# MaLaCo: Masked Latent Completion for Efficient Generative Recommendation

**Anonymous authors**
Paper under double-blind review

## Abstract

Semantic-ID (SID) recommenders retrieve items by generating short discrete identifiers, but masked-diffusion decoding repeatedly processes every beam hypothesis through the full history backbone. We introduce *masked latent completion* (MaLaCo), which factorizes retrieval into a single bidirectional history-completion pass followed by trie-constrained SID generation with a small autoregressive head. Beam-dependent computation is therefore confined to the head. On one A100, at beam width 50, MaLaCo reduces amortized generation cost from $13.097$ to $0.526$ ms/query at batch 20 ($24.9\times$) and single-request latency from $24.453$ to $9.984$ ms ($2.4\times$), despite a higher total parameter count relative to LLaDA-Rec, a leading generative retrieval baseline. Across four datasets, MaLaCo improves Recall@$k$ over LLaDA-Rec at every reported cutoff $k \geq 10$, with larger gains at deeper cutoffs. It trades Recall@1 for stronger deep recall on Industrial & Scientific, Musical Instruments, and Video Games, while also improving Recall@1 on Arts, Crafts & Sewing. A four-seed $2 \times 2$ architecture–encoder study confirms that this depth-dependent advantage persists under either item encoder, while the stronger encoder consistently benefits MaLaCo. We further introduce a collision-aware SID construction pipeline combining balanced residual quantization, last-level deduplication, learned candidate refinement, and catalog-level assignment. Across all four catalogs, it reduces the excess-collision rate from $18.12\%$–$31.04\%$ to at most $0.0122\%$. On an item-arrival partition of Arts, the method inserts the newest quarter of the catalog without changing existing identifiers or introducing new collisions. After recommender retraining, the extended vocabulary obtains $0.04526$ Recall@10 versus $0.04598$ after complete vocabulary refitting, a $1.6\%$ relative shortfall. Together, these results position MaLaCo as an efficient, high-recall candidate generator with collision-aware and stably extensible semantic identifiers.

## 1 Introduction

Generative retrieval replaces catalog-wide scoring with sequence generation: each item is assigned a semantic identifier (SID), and a model emits the next item's identifier directly (Tay et al., 2022; Rajput et al., 2023; Ju et al., 2025). Shared prefixes place related items near one another, while a catalog trie prevents invalid outputs. This formulation is attractive for recommendation: it compresses item identities into a small vocabulary and naturally returns a candidate set. However, SIDs come from a quantizer fit to a fixed catalog, so new items cannot be indexed without re-training, and a consolidated pipeline carries its own serving costs.

LLaDA-Rec (Shi et al., 2025), which leads current SID and Item-ID based generation methods in recall, uses masked discrete diffusion to predict an SID with bidirectional attention and adaptive generation order. Its beam search operates inside the denoising model: every surviving hypothesis re-enters the network at every denoising step. If $T$ is the step count and $B$ the beam width, the history backbone processes $TB$ long sequences per query. Batching improves utilization, but does not remove this arithmetic. Moreover, parallel denoising conflicts with the coarse-to-fine structure of conventional SIDs, in which later tokens refine the category established by earlier ones. As a workaround, LLaDA-Rec adopts balanced SIDs that distribute information equally across positions, sacrificing prefix interpretability, efficient prefix-based retrieval, semantic-neighbor lookup, and compatibility with autoregressive generation. We ask whether one can retain an ordered, catalog-constrained SID while making the expensive history representation independent of discrete search.



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

**Masked latent completion (MaLaCo)** separates history encoding from SID generation. It feeds continuous representations of the user's recent items, together with a blank slot for the next item, into a bidirectional Transformer. A single forward pass produces a contextual representation of the blank slot. A small autoregressive head then uses this representation to generate the next item's short SID. Trie-constrained beam search ensures that every generated SID corresponds to a catalog item. Importantly, beam search runs only in the small SID head, so the history encoder is evaluated just once. Moreover, because the encoded state changes only when the user's history changes, it can be cached and reused across requests with the same history.

