a structured attention mask: condition positions attend causally among themselves; each target attends to its condition prefix but not to other targets, so several targets are generated independently in one pass; and condition positions do not attend to targets, so the condition is encoded once and reused. The time t is injected at the target positions through adaptive layer normalization, and the network predicts a flow-matching velocity; integrating the resulting ODE from Gaussian noise yields the output query embedding in

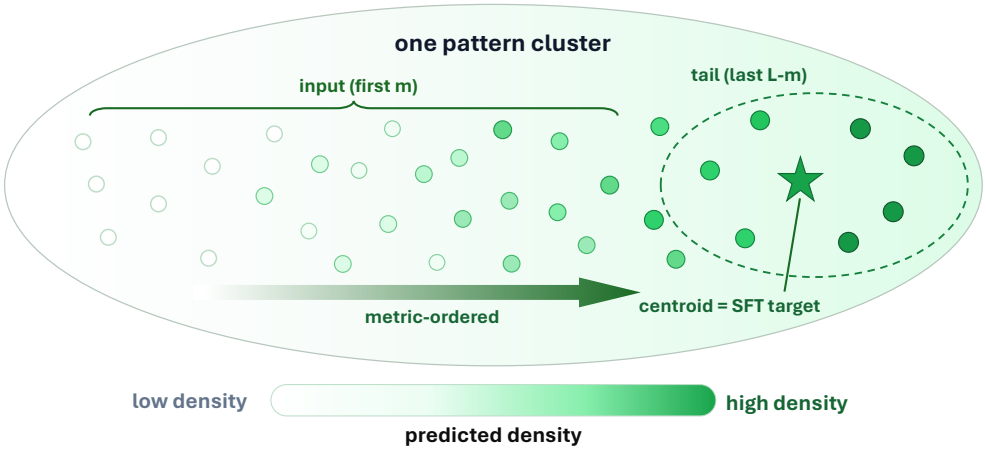

Figure 3: **Metric-ordered in-pattern trajectory construction.** Within each latent pattern cluster, items are scored by a lightweight predictor of online recall density and sorted from low to high predicted attribute density. The predictor is used only to order training sequences; final metrics are always computed by real top-K retrieval. Continuation pretraining supervises local prefix-to-target transitions along the ordered trajectory, while tail-centroid SFT maps the first m items to the centroid of the final $L - m$ high-density tail. This converts sparse online labels into dense same-pattern supervision for learning the direction from ordinary in-pattern items toward high-attribute-density regions.

the same d-dimensional space as the item embeddings, which is searched directly in the vector index with no quantization or identifier decoding. Because each target attends to the full condition sequence rather than a pooled summary, the model retains fine-grained information from the seed items; because the output is continuous, it can synthesize a query that lies between observed items rather than copying one.

4.2 From Sparse Metric Labels to Ordered Sequences

The main difficulty is obtaining training targets that move a query in the desired direction. For each domain y, we first collect items that are positive under the domain scorer and cluster their frozen embeddings into C groups with k-means. These clusters are not assumed to be human-labeled semantic classes. They are scalable latent pattern proxies: they define the local region in which the generated query should remain, and they allow evaluation to distinguish same-pattern positives from arbitrary positives.

Next, we estimate where high-density regions lie inside each pattern. For a sampled set of anchor embeddings, we compute online recall density by retrieving top-K neighbors from the production-like vector index and applying the target scorer to the recalled items. This label measures the actual retrieval consequence of using the anchor as a query. It is expensive because it requires retrieval, item lookup, and scorer access; therefore, MO-DiT+HPPO labels only a subset online.

The sparse labels are expanded to the full item pool with a density predictor. We train a ridge regression predictor $\hat{D}_y(e)$ from frozen embeddings to online recall density. The predictor is deliberately simple and is used for ordering, not for final evaluation. This distinction matters: a predicted density value is never reported as a retrieval result. It only decides the relative order of items during sequence construction, while all final metrics are computed after real top-K vector retrieval.

Within each latent pattern cluster, we sample candidate groups and sort them by predicted density from low to high (Figure 3); Algorithm 1 summarizes the construction. Each emitted sequence has a fixed length L. Normal items may be mixed into the aggregate training pool at a fixed ratio, but the ratio is not forced within every sequence. This preserves a simple data pipeline while matching the target positive-to-normal scale. The key design choice is that every sequence contains a directional signal: early positions represent lower-density examples of the same pattern, and late positions represent higher-density examples.

Algorithm 1 MO-DiT Sequence Construction

Input: positive pool $\mathcal{P}_y$, optional normal pool $\mathcal{N}$, clusters c_y, predictor $\hat{D}_y$
for cluster c **do**
  collect positives $\mathcal{P}_{y,c} = \{i \in \mathcal{P}_y : c_y(i) = c\}$
  sample candidate groups from $\mathcal{P}_{y,c}$ and the aggregate normal pool
  score each item with $\hat{D}_y(e_i)$
  sort each group by predicted density from low to high
  emit a length-L sequence
end for
Output: length-L metric-ordered training sequences

We validate the predictor under item-hash, pattern-out, and super-pattern-out splits (Appendix E), where pattern-out holds out whole clusters and super-pattern-out holds out groups of related clusters. Pattern-out and super-pattern-out evaluations are important because they test whether the embedding geometry itself predicts density on unseen patterns, rather than only memorizing a mean density for seen clusters.

4.3 Metric-Ordered Training Objectives

The generated query should both preserve the input pattern and move toward the high-metric tail of the ordered sequence. We use two metric-ordered training stages to separate these requirements; they build on a backbone first pretrained on large-scale item sequences that carry a natural order (for example, session sequences, user behavior sequences, or time-ordered item streams), with the same flow-matching continuation objective—a raw-sequence pretraining that is not metric-ordered and only establishes a general continuous-retrieval prior.

The first metric-ordered stage is continuation pretraining (CPT). Given a metric-ordered sequence $S = (i_1, \ldots, i_L)$, the model is trained with dense prefix supervision: each target position predicts its own item embedding from the preceding prefix. This objective exposes the model to many local transitions along the low-to-high metric trajectory. It teaches the model the geometry of metric improvement before the final task asks it to produce one query embedding. Continuation pretraining is multi-domain: the ordered sequences of all domains are mixed into one shuffled pool, with per-domain repetition weights so that smaller domains are not drowned out, and a single shared model is trained over the pool. The subsequent SFT and preference stages are domain-specific and initialize from this shared checkpoint.

We train the generator by flow matching (Lipman et al., 2023; Liu et al., 2023). Because the retrieval embeddings are L_2-normalized and therefore lie on the unit hypersphere, we use the spherical (Riemannian) instantiation of flow matching (Chen & Lipman, 2024) rather than straight-line rectified flow. For a conditioning sequence X and a target embedding u, we sample noise $\epsilon \sim \mathcal{N}(0, I)$ projected onto the sphere and a time t from a logit-normal schedule (Esser et al., 2024) ($t = \sigma(\xi)$, $\xi \sim \mathcal{N}(0, 1)$, with σ the logistic function), form the geodesic path z_t from ϵ to u on the sphere, and regress the network velocity onto the path velocity $\dot{z}_t$:

$$\mathcal{L}_{\mathrm{FM}}(X, u) = \mathbb{E}_{t,\epsilon}\left[\|v_\theta(z_t, t, X) - \dot{z}_t\|_2^2\right], \tag{6}$$

where v_θ is the transformer velocity field. At inference we integrate $\dot{z} = v_\theta$ from $z_0 = \epsilon$ to produce the query embedding. We write $\mathcal{L}_{\mathrm{FM}}(X, u)$ for this per-target generation loss and reuse it in every stage. For a metric-ordered sequence $S = (i_1, \ldots, i_L)$, continuation pretraining applies it to supervised prefix-target pairs:

$$\mathcal{L}_{\mathrm{CPT}} = \sum_{r \in \mathcal{I}} \mathcal{L}_{\mathrm{FM}}\left((e_{i_1}, \ldots, e_{i_{r-1}}), e_{i_r}\right), \tag{7}$$

where $\mathcal{I}$ is the set of supervised target positions (dense supervision covers every position with a preceding prefix), each predicting its item e_{i_r} from that prefix $e_{i_1}, \ldots, e_{i_{r-1}}$; the diagonal target–target mask lets all of them be supervised in a single pass. This objective gives the model dense training signal along the low-to-high metric trajectory.