In MaLaCo, the continuous item representations play two roles: directly as inputs to the history encoder and to construct the SIDs predicted by the autoregressive head. In LLaDA-Rec, by contrast, the recommendation model consumes only the quantized SIDs, so the text encoder affects retrieval only through SID construction. A stronger text encoder may therefore benefit the two architectures differently. To separate these effects, we evaluate all four combinations of MaLaCo and LLaDA-Rec with sentence-T5-base and BGE-Multilingual-Gemma2 using four seeds (Ni et al., 2021; Chen et al., 2025). Under both encoders, MaLaCo performs worse at rank 1 but increasingly better as the retrieval cutoff $k$ grows. Replacing sentence-T5 with BGE-Multilingual-Gemma2 consistently improves MaLaCo's recall, while providing no consistent benefit to LLaDA-Rec.

The SID vocabulary itself is a separate problem. Ideally the SIDs should preserve semantic structure, so that similar items share prefixes, but each catalog item should also have a unique full SID. While these objectives are related, in practice most current SID methodologies introduce some level of collision. To deal with collisions while maintaining interpretability in the SID encoding, we propose constructing the vocabulary in several steps. Balanced residual $k$-means first creates a coarse-to-fine semantic hierarchy. A last-level Sinkhorn deduplication step then reassigns the final code to remove most repeated tuples while preserving the coarser SID prefixes. A collision-aware autoencoder learns alternative SID candidates for each item, after which a confidence-ordered catalog assignment selects unclaimed candidates to resolve the remaining conflicts. The same assignment procedure also supports *stable insertion*: it reserves all existing SIDs and assigns unused candidates to new items without changing existing identifiers. However, assigning a valid SID to a new item does not mean that a frozen recommendation model can retrieve it (Peng et al., 2026). Our experiments therefore evaluate vocabulary extension and recommendation of new items as two separate problems. Our contributions are:

**(1) Beam-independent history encoding.** MaLaCo encodes the interaction history once and confines trie-constrained beam search to a small SID head. At beam width $50$ on a single A100, MaLaCo requires $0.5263$ ms per query at batch size $20$, compared with $13.097$ ms for our LLaDA-Rec reproduction, a $24.9\times$ reduction in amortized cost; at batch size $1$, where nothing is amortized, the reduction is $2.4\times$ ($9.984$ against $24.453$ ms). At batch $256$ the baseline exhausts a $40$ GiB device while MaLaCo uses $599$ MiB. All of this holds even though MaLaCo has more parameters.

**(2) Controlled recall-at-depth gains.** Across four datasets, MaLaCo improves Recall@$k$ at every reported cutoff $k \geq 10$; a four-seed $2 \times 2$ architecture–encoder study confirms that its advantage increases with retrieval depth under either item encoder.

**(3) Collision-aware, extensible identifiers.** The proposed SID construction reduces the excess-collision rate to at most $0.0122\%$ across four catalogs. When reserving the newest quarter of items from vocabulary fitting, the method assigns SIDs to these unseen items with zero identifier churn and no newly introduced collisions.. After recommender retraining, the extended vocabulary obtains $0.04526$ Recall@10 versus $0.04598$ after full-catalog vocabulary refitting; existing-item Recall@10 is $0.0575$ versus $0.0580$.

## 2 Related Work

**Sequential and generative recommendation.** SASRec (Kang & McAuley, 2018) models interaction sequences with causal self-attention, whereas BERT4Rec (Sun et al., 2019) predicts masked items using bidirectional context. Diffusion-based recommenders corrupt and iteratively denoise target representations or recommendation signals (Li et al., 2023; Du et al., 2023; Yang et al., 2023). MaLaCo instead uses masking only as a training corruption. At inference, it completes one blank target slot with a single deterministic backbone pass.

TIGER introduced residual-quantized SIDs for sequential recommendation (Rajput et al., 2023). LC-Rec and LETTER incorporate language semantics, collaborative signals, and code-diversity objectives into SID-based recommendation (Zheng et al., 2024; Wang et al., 2025), while OneRec introduces session-wise generation (Deng et al., 2025). LIGER combines generative and dense retrieval to improve ranking and cold-item recommendation (Lan et al., 2025). RPG replaces ordered autoregressive SIDs with long, unordered codes whose tokens are predicted in parallel (Hou et al., 2025). VaLiDRec represents items with variable-length sets of native LLM tokens and ranks items from parallel token scores, avoiding autoregressive decoding and beam search (Qiao et al., 2026). MaLaCo instead retains ordered, hierarchical SIDs and trie-constrained autoregressive decoding, but isolates this search from the history backbone. LLaDA-Rec provides the closest architectural baseline: both methods use bidirectional masked prediction and retrieve catalog items by generating their SIDs with beam search (Shi et al., 2025). The difference is where SID generation occurs. LLaDA-Rec denoises masked SID tokens inside the full history backbone, so every surviving hypothesis re-enters that backbone at each denoising step. MaLaCo instead computes the continuous target state once and confines beam-dependent SID generation to a separate, smaller head.

**Semantic-ID construction and catalog change.** Residual-quantized autoencoders produce multi-level hierarchical discrete representations (Lee et al., 2022). Balanced assignment and diversity losses can improve the use of individual codes, but balanced code usage does not guarantee that every item receives a unique full SID. Recent work studies qualification-aware and adaptive handling of SID collisions (Hu et al., 2026; Pan et al., 2026). Our method instead learns a per-item candidate policy with a direct collision penalty and then resolves the remaining conflicts through catalog-level assignment, where the choice of an SID for one item depends on the assignments of all other items. A recent temporal study shows that SID models can occasionally retrieve new items whose tokens and prefixes were observed during training, but struggle with unsupported SID paths (Peng et al., 2026). Our catalog-growth experiment tests a related but distinct question: whether new items can be assigned SIDs without changing existing assignments and whether the fixed SID vocabulary remains usable after retraining the recommender. Because the frozen recommender assigns these unseen items almost no probability, we do not claim zero-shot retrieval.

**Masked objectives.** For absorbing-mask discrete diffusion, training can be expressed as weighted masked-token prediction (Austin et al., 2021; Sahoo et al., 2024), and LLaDA applies this principle at language-model scale (Nie et al., 2025). MaLaCo uses a related masking corruption over continuous item slots, but it is not a diffusion sampler: inference uses neither a reverse diffusion schedule nor iterative denoising.

# 3 METHOD

## 3.1 MASKED LATENT COMPLETION AND CONSTRAINED DECODING

Let $x = (i_1, \ldots, i_h)$ denote a user's interaction history and $i^+ \in \mathcal{V}$ the next item. Each item $i$ has a continuous representation $\boldsymbol{z}_i \in \mathbb{R}^d$ and an $L$-token semantic ID $\boldsymbol{s}_i = (s_i^1, \ldots, s_i^L) \in [K]^L$. MaLaCo separates history encoding from SID generation:

$$H = f_\theta(x), \qquad p(\boldsymbol{s} \mid x) = p_\phi(\boldsymbol{s} \mid H) = \prod_{l=1}^{L} p_\phi(s^l \mid H, s^{<l}). \tag{1}$$

The history encoder $f_\theta$ produces $H$ once. A smaller autoregressive head $p_\phi$ then generates the SID, so different beam hypotheses do not require additional history-encoder passes.

**Single-pass latent completion.** The encoder operates on a fixed-length sequence of item positions. Each position has one of three roles: $\rho_n \in \{\text{OBS}, \text{GEN}, \text{PAD}\}$. An OBS position contains the continuous representation of an observed item. A GEN position represents an item that has been hidden and must be predicted, so its input vector is set to zero. A PAD position is also zero but is ignored as padding. The input embedding at position $n$ is $H_n^{(0)} = W_z Z_n + \text{pos}(n) + r(\rho_n)$, where $W_z \in \mathbb{R}^{d_{\text{model}} \times d}$ projects the $d$-dimensional item representation into the Transformer hidden space, $\text{pos}(n)$ is a positional embedding, and $r(\rho_n)$ identifies the row's role. The role embedding is important because both GEN and PAD positions use $Z_n = 0$. The use of GEN differs between training

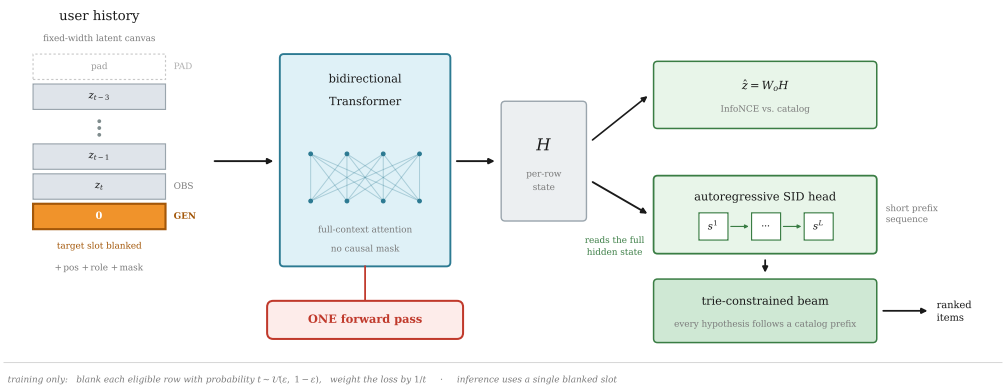



Figure 1: MaLaCo encodes the continuous interaction history and one blank target position in a single bidirectional pass. The resulting target state is used by an autoregressive SID head and an auxiliary contrastive projection. Beam search runs only in the SID head and never re-enters the history encoder.