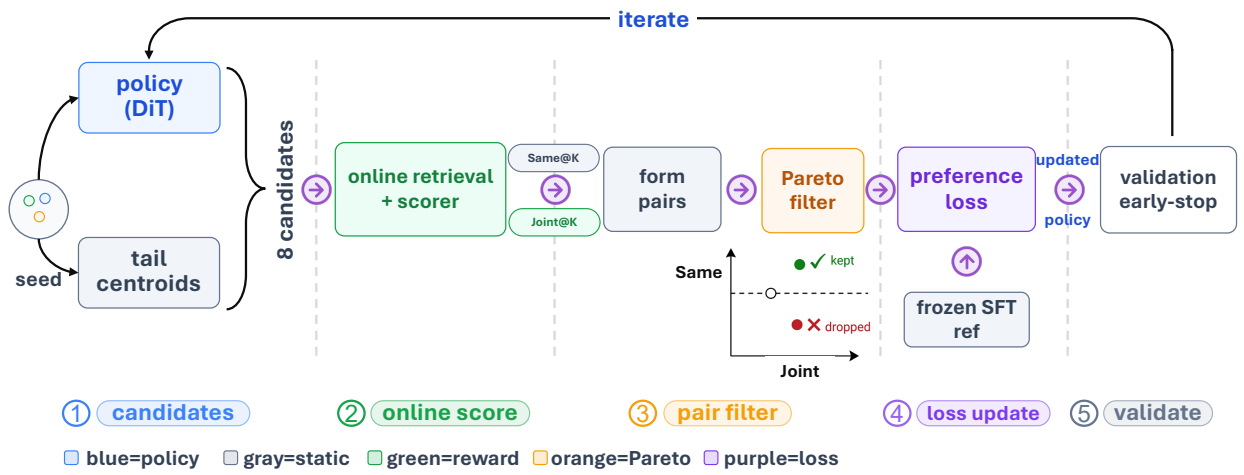

Figure 4: **Hybrid-Policy Preference Optimization.** For each seed case, HPPO builds a hybrid candidate pool from deterministic tail-centroid constructions and policy-generated guidance-fan queries, labels every candidate with real online `Joint@K` and `Same@K`, and forms preference pairs from candidates separated by a reward margin. The Pareto pair filter keeps a winner–loser pair only when the higher-Joint winner does not reduce same-pattern share relative to the loser, preventing attribute gains from being obtained through pattern drift. A frozen SFT reference and tail-anchor loss regularize the reference-anchored preference update, while on-policy rounds regenerate the policy candidates and select checkpoints by held-out online validation.

The second metric-ordered stage is tail-centroid supervised fine-tuning (SFT). Each ordered sequence is converted into one input-target pair. The input is the first m embeddings:

$$X = (e_{i_1}, \ldots, e_{i_m}). \tag{8}$$

The target is the centroid of the final $L - m$ embeddings:

$$t_{\text{tail}} = \frac{1}{L-m} \sum_{r=m+1}^{L} e_{i_r}. \tag{9}$$

The SFT loss is

$$\mathcal{L}_{\text{SFT}} = \mathcal{L}_{\text{FM}}\left((e_{i_1}, \ldots, e_{i_m}), t_{\text{tail}}\right). \tag{10}$$

The centroid target is a stable summary of multiple high-metric same-pattern examples. It is less noisy than a single top item and less narrow than a small top-k target. This is also the reason we do not supervise the model to exactly reproduce the highest-metric item: a top item may be an outlier, while the tail centroid represents a local high-metric region.

In short, each stage changes one component from the previous. Pretraining establishes the prior with a prefix-continuation objective (the per-position $\mathcal{L}_{\text{FM}}$) on raw, non-metric-ordered sequences; CPT keeps that objective but switches the data to metric-ordered sequences; and SFT keeps the metric-ordered data but switches the supervision target to the single tail centroid t_{tail}. The preference stage that follows changes neither the data nor a fixed target, but the learning signal itself, to the true online metric.

4.4 Hybrid-Policy Preference Optimization

Metric-ordered CPT and tail-centroid SFT use constructed targets ordered by the proxy metric, not the final online intersection metric itself. The last stage, Hybrid-Policy Preference Optimization (HPPO), closes this gap by directly optimizing online `Joint@K`. Figure 4 sketches the stage.

For each training case, HPPO forms a candidate set of query embeddings from a hybrid policy. One part is deterministic and metric-based: tail top-k centroids of the ordered sequence (averaging the top-k items for each depth k in a small set $\mathcal{K}$), which are strong pattern-preserving constructions. The other part is policy-generated: queries sampled from the current generative retriever. Because point-target tail-centroid SFT makes the conditional generator nearly deterministic, independently sampled policy queries collapse to almost the same embedding; we therefore diversify the policy candidates with a *guidance-scale fan*, generating the policy part at several classifier-free guidance scales (a small set $\mathcal{S}$). Lower scales follow the seed pattern more closely while higher scales push harder toward the attribute direction, so the fan spans a small spread of pattern-vs-attribute tradeoffs from one policy. Every candidate is searched in the real vector index and labeled by the primary intersection metric $J_y(q, c)$, so the preference signal is the true online objective rather than a learned proxy reward.

Given two candidates q^+ and q^- whose online rewards differ by at least margin δ, we score a candidate by its negative flow-matching loss, $s_\theta(X, q) = -\mathcal{L}_{\text{FM}}(X, q)$—a tractable surrogate for the policy log-likelihood, in the spirit of Diffusion-DPO (Wallace et al., 2024)—and use a frozen SFT reference model to avoid unconstrained drift:

$$\mathcal{L}_{\text{pref}} = -\log \sigma \left(\beta \left[s_\theta(X, q^+) - s_\theta(X, q^-) - s_{\text{ref}}(X, q^+) + s_{\text{ref}}(X, q^-) \right] \right). \tag{11}$$

Here β is the preference temperature. Intuitively, $-\mathcal{L}_{\text{FM}}(X, q)$ measures how readily the current flow model reconstructs the target q from the seed X—a tractable stand-in for the policy likelihood that tracks candidate quality when the generator is well calibrated near q, but is not an exact log-likelihood and can be unreliable for out-of-distribution candidates. The reward margin (paired candidates must differ in true online reward by at least δ) and the frozen SFT reference both anchor the update to the online metric rather than to this surrogate alone. The metric-ordered CPT/SFT stages thus provide the stable pattern-preserving initialization, and HPPO aligns the final query distribution with the online objective.

Tradeoff-aware pair filter. Optimizing the intersection metric naively still admits a degenerate shortcut: a winner can earn a higher J_y by retrieving many more attribute positives while drifting to other patterns, as long as the raw attribute gain outweighs the purity loss. Preferring such winners pushes the policy toward global attribute retrieval, exactly the failure mode the task is meant to avoid. We therefore add a Pareto-dominance constraint to pair construction using the same-pattern share $S_y(q, c)$ (`Same@K`) defined in the problem setup. Among the margin-passing pairs, we keep the pair (q^+, q^-) for the preference loss only if

$$S_y(q^+, c) \;\geq\; S_y(q^-, c) - \varepsilon, \tag{12}$$

that is, the higher-reward winner must not lose same-pattern share relative to the loser (up to a tolerance ε, with $\varepsilon = 0$ the strict setting). The filter is free: S_y is already computed during online labeling, so no extra retrieval is needed. It removes the purity-sacrificing pairs from the gradient and keeps only pairs whose higher reward comes from a Pareto-improving move, so HPPO is steered to push the attribute–pattern frontier outward rather than merely slide along it.

Full objective. A small anchor loss keeps preference updates near the pattern-preserving SFT solution by reusing its tail-centroid target t_{tail},

$$\mathcal{L}_{\text{anchor}} = \mathcal{L}_{\text{FM}}(X, t_{\text{tail}}), \tag{13}$$

and the full HPPO objective averages the preference loss over the retained pairs and adds this anchor,

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{(q^+, q^-) \in \mathcal{P}} \mathcal{L}_{\text{pref}}(q^+, q^-) \;+\; \lambda \, \mathcal{L}_{\text{anchor}}, \tag{14}$$

where $\mathcal{P}$ is the set of (winner, loser) pairs that pass both the reward margin δ and the Pareto filter, and λ is the anchor weight.

Algorithm 2 Iterated Pareto-Filtered HPPO

Input: SFT policy θ_0; frozen reference $\theta_{\text{ref}} = \theta_0$; cases $\{(X_n, c_n)\}$; centroid depths $\mathcal{K}$; guidance scales $\mathcal{S}$; margin δ; tolerance ε; anchor weight λ; max rounds R; validation slice $\mathcal{V}$
$\theta \leftarrow \theta_0$; $J^\star \leftarrow -\infty$
build static candidates (top-k centroids, $k \in \mathcal{K}$) for every case and label them once by online retrieval, giving (J_y, S_y)
for round $r = 1, \ldots, R$ **do**
  $\mathcal{P} \leftarrow \emptyset$
  for each case n **do**
    generate policy candidates from θ at each scale $s \in \mathcal{S}$ and label them online, giving (J_y, S_y)
    $\mathcal{C} \leftarrow$ static $\cup$ policy candidates for case n
    if $\text{top1}(J_y) - \text{top2}(J_y) \geq \delta$ **then**
      $q^+ \leftarrow \arg\max_{q \in \mathcal{C}} J_y(q, c_n)$
      for each $q^- \in \mathcal{C}$ with $J_y(q^+, c_n) - J_y(q^-, c_n) \geq \delta$ **do**
        if $S_y(q^+, c_n) \geq S_y(q^-, c_n) - \varepsilon$ **then**
          $\mathcal{P} \leftarrow \mathcal{P} \cup \{(q^+, q^-)\}$
        end if
      end for
    end if
  end for
  update θ for about one epoch on $\mathcal{P}$, minimizing the preference loss plus the λ-weighted tail anchor against reference θ_{ref}
  $J_r \leftarrow$ true online J_y of θ on $\mathcal{V}$ at the serving guidance scale
  if $J_r > J^\star$ **then**
    $J^\star \leftarrow J_r$; $\theta^\star \leftarrow \theta$
  else
    break (validation early stop)
  end if
end for
Output: aligned policy $\theta^\star$

Iterated hybrid policy with online validation. The preference stage has an off-policy and an on-policy regime. The *off-policy* variant collects candidates once from the frozen SFT policy plus the static constructions. The on-policy *hybrid policy* is iterated: in each round, the guidance-fan policy candidates are regenerated from the *current* policy and relabeled online, while the static candidates and the frozen SFT reference stay fixed. To avoid the drift that uncontrolled iteration produces, each round is short (about one epoch of the kept pairs), the reference is always the original SFT model, and round selection uses a held-out validation slice: a disjoint set of cases is generated at the fixed serving guidance scale, labeled with true online `Joint@K`, and the round checkpoint with the best validation `Joint@K` is carried forward; training stops when a round fails to improve the validation metric. Validation selection, rather than test selection, keeps the protocol honest. We report the off-policy variant and two hybrid-policy variants—*adaptive* (iterated on-policy, no pair filter) and *Pareto* (iterated on-policy with the tradeoff-aware filter)—so the ablation compares off-policy against on-policy (hybrid-policy), and the two hybrid-policy variants against each other. Algorithm 2 summarizes the full iterated Pareto-filtered procedure.