and inference. During training, we randomly hide one or more items in the sequence. Each hidden item is changed from OBS to GEN, and the model is trained to recover that item from the remaining context. A training sequence may therefore contain several GEN positions, including positions inside the history. During next-item inference, unused leading positions are marked PAD, all positions containing real history items are marked OBS, and exactly one GEN position is placed at the end. The bidirectional Transformer processes the sequence once, and the output at this final position becomes the target state $H$. Because the inference-time GEN position is last, it cannot attend to any future interaction. The SID head uses $H$ to generate the next item's SID.

**Masked training.** During training, we mask a random subset of observed item positions. For each example, we sample a masking rate $t \sim \mathcal{U}(\varepsilon, 1-\varepsilon)$ and independently mask each eligible position with probability $t$. We ensure that at least one position is masked. On Scientific this hides on average $4.4$ of the $8.5$ eligible positions per window (median $50\%$ of the history), and in $43\%$ of examples the final item stays visible so that only interior positions are recovered. Let $\mathcal{M}$ denote the masked positions and $\mathcal{E}$ all eligible positions. The masking pattern is resampled each epoch, so a history is not deterministically paired with the same fixed completion target throughout training. Section G isolates this corruption against an otherwise identical arm that hides only the final item.

**Training losses.** For each masked position $n$, the encoder produces a hidden state $H_n$. We train this state with two objectives. First, a learned projection $W_o \in \mathbb{R}^{d \times d_{\text{model}}}$ maps $H_n$ back into the $d$-dimensional item-latent space: $\hat{\boldsymbol{z}}_n = W_o H_n$. The InfoNCE loss encourages $\hat{\boldsymbol{z}}_n$ to be similar to the true latent $\boldsymbol{z}_n^+$ of the masked item and dissimilar to sampled catalog negatives:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\cos(\hat{\boldsymbol{z}}_n, \boldsymbol{z}_n^+)/\tau)}{\exp(\cos(\hat{\boldsymbol{z}}_n, \boldsymbol{z}_n^+)/\tau) + \sum_{\boldsymbol{z}^- \in \mathcal{N}} \exp(\cos(\hat{\boldsymbol{z}}_n, \boldsymbol{z}^-)/\tau)}. \tag{2}$$

Second, the autoregressive SID head reads the full hidden state $H_n$ and is trained to generate the masked item's SID with loss $\mathcal{L}_{\text{SID}} = \sum_{l=1}^{L} \text{CE}\left(p_\phi(\cdot \mid H_n, s^{<l}), s^l\right)$. The loss for position $n$ is $\ell_n = \mathcal{L}_{\text{InfoNCE}} + \mathcal{L}_{\text{SID}}$. The projection $W_o$ is used only to provide auxiliary contrastive supervision during training. It is not used by the SID head and does not score or rerank items at inference. The full masking loss is $\mathcal{L}_{\text{mask}} = \frac{1}{|\mathcal{E}|} \sum_{n \in \mathcal{M}} \frac{1}{t} \ell_n$. The factor $1/t$ prevents examples with high masking rates from receiving more weight simply because they contain more masked positions. Because we force every example to contain at least one mask, we use this factor as a practical reweighting rather than claim that it is exactly unbiased.

The SID head is a causal Transformer that takes the projected history state $W_h H$, a BOS token, and the previously generated SID tokens as input. It shares input embeddings across levels but uses a separate output classifier at each level. During beam search, a catalog trie masks any token that would create an invalid SID prefix. Complete SIDs are ranked by their summed token log-probabilities,

$\text{score}(i) = \sum_{l=1}^{L} \log p_\phi(s_i^l \mid H, s_i^{<l})$. The contrastive projection is not used for ranking. All Recall@$k$ and NDCG@$k$ metrics are item-level: each generated SID is resolved through a fixed SID-to-item mapping, with collisions broken by the tokenizer's fixed random construction order (Section 5).

**Inference cost.** Let $\mathcal{C}(D, S, d) = \Theta(D(Sd^2 + S^2 d))$ denote the cost of a Transformer with $D$ layers, sequence length $S$, and hidden width $d$. With $T$ denoising steps and beam width $B$, the costs of LLaDA-Rec and MaLaCo are

$$\mathcal{C}_{\text{LLaDA}} = TB \times \mathcal{C}(D_b, S_b, d_b), \qquad \mathcal{C}_{\text{MaLaCo}} = \mathcal{C}(D_m, S_m, d_m) + \sum_{l=1}^{L} B_l \mathcal{C}(D_h, l+1, d_h), \quad (3)$$

where $B_l \leq B$ is the number of active beams at level $l$. In LLaDA-Rec, beam width multiplies the cost of the full history backbone. In MaLaCo, the history is encoded once, and beam width affects only the short SID-head sequences. The actual speedup depends on the hardware and implementation.

### 3.2 Collision-aware SID construction

A frozen item encoder maps each item's title, description, and category metadata to $\boldsymbol{e}_i$. We center these embeddings, keep the leading $d$ principal components, and whiten them to obtain $\boldsymbol{z}_i$. We then construct the SID table in four stages. Full objectives and pseudocode are given in Section A.

**1. Semantic warm start.** Balanced residual $k$-means learns $K$ centroids at each SID level. Starting with $\boldsymbol{r}_i^1 = \boldsymbol{z}_i$, level $l$ assigns code $s_i^l$ using entropy-regularized Sinkhorn transport over the distances to $\boldsymbol{c}_k^l * k = 1^K$. It then updates the residual as $\boldsymbol{r}_i^{l+1} = \boldsymbol{r}_i^l - \boldsymbol{c} * {s_i^l}^l$. This produces a balanced, coarse-to-fine semantic hierarchy, but different items may still receive the same full SID.

**2. Last-level deduplication.** The warm start leaves a fifth to a third of each catalog sharing a full SID, so we resolve the locally fixable conflicts before learning anything. Each group of items that share a SID is re-quantized as its own population: the last code is re-assigned by the same Sinkhorn transport, now applied within the group so that the balance constraint spreads its members across distinct codewords, while levels $1, \ldots, L-1$ are kept at their $\arg\min$ codes. Repeating this until no group remains, or a round budget is spent, removes most of the collision (Table 4). Because only the last code can change, the coarse partition is untouched by construction, and level-0 statistics are identical before and after. What survives is the conflict that no local re-assignment can fix, which is what the next two stages address.

**3. Collision-directed candidate policy.** An autoregressive encoder $\pi_\psi(\boldsymbol{s} \mid \boldsymbol{z})$ proposes SIDs, and a decoder $g_\omega(\boldsymbol{s})$ reconstructs the item latent. The models first learn to reproduce the warm-start codes and reconstruct the latent. They are then refined with REINFORCE using the reward $R(\boldsymbol{s}, \boldsymbol{z}) = \cos(\boldsymbol{z}, g_\omega(\boldsymbol{s})) - \lambda_{\text{col}} \mathbb{1}[\boldsymbol{s} \text{ is already claimed}]$, together with level-wise KL regularization toward the warm-up policy. A catalog-wide claim table identifies SIDs already assigned to other items. This stage improves the candidates proposed for each item, but it cannot guarantee a one-to-one assignment across the catalog.

**4. Catalog assignment.** Beam search produces $W$ SID candidates and their log-probabilities for each item. We process items in decreasing order of the margin between their two highest-scoring candidates and assign each item its highest-ranked unclaimed SID. If all $W$ candidates are already occupied, we keep the top candidate, which may leave a residual collision. The method is therefore a deterministic catalog-level heuristic rather than a guarantee of injectivity. When the catalog grows, we reserve all existing SIDs and apply the assignment only to new items. This prevents existing identifiers from changing; the absence of new collisions is an empirical result at the tested candidate widths, not a general guarantee.