The stage is computationally light. For one domain, about 50,000 cases with eight candidates each yield 400,000 candidate queries, which a single machine labels with real retrieval in under an hour (about 127 queries per second, dominated by vector recall). Margin filtering keeps roughly 12% of cases, and a round of preference training runs in minutes on eight GPUs. The iterated variants reuse this first-round labeling and, in each later round, relabel only the regenerated policy fan; a full run is a few such rounds. The expensive parts of MO-DiT+HPPO are the offline pretraining stages, so the iterated online alignment remains cheap, which makes per-domain refreshes practical.

Table 1: Strict-split data scale. Values are sequence rows, not distinct item IDs. Train rows exclude the held-out clusters; Eval-I and Eval-P are the item-holdout and pattern-holdout evaluation splits; Clusters is held-out over total pattern clusters.

Domain	Train	Eval-I	Eval-P	Normal	Clusters
D1	1.54M	156K	7.8K	10%	50/1000
D2	5.95M	163K	32.8K	20%	50/1000
D3	9.22M	163K	48.7K	20%	50/1000
D4	478K	36K	1.8K	20%	25/500

4.5 Inference

At inference the model conditions on a seed set of same-pattern item embeddings (in place of the training-time ordered prefix) and generates a query embedding. Generation starts from Gaussian noise projected onto the unit sphere and integrates the learned velocity field with a few Euler steps, renormalizing onto the sphere after each step—the spherical counterpart of the training-time flow—and applying classifier-free guidance at the serving scale. Because the conditioning prefix does not attend to the target tokens, it is prefilled once and its key/value cache is reused across all integration steps, so only the target token iterates; serving cost is therefore one short ODE integration plus one approximate-nearest-neighbor query, both cheap. The resulting query is L_2-normalized and searched in the same vector index as the baselines: serving uses the single generated query, while drawing several noise samples yields multiple candidate queries (used by HPPO, and available for inference-time reranking). Unless otherwise stated, evaluation uses top-K retrieval and computes all metrics on the retrieved items, not on predicted density scores.

5 Experiments

5.1 Setup

5.1.1 Domains

We evaluate on four anonymized attribute domains drawn from our internal data, denoted D1, D2, D3, and D4. Each domain has a different attribute definition—for example, criteria related to content quality, policy compliance, originality, or behavioral integrity—and a different data scale; the mapping of labels to criteria is withheld. D1 is the first domain used to validate metric-ordered training; D2 and D3 are larger follow-up domains; D4 is a smaller domain for which a strict held-out-cluster protocol is available. The seed items in each domain are drawn from a single pattern cluster, and the task is recall expansion: from a seed set, retrieve many more items of the same attribute-bearing pattern, a primitive useful for a range of downstream applications such as content discovery, recommendation, near-duplicate and variant detection, and dataset curation. Table 1 summarizes the data scale and split type of each domain.

Labeling and full-pool scoring. Online recall-density labels are used only to train and validate the density predictor. After validation, the predictor scores the full item pool and sequence construction uses the predicted scores to order items. The final retrieval metrics are never computed with predicted density: they are computed by querying the vector index and evaluating the actually retrieved top-K items. This separation keeps labeling cost manageable while preserving an online-retrieval evaluation target.

5.1.2 Baselines and Variants

We compare the following models and query constructions, all evaluated under the strict item/cluster double-holdout protocol—the hardened configuration for the main results, and a lighter (fewer-case) configuration for the order ablation.

- **Pretrained GR (Pre. GR)**: the shared pretrained continuous retriever.
- **Average pooling (Avg.)**: the mean of the input embeddings, used directly as the query.

- **MO-DiT (CPT+SFT)**: shared multi-domain metric-ordered CPT followed by domain-specific tail-centroid SFT—the main pipeline. Its intermediate stage *CPT only* (continuation pretraining without centroid SFT) is traced in the stage-wise progression (Table 4).
- **Single-domain CPT+SFT**: the same pipeline with a per-domain CPT instead of the shared multi-domain CPT; used in the order ablation (Table 5).
- **HPPO variants**: online-reward preference optimization initialized from the MO-DiT SFT checkpoint, with static tail-centroid candidates and guidance-fan policy candidates labeled by online `Joint@K`—*off-policy* (single round), *adaptive* (iterated, no pair filter), and *Pareto* (iterated with the tradeoff-aware pair filter); the reported HPPO row uses the best per-domain variant.

A further set of oracle constructions—sharp target queries from the true tail (**Target top-1/top-5** and **Target centroid**)—exists only under an earlier non-strict, single-domain evaluation and is deferred to Appendix F; we use them only for their relative conclusions, as their absolute numbers are not comparable to the strict tables.

5.1.3 Evaluation Protocol

For each evaluation sequence, we generate a query embedding and retrieve top-K nearest neighbors. We report raw attribute density, intersection density, conditional same-pattern purity among positives, and same-pattern share among all retrieved items. Space-constrained tables use the column names Attr, Same, Joint, and Cond for `Attr@K`, `Same@K`, `Joint@K`, and `Cond@K` as defined in the problem setup. The AUC column reports within-pattern directional discrimination: the positives are the held-out high-metric tail items of a case and the negatives are input items of the same cluster, both scored by similarity to the generated queries. AUC therefore measures displacement along the metric direction inside a pattern; a query that stays at the input centroid, such as average pooling, does not displace toward the tail; because the AUC negatives are the input items it is built from, those negatives rank just above the displaced tail positives and it sits just below 0.5 (marginally above on D4, whose embedding geometry differs), whereas the trained models move clearly above it. Reported values are query-level means; queries with zero retrieved attribute positives are skipped in the conditional mean, so aggregate columns do not multiply out exactly. For the strict-split experiments, every domain reserves both an item-holdout and a pattern-holdout (held-out-cluster) evaluation split, and all weight-updating stages exclude the held-out clusters. The main results (Tables 2, 4, 6 and Figure 6) use a hardened configuration: the within-pattern AUC over up to 1,000 cases per split, and the retrieval-density metrics over 500 cluster-stratified cases per split, with all models scored on the identical, deterministically sampled case set, so the rows are paired and directly comparable. The order ablation (Table 5) uses a lighter matched configuration with 50 stratified density cases per split; its values are comparable within the table but not in absolute magnitude to the hardened tables, and density differences below roughly 1 pp there should be read as within evaluation noise. Significance for the main contrasts is assessed by paired bootstrap over the shared cases (Figure 6).

Checkpoint selection for preference rows stays off the test set. The iterated variants (adaptive and Pareto) select the round and checkpoint on a disjoint held-out validation slice labeled with true online `Joint@K`, and stop when a round does not improve it; the off-policy variant reports the best of three saved checkpoints on the split. Early selection is part of the protocol rather than a tuning trick: continued preference training keeps increasing raw attribute density and AUC while same-pattern purity decays, so dense checkpointing with early selection is required to stay on the favorable side of the attribute–pattern tradeoff. The validation-based selection of the iterated variants is, if anything, stricter than the off-policy variant's split-based selection, so the reported Pareto-over-off-policy gains are conservative.

5.2 Main Results

5.2.1 Main Comparison

Table 2 reports the main result on strict item-holdout and pattern-holdout splits, with all models evaluated under one unified hardened protocol so the rows are directly comparable. Average pooling behaves exactly

Table 2: Main result on strict item-holdout and pattern-holdout splits. Avg. is average pooling of the input embeddings; Pre. GR is the shared pretrained retriever; MO-DiT is the shared multi-domain metric-ordered CPT plus domain-specific tail-centroid SFT; HPPO is the best per-domain preference variant (the iterated Pareto-filtered policy for D1/D2/D3, and the off-policy single round for D4, the one domain where the filter does not help; see Table 6), selecting the checkpoint with the best *validation* `Joint@K` per split. All rows use one unified hardened evaluation protocol with 500 cluster-stratified cases per split shared across models, so values are directly comparable; `Joint@K` differences are tested for significance by paired bootstrap (Figure 6). Attr, Same, Joint, and Cond are percentages; AUC is a 0–1 discrimination score.