## 4 Experiments

We organize the evaluation around four questions: (Q1) How does separating history encoding from SID decoding affect recommendation quality at different ranking depths? (Q2) Are the observed

Table 1: Head-to-head recommendation quality (four-seed mean ± std). Full baselines in Section B.

| Dataset | Method | R@1 | R@10 | R@50 | N@10 |
|---|---|---|---|---|---|
| Scientific | RPG | 0.0083±.0006 | 0.0407±.0006 | 0.0965±.0008 | 0.0220±.0005 |
| | LLaDA-Rec | **0.0097**±.0004 | 0.0466±.0010 | 0.1081±.0012 | 0.0253±.0005 |
| | **MaLaCo** | 0.0089±.0004 | **0.0499**±.0011 | **0.1173**±.0018 | **0.0261**±.0008 |
| Instrument | RPG | 0.0118±.0003 | 0.0567±.0006 | 0.1310±.0007 | 0.0308±.0004 |
| | LLaDA-Rec | **0.0122**±.0002 | 0.0615±.0009 | 0.1420±.0010 | **0.0330**±.0004 |
| | **MaLaCo** | 0.0111±.0002 | **0.0621**±.0008 | **0.1467**±.0012 | 0.0325±.0003 |
| Game | RPG | **0.0209**±.0003 | 0.0925±.0002 | 0.2097±.0005 | **0.0513**±.0003 |
| | LLaDA-Rec | **0.0198**±.0003 | 0.0939±.0001 | 0.2122±.0010 | **0.0512**±.0002 |
| | **MaLaCo** | 0.0163±.0004 | **0.0951**±.0004 | **0.2244**±.0008 | 0.0494±.0001 |
| Arts | RPG | **0.0097**±.0001 | 0.0416±.0002 | 0.0910±.0003 | 0.0233±.0001 |
| | LLaDA-Rec | 0.0081±.0001 | 0.0433±.0001 | 0.0986±.0012 | 0.0229±.0004 |
| | **MaLaCo** | 0.0085±.0001 | **0.0460**±.0002 | **0.1089**±.0010 | **0.0243**±.0001 |

differences caused solely by the model architecture, the item encoder, or an interaction? (Q3) Does moving beam search to the small SID head reduce measured inference latency? (Q4) Does collision-aware SID construction preserve semantics, approach injectivity, and support stable catalog growth?

**Protocol.** We use four categories from the public 2023 product-review benchmark introduced by Hou et al. (2026) (hereafter, *Review-2023*): Industrial & Scientific (50,985 users and 25,848 items), Musical Instruments (57,439; 24,587), Video Games (94,762; 25,612), and Arts, Crafts & Sewing (197,286; 89,958). We follow LLaDA-Rec's procedures for preprocessing and data splits. All four categories are used for the head-to-head MaLaCo–LLaDA-Rec recommendation comparison. The Scientific, Musical Instruments, and Video Games datasets additionally receive the full baseline comparison, while Arts, Crafts & Sewing is also used for the catalog-growth study. We reserve each user's final interaction as the test targets, the penultimate interaction as the validation set, and earlier interactions form the training history. Evaluation is full–catalog, using Recall@$k$ and NDCG@$k$. Unless explicitly noted, MaLaCo and reproduced baselines report means and standard deviations over seeds 2024–2027, select by validation performance, and use a trie-constrained beam of width $50$. Tokenizers are fixed across model seeds.

Item text is encoded with BGE-Multilingual-Gemma2 (Chen et al., 2025), giving 3,584-dimensional vectors that we reduce and PCA-whiten to $d = 128$. Each item then receives a semantic ID of $L = 3$ codes drawn from codebooks of size $K = 256$. MaLaCo reads a fixed window of $N = 20$ history rows. Its backbone is a Transformer of width $256$ with four heads and $4\times$ feed-forward expansion; validation selects two layers for Scientific, Instrument and Arts, and four for Game. The SID head is four layers at the same width. Training uses a global batch of $1{,}024$ and an InfoNCE loss with $250$ popularity-sampled negatives at temperature $\tau = 0.07$. Decoding uses a trie-constrained beam of width $50$. Section A lists the remaining optimization and tokenizer settings.

LLaDA-Rec is reproduced from the authors' public implementation with the released category-specific hyperparameters, except that the available reproduction uses one GPU and global batch $1024$ rather than the released two-GPU launch. The seed-2024 Scientific run reproduces the published Recall@10 exactly; the four-seed means are reported below. SASRec, RPG, BERT4Rec are retrained under the same split and full-catalog evaluation.

LC-Rec values are quoted from prior work and therefore carry no local uncertainty estimate or matched latency measurement. LLaDA-Rec is the strongest baseline at Recall@10 on all three main datasets; therefore, Table 1 presents the controlled MaLaCo–LLaDA-Rec comparison, while Appendix B reports the full baseline results. LIGER is not tabulated because its reported experiments use the earlier Beauty, Sports and Toys categories, so no protocol-matched number exists. VaLiDRec, QuaSID and AdaSID report neither results on these categories nor runnable implementations we could match to this protocol Qiao et al. (2026); Hu et al. (2026); Pan et al. (2026). We therefore cannot say how MaLaCo compares to these methods.

**Q1: Recommendation Quality at Depth** Table 1 gives the closest head-to-head, showing MaLaCo against our reproduction of LLaDA-Rec, run with the same seeds. The performance depends on the

Table 2: Encoder × architecture factorial across two datasets. MaLaCo / Gemma2 is our main model.

| Dataset | Architecture | Encoder | R@1 | R@5 | R@10 | R@20 | R@50 | N@5 | N@10 | N@20 | N@50 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Scientific | LLaDA-Rec | sentence-T5 | **.00969**±.0004 | .03020±.0009 | .04663±.0010 | .06961±.0008 | .10809±.0012 | .02000±.0005 | .02528±.0005 | .03107±.0005 | .03870±.0004 |
| | | **Gemma2** | .00929±.0004 | .03008±.0006 | .04666±.0010 | .06896±.0004 | .10719±.0009 | .01972±.0005 | .02505±.0006 | .03066±.0004 | .03824±.0006 |
| | MaLaCo | sentence-T5 | .00846±.0004 | .02994±.0006 | .04727±.0011 | .07154±.0012 | .11154±.0015 | .01924±.0005 | .02480±.0006 | .03090±.0006 | .03885±.0008 |
| | | **Gemma2** | .00885±.0004 | **.03139**±.0013 | **.04993**±.0011 | **.07504**±.0019 | **.11735**±.0018 | **.02011**±.0009 | **.02607**±.0008 | **.03238**±.0011 | **.04077**±.0010 |
| Musical Instr. | LLaDA-Rec | sentence-T5 | .01218±.0002 | .03968±.0006 | .06147±.0009 | .09087±.0009 | .14198±.0010 | .02603±.0003 | .03304±.0004 | .04043±.0004 | .05056±.0004 |
| | | Gemma2 | **.01278**±.0003 | **.03997**±.0006 | .06133±.0007 | .09135±.0007 | .14191±.0004 | **.02648**±.0003 | **.03334**±.0002 | **.04089**±.0003 | .05091±.0003 |
| | MaLaCo | sentence-T5 | .01077±.0004 | .03828±.0006 | .06050±.0009 | .09070±.0012 | .14336±.0009 | .02456±.0003 | .03171±.0004 | .03930±.0005 | .04973±.0005 |
| | | **Gemma2** | .01107±.0002 | .03913±.0005 | **.06208**±.0008 | **.09316**±.0008 | **.14674**±.0012 | .02512±.0002 | .03249±.0003 | .04031±.0003 | **.05094**±.0003 |

retrieval depth. On Scientific, Instrument and Game, MaLaCo is behind at rank 1: Recall@1 drops by 0.0008, 0.0011, and 0.0035, respectively; on Game, the deficit extends to Recall@5 ($-0.0013$). By Recall@10, the gap has moved in MaLaCo's favour ($+0.0033$, $+0.0006$, and $+0.0012$), and at Recall@50 MaLaCo leads by $+0.0092$, $+0.0047$, and $+0.0122$. Arts, the largest catalog we use (89,958 items), does not show the rank-1 deficit at all. MaLaCo leads at every cutoff, by $+0.0004$ at Recall@1 and $+0.0103$ at Recall@50. Thus, across datasets, its relative advantage increases with $k$.

NDCG shows the same trade-off from a different angle. Because NDCG weights the top of the list, the rank-1 deficit carries further: MaLaCo leads on Scientific at every cutoff, trails on Instrument and on Game through NDCG@20, and moves ahead only at NDCG@50. While we cannot describe the model as uniformly better, it provides strong, low-cost, candidate generation when recall at depth is what matters.

Appendix B adds reproduced SASRec, RPG, BERT4Rec results and quoted LC-Rec results on the four main datasets. MaLaCo outperforms all of these methods at Recall@10 on the datasets. These models are not matched for parameter count or optimization budget, so they place the method in context but do not isolate the proposed factorization.