Eval	Method	AUC	Attr	Same	Joint	Cond	Eval	Method	AUC	Attr	Same	Joint	Cond
D1							**D3**						
Item	Avg.	0.4793	12.63	90.51	8.29	87.65	Item	Avg.	0.4565	8.97	93.17	8.03	92.56
Item	Pre. GR	0.4715	11.86	67.16	5.99	64.73	Item	Pre. GR	0.4473	9.60	70.62	6.44	70.01
Item	MO-DiT	0.5278	18.11	88.37	11.37	84.51	Item	MO-DiT	0.5182	12.46	92.11	10.72	91.30
Item	HPPO	0.5992	35.34	76.29	**17.36**	65.90	Item	HPPO	0.5916	42.08	66.97	**21.42**	59.85
Pattern	Avg.	0.4741	8.80	93.94	6.06	92.17	Pattern	Avg.	0.4573	9.10	93.77	8.50	95.19
Pattern	Pre. GR	0.4659	8.58	73.30	4.58	68.70	Pattern	Pre. GR	0.4525	9.14	73.56	6.75	75.86
Pattern	MO-DiT	0.5171	11.55	89.92	7.74	83.92	Pattern	MO-DiT	0.5054	11.77	91.06	10.72	92.74
Pattern	HPPO	0.5952	26.18	80.50	**13.84**	70.96	Pattern	HPPO	0.5810	32.56	70.05	**17.48**	59.53
D2							**D4**						
Item	Avg.	0.4977	3.89	78.75	2.81	73.82	Item	Avg.	0.5631	16.21	87.53	9.53	82.93
Item	Pre. GR	0.4867	3.57	57.69	1.87	54.24	Item	Pre. GR	0.5485	14.42	68.04	6.86	66.90
Item	MO-DiT	0.5413	5.36	77.20	3.79	73.59	Item	MO-DiT	0.6093	21.35	82.51	11.19	75.81
Item	HPPO	0.6068	22.47	52.68	**10.28**	46.00	Item	HPPO	0.6561	30.07	71.27	**12.39**	60.90
Pattern	Avg.	0.5013	3.77	79.32	2.56	77.24	Pattern	Avg.	0.5251	17.00	80.23	7.90	73.02
Pattern	Pre. GR	0.4886	3.66	58.11	1.98	60.70	Pattern	Pre. GR	0.5117	14.69	66.44	5.82	62.13
Pattern	MO-DiT	0.5426	7.16	75.94	4.99	72.49	Pattern	MO-DiT	0.5871	22.67	73.66	8.99	65.16
Pattern	HPPO	0.6087	27.74	54.14	**14.40**	48.64	Pattern	HPPO	0.6216	28.11	67.70	**9.12**	56.59

as the motivation predicts: it keeps the highest same-pattern share but stays in a low-attribute region, which caps its `Joint@K`. MO-DiT improves `Joint@K` over both the pretrained retriever and average pooling on every domain and split. HPPO improves `Joint@K` over its MO-DiT initialization on all eight domain-split cells. The gains are paired-bootstrap significant with comfortable margins on seven of the eight cells; the exception is D4 pattern-holdout, where the gain (+0.12 pp, 95% CI $[+0.01, +0.24]$ pp) only marginally excludes zero and should be read as a near-tie (Figure 6): D4 offers the preference stage little intersection headroom, being by far the smallest domain. On D1, D2, and D3 the gains are uniformly large, from +6.0 pp (D1 item, CI $[+5.1, +6.9]$ pp) to over +10 pp (D3 item); the D4 item gain is smaller but still significant (+1.2 pp, CI $[+0.8, +1.6]$ pp). The detailed columns expose the mechanism: HPPO raises raw attribute density substantially while giving up part of the same-pattern share and conditional purity, and the net effect on the intersection metric is positive. Crucially, on most D1/D2/D3 cells the Pareto-filtered variant keeps the same-pattern share higher than the unconstrained preference policy at a higher `Joint@K`, the intended tradeoff-aware behavior: in Figure 5 the Pareto point sits up and to the right of the off-policy point on these cells, improving both axes, while Table 6 quantifies it. The exception is D3 pattern-holdout, where the higher `Joint@K` comes with a same-pattern-share cost rather than a gain (Table 12).

5.2.2 Significance

Because every model is evaluated on the identical set of cluster-stratified cases (deterministic sampling, verified 100% case-id overlap) and the per-query intersection values are stored, the comparisons are *paired*. We resample the shared cases with replacement ($B = 10{,}000$) and report the bootstrap distribution of each per-case mean difference. Figure 6 shows the three key `Joint@K` contrasts. The headline gain of HPPO over MO-DiT is significant on every cell except D4 pattern-holdout, where it is a marginal tie. The Pareto pair filter is significant over both the iteration-only policy (*adaptive*) and the single-round *off-policy* run on all six D1/D2/D3 cells; on D4 the filter does not help and the off-policy variant is the one reported in Table 2. Paired testing is what resolves the close contrasts: the marginal confidence intervals of the Pareto and adaptive runs overlap (e.g. D1 item `Joint@K` $17.4\,[15.8, 18.9]$ vs. $16.2\,[14.7, 17.7]$, in %), yet their paired difference $+1.1\,[+0.8, +1.4]$ pp excludes zero because case-difficulty variance cancels. This is eval-sampling

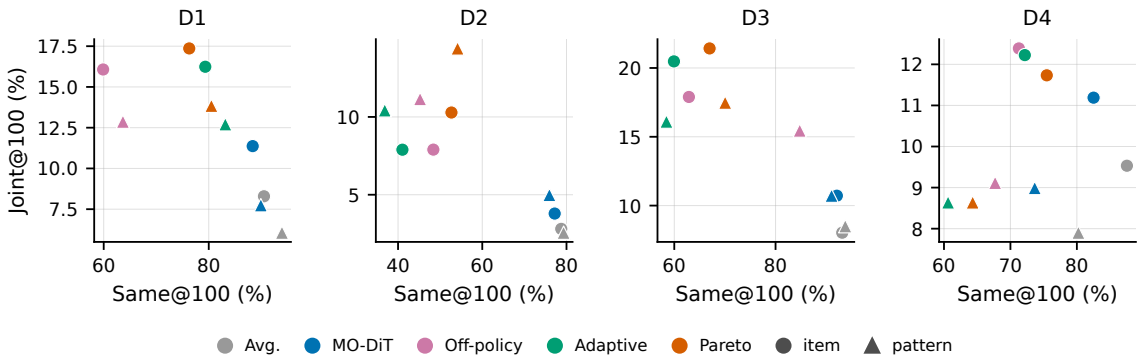

Figure 5: The attribute–pattern tradeoff, per domain. Each point places a model at its same-pattern share (`Same@K`, x) and intersection density (`Joint@K`, y) on the item (○) and pattern (△) holdouts. Average pooling and MO-DiT (SFT) keep high `Same@K` but low `Joint@K` (lower-right); the unconstrained off-policy preference policy raises `Joint@K` only by sacrificing `Same@K` (upper-left). On most domain–split cells the Pareto-filtered policy sits up and to the right of off-policy, improving the primary metric *and* pattern preservation—pushing the frontier outward rather than merely sliding along it. The D3 pattern-holdout is a partial exception where the higher `Joint@K` comes with some same-pattern-share cost, and D4 is the hard exception where the variants cluster.

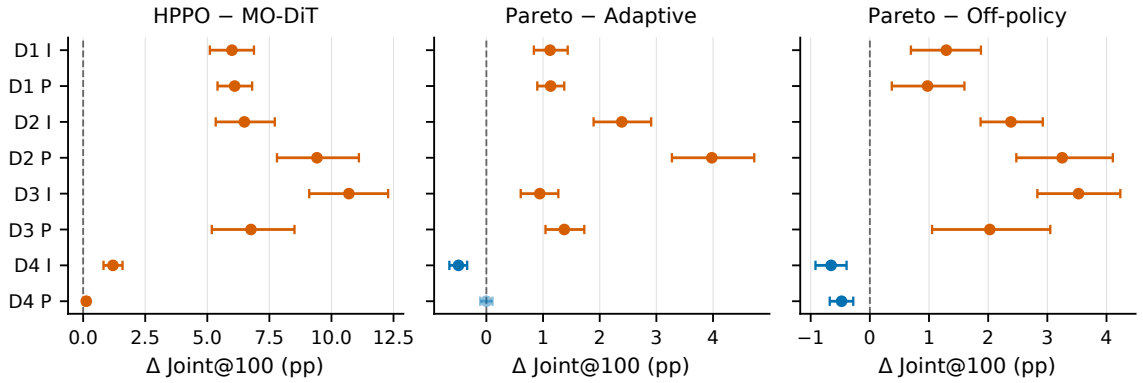

Figure 6: Paired-bootstrap `Joint@K` contrasts (B=10,000, n=500 shared cases). Each marker is the mean per-case difference with its 95% confidence interval; the dashed line is zero. Rows are the eight domain–split cells (I=item, P=pattern). *Left*: the best per-domain HPPO variant minus its MO-DiT initialization. *Middle* and *right*: the Pareto pair filter versus the iteration-only (adaptive) and single-round (off-policy) variants. Orange favors the first method in each title, blue the second; the one faded marker (Pareto vs. adaptive on D4 pattern) is the only interval that includes zero.

variance, the dominant noise source at this case count. The `Joint@K` improvement is also stable across independent training runs: re-training the iterated-Pareto preference stage with three random seeds—each re-samples the training cases, regenerates the candidates, and re-optimizes, while the evaluation set is held fixed—keeps every D1–D3 cell well above MO-DiT on both splits (Table 3).

5.2.3 Stage-wise Progress

Table 4 traces the primary `Joint@K` metric across the four training stages on the strict splits, all under the unified hardened protocol. The base initialization retrieves essentially no same-pattern positives; large-scale raw-sequence pretraining establishes a usable retrieval prior; the shared metric-ordered CPT adds a consistent increment over pretraining (the only exception is the small D4 pattern dip, within evaluation

Table 3: Seed robustness of the iterated-Pareto HPPO `Joint@K` (%, higher is better) across three independent training runs with different random seeds (each re-samples training cases, regenerates candidates, and re-optimizes; the evaluation set is held fixed), with MO-DiT for reference. Seed 1 is the run reported in Table 2. The improvement over MO-DiT is stable across seeds on D1–D3; D4 is marginal (cf. Appendix C; Table 2 reports the off-policy variant for D4).

Domain	Split	MO-DiT	Seed 1	Seed 2	Seed 3	Mean
D1	item	11.37	17.36	17.99	16.66	17.34
D1	pattern	7.74	13.84	15.94	14.43	14.74
D2	item	3.79	10.28	10.17	8.62	9.69
D2	pattern	4.99	14.40	13.70	11.32	13.14
D3	item	10.72	21.42	20.77	20.91	21.03
D3	pattern	10.72	17.48	15.88	15.79	16.38
D4	item	11.19	11.73	11.94	11.42	11.70
D4	pattern	8.99	8.64	9.96	9.58	9.39

Table 4: Stage-wise progress on the primary `Joint@K` metric. Base is the base initialization, Pre. is raw-sequence pretraining, CPT is the shared multi-domain metric-ordered CPT, SFT is domain-specific tail-centroid SFT from the shared CPT, and HPPO is the best per-domain preference variant. All stages are evaluated under the unified hardened protocol of Table 2. Values are `Joint@K` in percent.

Eval	Dom.	Base	Pre.	CPT	SFT	HPPO
Item	D1	0.00	5.99	7.52	11.37	17.36
Item	D2	0.00	1.87	2.26	3.79	10.28
Item	D3	0.02	6.44	7.47	10.72	21.42
Item	D4	0.00	6.86	7.30	11.19	12.39
Pattern	D1	0.00	4.58	5.03	7.74	13.84
Pattern	D2	0.00	1.98	3.09	4.99	14.40
Pattern	D3	0.01	6.75	7.61	10.72	17.48
Pattern	D4	0.00	5.82	5.32	8.99	9.12

noise); domain-specific tail-centroid SFT provides the main supervised jump; and HPPO further improves the metric on every domain and split. It increases monotonically across all four training stages on seven of the eight cells; the only non-monotone transition is the small D4 pattern dip at CPT ($5.82 \rightarrow 5.32$, within noise), which SFT more than recovers. The HPPO stage then improves the metric on all eight cells, with the D4 pattern gain a marginal tie.

5.3 Ablations and Analysis

5.3.1 Metric Ordering

The central hypothesis is that ordering same-pattern sequences by the proxy metric teaches a directional transformation. We therefore use controlled order ablations with identical data scale, split, model initialization, and training configuration: ascending density, random order, and descending density. These ablations use the single-domain CPT+SFT instantiation so that sequence order is the only varying factor. Table 5 reports the primary `Joint@K` metric on both item-holdout and pattern-holdout evaluation sets. Ascending density is the best row in every domain and split, and also exceeds the base initialization, the shared pretrained checkpoint, and continuation pretraining without SFT.

5.3.2 HPPO Ablations

Preference variants. Table 6 isolates the two design choices of HPPO—off-policy versus on-policy (hybrid-policy) candidate generation, and the Pareto pair filter—against the MO-DiT SFT initialization. Three observations hold across D1, D2, and D3. First, even the single-round off-policy run improves `Joint@K` over SFT on every cell. Second, iterating the hybrid policy (*adaptive*) adds a further gain on most cells, confirming that regenerating policy candidates from the improving policy is worthwhile once the guidance fan restores candidate diversity. Third, the Pareto pair filter is the largest and most consistent contributor: it is significantly above both off-policy and adaptive on all six D1/D2/D3 cells (Figure 6), and—unlike the

Table 5: Matched order ablation on the primary `Joint@K` metric (in percent), in the single-domain instantiation so that sequence order is the only varying factor. Item and Pattern denote item-holdout and pattern-holdout evaluation sets. Base is the base initialization, Pre. is the shared pretrained GR checkpoint, and CPT is continuation pretraining without centroid SFT. Because this table uses the single-domain instantiation under the light evaluation protocol, its absolute values are comparable only within the table (Asc. vs. Rand. vs. Desc.) and not across to Table 2 or Table 4, which report the shared multi-domain pipeline under the hardened protocol.

Eval	Dom.	Base	Pre.	CPT	Desc.	Rand.	Asc.
Item	D1	0.00	5.40	6.28	7.37	7.87	**9.25**
Item	D2	0.00	2.71	3.03	3.89	3.52	**4.11**
Item	D3	0.00	6.07	7.36	7.22	7.19	**11.06**
Item	D4	0.00	6.73	6.91	7.88	8.98	**11.40**
Pattern	D1	0.01	4.39	4.81	4.94	5.96	**7.07**
Pattern	D2	0.00	1.54	2.16	1.74	1.46	**2.40**
Pattern	D3	0.01	6.79	6.91	8.25	8.53	**9.80**
Pattern	D4	0.00	6.14	5.98	5.42	5.70	**9.32**

Table 6: HPPO preference-variant ablation on the primary `Joint@K` metric, best validation-selected checkpoint per split. The ablation makes two comparisons: off-policy versus on-policy (hybrid-policy) candidate generation, and the two hybrid-policy variants against each other (without vs. with the Pareto pair filter). SFT is the MO-DiT initialization (no preference stage). Off-pol. is the off-policy variant: candidates are collected once from the frozen SFT policy plus the static tail centroids. Hyb-adapt. and Hyb-pareto. are the hybrid-policy (on-policy) variants: each round regenerates the guidance-fan candidates from the *current* policy and blends them with the static tail centroids, Hyb-adapt. without and Hyb-pareto. with the tradeoff-aware pair filter. Best per row in bold. D4 is the one domain where the filter does not help. Values are `Joint@K` in percent.

Eval	Dom.	SFT	Off-pol.	Hyb-adapt.	Hyb-pareto
Item	D1	11.37	16.07	16.24	**17.36**
Item	D2	3.79	7.90	7.89	**10.28**
Item	D3	10.72	17.89	20.48	**21.42**
Item	D4	11.19	**12.39**	12.22	11.73
Pat.	D1	7.74	12.87	12.71	**13.84**
Pat.	D2	4.99	11.15	10.42	**14.40**
Pat.	D3	10.72	15.45	16.10	**17.48**
Pat.	D4	8.99	**9.12**	8.64	8.64

unconstrained variants—on most cells it raises `Joint@K` *while keeping same-pattern share high* (D1 pattern `Same@K` 80.5% for Pareto vs. 63.7% for off-policy; the one exception is D3 pattern-holdout, where the share dips below off-policy). This is the intended behavior: the filter removes purity-sacrificing pairs, so the policy improves the intersection by a Pareto-dominant move rather than by drifting toward global attribute retrieval. D4 is the exception—the filter neither helps nor hurts there and the simpler off-policy run is reported—because D4 leaves little intersection headroom for any variant to exploit (Appendix C).

Candidate complementarity. A second ablation on D1 toggles the candidate families inside the off-policy regime, with all rows sharing the same cases and online labels. The full hybrid set (static tail centroids plus policy samples) reaches `Joint@K` 16.1% item / 12.9% pattern, above static-only (14.7/11.1) and policy-only (13.4/10.1); Table 7 lists the precise values. Both families contribute, so the hybrid design is justified. This ablation also motivated the guidance fan: after point-target tail-centroid SFT the conditional generator is nearly deterministic, so independently sampled policy queries collapse to one effective candidate; fanning the guidance scale of the current policy restores the diversity that makes iteration and the pair filter effective.

On-policy iteration at fixed compute. The off-policy-vs-adaptive comparison above is not step-matched. To isolate the benefit of regenerating candidates on-policy from the sheer amount of training, we fix the total preference budget at 100 steps on D1 and vary only the number of on-policy rounds. Table 8

Table 7: Candidate-pool complementarity (D1, off-policy regime, hardened, best checkpoint). Static = four tail centroids; policy = four guidance-fan policy samples; hybrid = both families. Values are `Joint@K` in percent.

Candidate pool	Item	Pattern
Static-only (4 tail centroids)	14.68	11.05
Policy-only (4 fan samples)	13.40	10.08
Hybrid (static + policy)	**16.07**	**12.87**

Table 8: On-policy iteration at fixed total compute (D1, hardened): same data and the same 100 total preference-training steps, varying only the number of on-policy candidate-regeneration rounds; the 1×100 single-round row is the off-policy round-1 run. Values are `Joint@K` in percent.

Schedule (rounds×steps)	Item	Pattern
1×100 (single round)	12.96	9.14
2×50	13.37	9.57
4×25	**13.74**	**10.02**

Table 9: Best-of-N oracle on held-out cases (`Joint@K`, 1,000 cases/split). SFT-oracle and HPPO-oracle pick the best of eight candidates per case in hindsight; HPPO-single is the policy's served query. On D1/D2/D3 the HPPO single query exceeds the SFT eight-candidate oracle. Values are `Joint@K` in percent.

Eval	Dom.	SFT-oracle	HPPO-single	HPPO-oracle
Item	D1	10.20	**12.60**	15.70
Item	D2	1.90	**5.60**	6.80
Item	D3	9.70	**15.70**	19.10
Item	D4	11.90	12.60	13.70
Pat.	D1	10.50	**13.50**	17.10
Pat.	D2	1.70	**8.00**	9.20
Pat.	D3	8.00	**12.80**	15.90
Pat.	D4	8.20	7.20	8.60

shows `Joint@K` rising monotonically with the number of rounds on both splits at identical data and identical total steps, so the gain comes from on-policy candidate regeneration rather than from more optimization.

5.3.3 Best-of-N Oracle: Generation vs. Selection

A natural question is whether HPPO merely learns to select a good candidate from the menu it already had at initialization, or whether it learns to *generate* better queries. We answer it with a best-of-N oracle analysis on 1,000 held-out cases per split. For a fixed policy we form the same eight candidates used in training (four static tail centroids and four guidance-fan policy samples), label each with true online `Joint@K`, and compare the policy's single served query against per-case oracles that pick the best candidate in hindsight. Table 9 reports the result. On D1, D2, and D3 the HPPO policy's *single* generated query already exceeds the SFT policy's full eight-candidate oracle: HPPO has moved the generator itself, not just improved selection over the initialization's menu. The effect is starkest on D2, where the static centroids are nearly useless (`Joint@K` $\leq 2\%$) yet the HPPO single query reaches 5.6% (item) and 8.0% (pattern). An eight-candidate oracle over the HPPO policy adds a further 15–27% relative, indicating real headroom for a future inference-time reranker. D4 is again the exception, where generation and selection are close—reflecting its limited headroom.

6 Conclusion

We presented MO-DiT+HPPO, a staged training framework for attribute-seeking and pattern-preserving generative retrieval. Starting from large-scale raw-sequence pretraining, MO-DiT converts sparse online metric labels into metric-ordered in-pattern trajectories, trains one shared multi-domain continuation-pretrained retriever and a lightweight domain-specific tail-centroid SFT, and HPPO aligns the final query distribution with the online intersection metric through iterated reference-anchored preference optimization over hybrid

candidates, using a tradeoff-aware Pareto pair filter to raise the attribute metric without sacrificing pattern purity. Across four large-scale domains, the metric-ordered stages improve the primary intersection metric over a strong pretrained retriever, and HPPO further improves it, with paired-bootstrap-significant gains on seven of the eight domain-split cells and a marginal tie on the hardest domain's pattern split. Controlled order ablations, a preference-variant ablation, and the candidate-policy ablation explain where the gains come from and show that the Pareto filter is the component that keeps the gains on the favorable side of the attribute–pattern tradeoff.

Limitations. MO-DiT+HPPO requires a frozen item-embedding space and a production attribute scorer, and it derives the fine-grained pattern by clustering those embeddings—a scalable but approximate, rather than human-labeled, notion of a pattern, which we mitigate with held-out-cluster evaluation and a sampled qualitative check that the clusters agree with manual pattern judgments (Section 3). The metric predictor is used only to order the training sequences and not for final evaluation, so its errors affect data construction rather than the reported metrics. A fully public benchmark and stronger purity-constrained preference objectives are natural directions for future work.

Broader impact. By finding more same-pattern examples of an attribute, the method can improve quality, safety, and integrity workflows, but the same capability could amplify biased or incorrectly defined attribute scorers; deployment should pair it with scorer auditing, human review for sensitive domains, and monitoring for pattern drift or over-retrieval of narrow content groups.

References

Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations*, 2024.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866, 2023a.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, 2023b.

Geonmo Gu, Sanghyuk Chun, Wonjae Kim, Heejae Jun, Yoohoon Kang, and Sangdoo Yun. CompoDiff: Versatile composed image retrieval with latent diffusion. *Transactions on Machine Learning Research (TMLR)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, 2018.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. *arXiv preprint arXiv:2505.05470*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Yongxin Ni, Yu Cheng, Xiangyan Liu, Junchen Fu, Youhua Li, Xiangnan He, Yongfeng Zhang, and Fajie Yuan. A content-driven micro-video recommendation dataset at scale. *arXiv preprint arXiv:2309.15379*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, 2023.

Joseph J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton (ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice-Hall, 1971.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*, volume 35, pp. 21831–21843, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Conference on Computer Vision and Pattern Recognition*, 2024.

Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 832–841, 2023.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 24247–24261, 2023.

Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10586–10613, 2024.

This appendix collects architecture and configuration details (A–B), full per-domain results with confidence intervals (C), best-of-N, candidate, and guidance analyses (D), metric-predictor validation (E), the earlier full-protocol single-domain results (F), negative and diagnostic results (G), and reproducibility and anonymization notes (H).

A Architecture and Implementation Details

Unified diffusion transformer. The generator is a single diffusion-transformer backbone trained with a flow-matching objective. The condition items and the noised target token(s) form one sequence that is processed by one transformer; there is no separate condition encoder feeding a pooled vector to a generation head. Each item contributes one token (its embedding projected to the hidden width plus a positional encoding and a learned role marker that distinguishes condition from target positions); a binary condition mask flags valid positions.

Structured attention mask. The attention pattern has four blocks. (i) Condition–condition is causal (lower triangular), so the conditioning prefix is autoregressive. (ii) Condition–target is empty: condition positions never attend to targets, so the condition can be encoded once and cached. (iii) Target–condition is prefix-visible: each target attends to the condition prefix up to its own position. (iv) Target–target is diagonal: a target attends only to itself, so several targets denoise/generate independently in a single pass (used by the dense-prefix CPT objective; SFT uses a single target). Because each target attends to the full condition sequence rather than a pooled summary, fine-grained information from the seed items is preserved; this is the design difference from a lightweight single-vector-conditioned generation head.

Flow matching, normalization, and guidance. The network predicts a flow-matching velocity on the L_2-normalized d-dimensional embedding sphere, using the spherical (Riemannian) flow-matching instantiation (Chen & Lipman, 2024) with geodesic paths and a Jacobi curvature weighting on the velocity loss, rather than straight-line rectified flow; the diffusion timestep is injected at target positions through adaptive layer normalization (AdaLN). Classifier-free guidance is implemented as a condition/attention dropout during training, and the serving guidance scale is 3.0 (validated near-optimal in Appendix D). Inference—spherical ODE integration with key/value caching of the condition prefix, then approximate-nearest-neighbor search with no quantization or identifier decoding—is described in the main-text Inference subsection.

B Training and Evaluation Configuration

Default constants. Unless stated otherwise, the retrieval depth is $K = 100$, the embedding dimension $d = 1536$, the metric-ordered sequence length $L = 200$, and the input prefix $m = 150$ (so the high-metric tail has $L - m = 50$ items).

Stages. All models initialize from a pretrained continuous retriever obtained by large-scale raw-sequence pretraining over naturally ordered item sequences. The shared multi-domain metric-ordered CPT mixes the four domains with repetition weights D1×2, D2×1, D3×1, D4×3, one epoch over the pool: global batch 4096, per-device batch 16, gradient accumulation 4, learning rate 5×10^{-4}, warmup ratio 0.01, minimum learning-rate ratio 0.1 (4806 steps). Domain-specific tail-centroid SFT initializes from the shared CPT: global batch 4096, per-device batch 64, gradient accumulation 1, learning rate 5×10^{-5}, warmup ratio 0.01, tail-centroid target length $L - m$; D1/D2/D3 one epoch (377/1452/2250 steps), D4 two epochs (234 steps).

HPPO. Candidates per case are four static tail centroids (top-k averages over $\mathcal{K} = \{10, 25, 35, 50\}$) and four policy samples from a guidance-scale fan over $\mathcal{S} = \{1, 2, 3, 6\}$. The reward is true online `Joint@K`. The loss is pairwise reference-anchored (DPO-style) with the frozen SFT model as reference, margin $\delta = 0.03$ on the reward gap, and a tail-centroid anchor of weight 0.1; learning rate 1×10^{-5}. The iterated variants run short rounds (about one epoch of the kept pairs, roughly 100 steps), regenerate the policy fan from the current policy each round, select the round checkpoint on a disjoint validation slice (held-out tail cases

generated at the serving scale and labeled with true online `Joint@K`), and stop when a round does not improve the validation metric.

Evaluation protocols. The hardened protocol (main results) uses up to 1,000 AUC cases and 500 cluster-stratified density cases per split, with all models scored on one deterministic shared case sample (paired). The light protocol (order/CPT ablations) uses 50 density cases per split; differences below ≈ 1 pp there are within noise. Significance is paired bootstrap over the shared cases, $B = 10{,}000$ resamples of the per-case differences.

Feasibility. For one domain, 50,000 cases $\times$ 8 candidates = 400,000 candidate queries are labeled by real retrieval in under an hour on one machine (≈ 127 queries/s, recall-dominated). Margin filtering keeps $\approx 12\%$ of cases; a preference round trains in minutes on eight GPUs; a full iterated run is a few rounds.

C Full Per-Domain Results with Confidence Intervals

Tables 10, 11, 12, and 13 give every variant on every split with all five metrics; the `Joint@K` cell carries its 95% paired-bootstrap confidence interval as a subscript. Pre. GR is not in the bootstrap set, so it has no interval. For D4, the Pareto-strict run's auxiliary metrics were not separately archived (D4 reports the off-policy variant in the main table); its `Joint@K`/`Same@K` are shown.

Table 10: D1, hardened protocol. AUC / Attr / Same / Joint$_{[95\%\ \mathrm{CI}]}$ / Cond; densities in % (AUC excepted).

Eval	Method	AUC	Attr	Same	Joint	Cond
Item	Avg.	0.4793	12.63	90.51	$8.29_{[7.30,9.30]}$	87.65
Item	Pre. GR	0.4715	11.86	67.16	5.99	64.73
Item	MO-DiT	0.5278	18.11	88.37	$11.37_{[10.10,12.70]}$	84.51
Item	Off-policy	0.6229	47.82	59.89	$16.07_{[14.60,17.60]}$	50.75
Item	Adaptive	0.5882	31.38	79.34	$16.24_{[14.70,17.70]}$	69.23
Item	Pareto	0.5992	35.34	76.29	$17.36_{[15.80,18.90]}$	65.90
Item	Pareto-ε	0.6184	45.48	64.92	$\mathbf{17.68}_{[16.10,19.30]}$	54.22
Pat.	Avg.	0.4741	8.80	93.94	$6.06_{[5.30,7.00]}$	92.17
Pat.	Pre. GR	0.4659	8.58	73.30	4.58	68.70
Pat.	MO-DiT	0.5171	11.55	89.92	$7.74_{[6.80,8.70]}$	83.92
Pat.	Off-policy	0.6178	38.79	63.66	$12.87_{[11.80,14.00]}$	53.78
Pat.	Adaptive	0.5831	22.68	83.15	$12.71_{[11.60,13.90]}$	73.78
Pat.	Pareto	0.5952	26.18	80.50	$13.84_{[12.70,15.10]}$	70.96
Pat.	Pareto-ε	0.6144	36.57	69.61	$\mathbf{14.97}_{[13.80,16.20]}$	58.56

Table 11: D2, hardened protocol; densities in %.

Eval	Method	AUC	Attr	Same	Joint	Cond
Item	Avg.	0.4977	3.89	78.75	$2.81_{[2.40,3.20]}$	73.82
Item	Pre. GR	0.4867	3.57	57.69	1.87	54.24
Item	MO-DiT	0.5413	5.36	77.20	$3.79_{[3.20,4.50]}$	73.59
Item	Off-policy	0.5909	26.71	48.38	$7.90_{[6.70,9.20]}$	41.65
Item	Adaptive	0.5994	32.87	41.04	$7.89_{[6.70,9.20]}$	34.12
Item	Pareto	0.6068	22.47	52.68	$\mathbf{10.28}_{[8.90,11.70]}$	46.00
Item	Pareto-ε	0.6007	35.23	37.80	$8.82_{[7.50,10.20]}$	31.43
Pat.	Avg.	0.5013	3.77	79.32	$2.56_{[2.20,3.00]}$	77.24
Pat.	Pre. GR	0.4886	3.66	58.11	1.98	60.70
Pat.	MO-DiT	0.5426	7.16	75.94	$4.99_{[4.00,6.20]}$	72.49
Pat.	Off-policy	0.5965	33.28	45.23	$11.15_{[9.60,12.80]}$	41.94
Pat.	Adaptive	0.6035	38.88	36.85	$10.42_{[8.90,12.10]}$	34.12
Pat.	Pareto	0.6087	27.74	54.14	$\mathbf{14.40}_{[12.50,16.40]}$	48.64
Pat.	Pareto-ε	0.6031	39.60	34.98	$11.50_{[9.80,13.30]}$	33.17

The off-policy checkpoints show the characteristic drift of unconstrained preference training (Figure 7): raw attribute density keeps rising while same-pattern share falls and `Joint@K` peaks early and then declines. This is why preference rows use early validation-selected checkpoints.

Table 12: D3, hardened protocol; densities in %.

Eval	Method	AUC	Attr	Same	Joint	Cond
Item	Avg.	0.4565	8.97	93.17	$8.03_{[7.30,8.80]}$	92.56
Item	Pre. GR	0.4473	9.60	70.62	6.44	70.01
Item	MO-DiT	0.5182	12.46	92.11	$10.72_{[9.70,11.80]}$	91.30
Item	Off-policy	0.5948	41.46	62.89	$17.89_{[16.20,19.60]}$	57.64
Item	Adaptive	0.5958	47.05	59.93	$20.48_{[18.60,22.40]}$	53.55
Item	Pareto	0.5916	42.08	66.97	$\mathbf{21.42}_{[19.60,23.30]}$	59.85
Item	Pareto-ε	0.5941	45.45	62.58	$21.20_{[19.30,23.10]}$	55.66
Pat.	Avg.	0.4573	9.10	93.77	$8.50_{[7.90,9.10]}$	95.19
Pat.	Pre. GR	0.4525	9.14	73.56	6.75	75.86
Pat.	MO-DiT	0.5054	11.77	91.06	$10.72_{[9.90,11.60]}$	92.74
Pat.	Off-policy	0.5600	20.46	84.78	$15.45_{[14.10,16.90]}$	81.79
Pat.	Adaptive	0.5845	39.93	58.45	$16.10_{[14.40,18.00]}$	47.79
Pat.	Pareto	0.5810	32.56	70.05	$\mathbf{17.48}_{[15.80,19.30]}$	59.53
Pat.	Pareto-ε	0.5835	36.56	63.65	$16.99_{[15.20,18.90]}$	53.19

Table 13: D4, hardened protocol; densities in %. The reported main-table variant for D4 is off-policy.

Eval	Method	AUC	Attr	Same	Joint	Cond
Item	Avg.	0.5631	16.21	87.53	$9.53_{[8.60,10.50]}$	82.93
Item	Pre. GR	0.5485	14.42	68.04	6.86	66.90
Item	MO-DiT	0.6093	21.35	82.51	$11.19_{[10.20,12.20]}$	75.81
Item	Off-policy	0.6561	30.07	71.27	$\mathbf{12.39}_{[11.30,13.50]}$	60.89
Item	Adaptive	0.6466	28.96	72.17	$12.22_{[11.20,13.30]}$	62.92
Item	Pareto	–	–	75.47	$11.73_{[10.70,12.80]}$	–
Item	Pareto-ε	0.6353	26.10	76.05	$11.81_{[10.80,12.90]}$	66.99
Pat.	Avg.	0.5251	17.00	80.23	$7.90_{[7.40,8.50]}$	73.02
Pat.	Pre. GR	0.5117	14.69	66.44	5.82	62.13
Pat.	MO-DiT	0.5871	22.67	73.66	$8.99_{[8.40,9.60]}$	65.16
Pat.	Off-policy	0.6216	28.11	67.70	$\mathbf{9.12}_{[8.60,9.70]}$	56.59
Pat.	Adaptive	0.6505	31.86	60.62	$8.64_{[8.10,9.20]}$	47.85
Pat.	Pareto	–	–	64.32	$8.64_{[8.10,9.20]}$	–
Pat.	Pareto-ε	0.6326	28.61	64.94	$8.72_{[8.20,9.30]}$	52.60

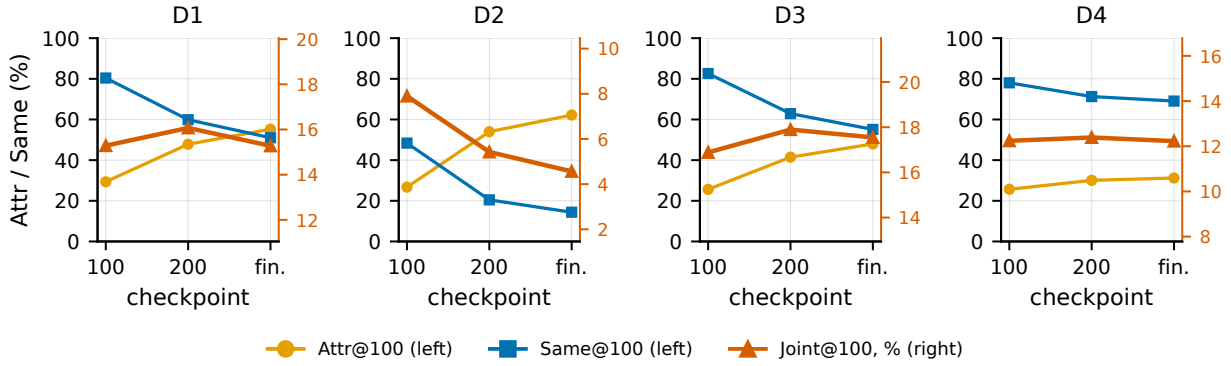

Figure 7: Off-policy checkpoint drift on the item split. As training continues (at 100 → 200 →final steps), `Attr@K` (left axis) rises while `Same@K` (left axis) falls, so the intersection `Joint@K` (right axis, magnified) peaks early and then declines—motivating early validation-based checkpoint selection. D4 drifts least.

The D4 exception. D4 departs from the other three domains in the same direction throughout: the preference variants cluster (Table 6), the Pareto filter neither helps nor hurts, the pattern-holdout gain is a marginal tie (Figure 6), and generation barely beats selection in the best-of-N oracle (Table 14). One observation accounts for all of these: on D4 every variant lands in a narrow `Joint@K` band (item 11.2–12.4%, pattern 8.6–9.1%), whereas on the other domains the band spans roughly 6–11 pp from SFT to the best variant. There is little intersection headroom for the preference stage to exploit, so (i) unconstrained and

Pareto-filtered updates reach nearly the same point, leaving the filter no purity-sacrificing gain to correct, and (ii) selecting among candidates adds little over generating one. This is consistent with D4 being by far the smallest and most heavily reused domain (Table 1: 478K rows, 500 clusters, ∼31× positive reuse, two training epochs), which bounds how far attribute density can be pushed within a pattern, and with this attribute being intrinsically harder to separate in the embedding space. We therefore report the simpler off-policy variant for D4 in Table 2.

D Best-of-N Oracle, Candidate, and Guidance Analyses

Best-of-N oracle (Table 14). For a fixed policy we score all eight candidates with true online `Joint@K` and compare the served single query against per-case oracles (best static candidate, best policy candidate, best of all eight). On D1/D2/D3 the HPPO single query exceeds the SFT eight-candidate oracle; an oracle over the HPPO candidates adds a further 15–27% relative, indicating headroom for an inference-time reranker. D4 is the exception.

Table 14: Best-of-N oracle on 1,000 held-out cases/split (`Joint@K` in %). "or" = per-case oracle. The two static-or columns coincide by construction, as the static tail centroids do not depend on the policy.

Eval	Dom	SFT static-or	SFT 8-or	HPPO single	HPPO static-or	HPPO gr-or	HPPO 8-or
Item	D1	8.70	10.20	**12.60**	8.70	15.30	15.70
Pat.	D1	9.30	10.50	**13.50**	9.30	16.40	17.10
Item	D2	1.60	1.90	**5.60**	1.60	6.50	6.80
Pat.	D2	1.30	1.70	**8.00**	1.30	9.10	9.20
Item	D3	8.80	9.70	**15.70**	8.80	18.00	19.10
Pat.	D3	6.20	8.00	**12.80**	6.20	15.60	15.90
Item	D4	10.90	11.90	12.60	10.90	13.40	13.70
Pat.	D4	7.70	8.20	7.20	7.70	7.70	8.60

Candidate-policy ablation (D1, off-policy regime, hardened best checkpoint). Static + policy reaches `Joint@K` 16.07% item / 12.87% pattern, above static-only (14.68/11.05) and policy-only (13.40/10.08); both candidate families contribute. The round-1 checkpoint (at 100 steps) gives 12.96/9.14.

Guidance-scale calibration (Figure 8). Mean true online `Joint@K` per fan scale, by domain and preference round. Scales 2–3 are consistently best and 1 is worst; on drifted late rounds higher scales help (same-pattern share rises with scale). The serving default 3.0 is near-optimal.

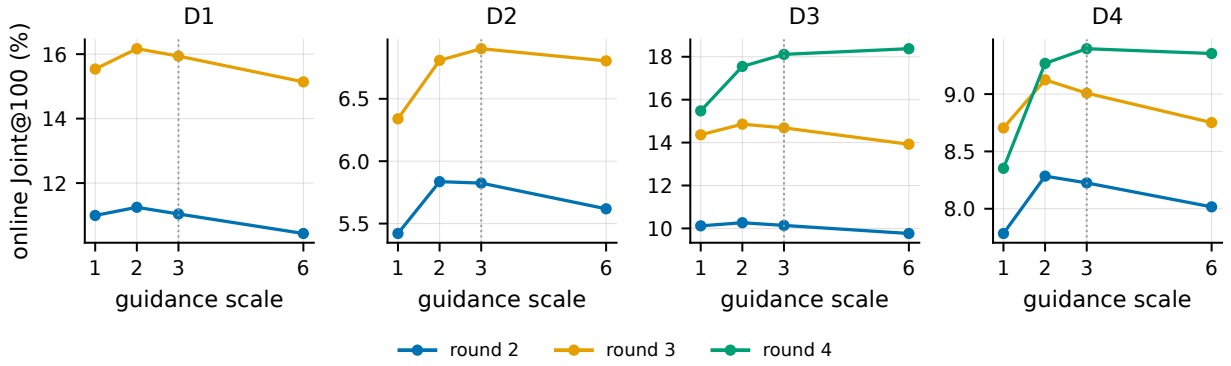

Figure 8: Guidance-fan calibration: mean true online `Joint@K` of the policy candidate at each guidance scale, by domain and preference round. The dotted line marks the serving default (s=3). Scales 2–3 are consistently near the peak.

Hyperparameter sensitivity (D1). We sweep the two main HPPO hyperparameters—the reward margin δ and the preference temperature β—around their defaults ($\delta = 0.03$, $\beta = 0.05$) on D1, holding everything

Table 15: HPPO hyperparameter sensitivity on D1: primary `Joint@K` metric (in percent) under the hardened protocol, varying the reward margin δ and the preference temperature β around their defaults. Single-domain runs; absolute values are not directly comparable to the main tables.

Hyperparameter	Value	Item	Pattern
Margin δ	0.01	17.82	15.77
	0.03 (default)	**17.99**	**15.94**
	0.05	17.21	15.01
Temperature β	0.025	17.13	15.30
	0.05 (default)	**17.99**	**15.94**
	0.10	17.54	15.47

else fixed and reusing the cached round-one labels. Table 15 reports the primary `Joint@K` metric under the hardened protocol. It is stable across both knobs (within about one percentage point), and the defaults sit at or near the optimum, so the reported gains do not depend on hyperparameter tuning. These are single-domain runs, so absolute values are not directly comparable to the main tables.

E Metric Predictor Validation

Table 16 reports the full metric-predictor validation: the strict pattern-out and super-pattern-out splits stay strong, showing that the frozen embeddings carry density-ranking signal beyond memorized cluster identity, not just a memorized per-cluster mean.

Table 16: Metric (attribute-density) predictor validation. AUC20/80 evaluates whether the predictor separates the lowest-density 20% and highest-density 20% anchors. Lift is Top10 true density divided by Bottom10 true density. D4 is shown on item-hash and pattern-out only.

Domain	Split	AUC20/80	Spearman	Top10 (%)	Bottom10 (%)	Lift
D1	item-hash	0.959	0.663	39.40	1.75	22.50
D1	pattern-out	0.929	0.606	35.60	2.26	15.78
D1	super-pattern	0.900	0.552	25.00	1.95	12.83
D2	item-hash	0.945	0.647	19.90	1.07	18.53
D2	pattern-out	0.951	0.661	21.00	0.99	21.10
D2	super-pattern	0.920	0.591	15.50	1.29	12.01
D3	item-hash	0.957	0.659	45.40	3.18	14.28
D3	pattern-out	0.934	0.621	40.90	3.96	10.33
D3	super-pattern	0.882	0.530	28.60	3.30	8.67
D4	item-hash	0.993	0.800	58.50	1.21	48.38
D4	pattern-out	0.973	0.752	42.90	1.67	25.71

F Original Full-Protocol Results (single-domain instantiation)

These are the earlier formal results under a full (non-strict, larger-case) evaluation of the single-domain instantiation. They agree with the strict-protocol conclusions but are not directly comparable in magnitude to the hardened tables. The stage names match the main tables, except that CPT here is the per-domain continuation pretraining of this single-domain instantiation rather than the shared multi-domain CPT.

Headline. Metric-ordered CPT+SFT improves `Joint@K` over the pretrained retriever on all four domains: D1 $5.07 \rightarrow 10.57$, D2 $4.90 \rightarrow 7.18$, D3 $6.37 \rightarrow 12.05$, D4 $6.49 \rightarrow 9.41$ (all %; +44.9% to +108.4% relative); per-stage detail for D2 and D3 is in Table 17.

Target design (D4, full protocol). The oracle tail centroid reaches `Joint@K` 9.98% at conditional purity 95.53%; oracle top-1/top-5 targets reach higher attribute density (17.38/17.47) but lower `Joint@K` (6.34/6.94) because conditional purity collapses (70.59/72.96). The learned MO-DiT model reaches `Joint@K` 9.41% at

Table 17: D2 and D3, full-protocol single-domain instantiation (AUC; Attr / Cond / Joint / Same in %).

Model	AUC	Attr	Cond	Joint	Same
D2 Pre. GR	0.4898	9.53	60.45	4.90	62.02
D2 CPT	0.5026	9.68	65.46	5.30	64.89
D2 SFT	0.5439	11.24	77.70	**7.18**	81.84
D3 Pre. GR	0.4457	9.27	68.42	6.37	69.67
D3 CPT	0.4727	9.89	72.04	7.08	73.33
D3 SFT	0.5150	12.76	94.82	**12.05**	97.12

the highest conditional purity (99.54%). All values are percentages. The tail centroid is therefore the right supervision target: sharper top-k targets trade the intersection away (Table 18).

Table 18: D4, full-protocol single-domain instantiation, and target design (AUC; Attr / Cond / Joint / Same in %).

Model	AUC	Attr	Cond	Joint	Same
Pre. GR	0.5129	10.99	–	6.49	–
MO-DiT (SFT)	0.5731	14.02	99.54	**9.41**	98.20
target centroid	0.6498	15.85	95.53	9.98	96.86
target top-1	0.6217	17.38	70.59	6.34	66.06
target top-5	0.6681	17.47	72.96	6.94	77.45

G Negative and Diagnostic Results

We record results that did not enter the main paper, both for completeness and to motivate the final recipe.

Unconstrained off-policy preference (negative). An earlier 200k-case off-policy/online preference suite without the Pareto filter and with longer training was *below* the SFT initialization on `Joint@K` on both holdouts; hybrid/online variants raised attribute density and AUC but collapsed conditional purity (reward hacking toward global attribute retrieval). This motivated the validation-based early stop, the fixed SFT reference, and the Pareto pair filter.

Local/proxy reward (negative). Pure-listwise training against a local approximate-retrieval bank, and a selected-case proxy bank, were unreliable: local reward over-amplified some candidates (e.g. Rocchio-style vectors) relative to true online retrieval, and downstream metrics degraded. True online `Joint@K` labels were necessary.

Guidance fan is necessary for the filter (negative). Replacing the guidance fan with duplicate policy samples (no diversity) under the Pareto filter produced validation scores below baseline and early-stopped at SFT: the candidate diversity of the fan is what makes the pair filter effective.

The eps knob is not a clean curve (diagnostic). Sweeping the purity-tolerance ε to trace an attribute–purity frontier does not yield a clean monotone curve, because each run early-stops at a different round and the round count is itself a frontier slider; ε and training amount are entangled. We therefore report only the strict (ε=0, "Pareto") and ε=0.02 ("Pareto-ε") settings rather than a frontier; a tradeoff-parameter-conditioned policy is the proper route to a controllable frontier and is left to future work.

H Reproducibility and Anonymization

The four domains are anonymized because they use our internal data. The method depends only on: frozen item embeddings; a binary or real-valued attribute scorer; a vector index for nearest-neighbor retrieval; and sparse online recall-density labels for predictor training—all common components of large-scale item-embedding retrieval systems. Sequence construction, metric-ordered CPT, tail-centroid SFT, HPPO, and the evaluation protocol are described in the main text and Appendices A–B. All reported numbers derive

from fixed deterministic case samples; the significance analysis covers evaluation-sampling variance via paired bootstrap and training-run variability via three independent re-runs of the preference stage with different random seeds (Table 3).