**Q2: Encoder Quality and Architecture Interaction.** Because the main comparison changes both architecture and item encoder, we cross each architecture with sentence-T5-base and BGE-Multilingual-Gemma2 on Scientific and Instruments, holding dimensions, seeds, and architecture-specific hyperparameters fixed.

Table 2 supports two conclusions. supports two conclusions. First, in all four architecture–encoder cells, MaLaCo begins behind at rank 1 and its relative performance improves monotonically with $k$, ending ahead at Recall@50. Gemma2 shifts the crossover earlier, particularly on Musical Instruments, where it moves from Recall@50 under sentence-T5 to Recall@10 under Gemma2. Second, Gemma2 improves MaLaCo at every reported cutoff, increasing Recall@10/50 by $+0.00265/+0.00581$ on Scientific and $+0.00158/+0.00338$ on Musical Instruments, while changing LLaDA-Rec only marginally. This asymmetry is consistent across both datasets, although two datasets are insufficient to establish that discrete-only models generally cannot exploit stronger encoders.

**Q3: Effect of Factorization on Generation Cost** Table 3 shows that factorization cuts generation cost by $\mathbf{2.4}\times$ online and $\mathbf{24.9}\times$ at batch 20 and where that cost goes. We count work in *position-evaluations*: one token position processed by one Transformer forward pass, without key–value caching. LLaDA-Rec rocesses each of 50 beam hypotheses through its backbone each of four denoising steps, totaling 16,800 position-evaluations; MaLaCo encodes the 20-row history once and confines beam search to the SID head, totalling 352, roughly $47\times$ less work per query.

Measured latency falls from 13.097 to 0.526 ms per query amortized over a batch of 20, with the advantage increasing with batch size because MaLaCo amortizes history encoding while LLaDA-Rec repeatedly re-encodes each hypothesis. Peak memory shows the same pattern: at batch 128 LLaDA-Rec holds 18.7 GiB versus MaLaCo's 326 MiB, and at batch 256 it exhausts the 40 GiB device entirely while MaLaCo uses 599 MiB. This is not explained by model size or baseline quality: MaLaCo is larger (6.19M versus 4.75M parameters), and the baseline checkpoint reproduces published Scientific Recall@10.

To control for codebase differences, we rerun our backbone using the baseline's decoding pattern—one forward pass per hypothesis per denoising step, while holding weights, device, and precision fixed. MaLaCo is $3.26\times$ faster at batch 20 and up to $7.43\times$ faster at batch 256, but slightly slower at batch 1 ($0.91\times$), showing that the matched benefit is throughput rather than single request latency. The larger $24.9\times$ end-to-end difference at batch 20 also reflects LLaDA-Rec's deeper backbone,

Table 3: Generation cost per query on Scientific, beam 50, one A100-SXM4-40GB, both systems timed on the same device with synchronized calls. The batch-20 column is amortized throughput; batch 1 is single-request latency.

| *Cost structure* | LLaDA-Rec | MaLaCo |
|---|---|---|
| Backbone layers / width | 4/256 | 2/256 |
| History positions | 84 | 20 |
| Backbone passes/query | 200 | **1** |
| Backbone position-evals | 16,800 | **20** |
| Head position-evals | — | 352 |
| Parameters | 4.75M | 6.19M |

| *Measured cost* | LLaDA-Rec | MaLaCo |
|---|---|---|
| ms / query, batch 20 | 13.097 | **0.526** |
| relative | 1.0× | **1/24.9** |
| ms / query, batch 1 | 24.453 | **9.984** |
| relative | 1.0× | **1/2.4** |
| Peak MiB, batch 20 | 3,013 | **92** |
| Peak MiB, batch 128 | 19,131 | **326** |
| Batch 256 | OOM | 599 MiB |
| Matched-decoder control | — | 1/3.26 |

Table 4: What each construction stage costs and buys, Scientific (more in Section D.1).

| Output vocabulary | Excess coll. | $c_0$ dept. pur. | $(c_0\, c_1)$ leaf pur. | $U(\text{dept} \mid c_0)$ |
|---|---|---|---|---|
| RQ-VAE + Sinkhorn, last level | 0.3095% | 0.805 | **0.899** | 0.784 |
| Balanced residual $k$-means | 25.5455% | **0.842** | 0.835 | **0.819** |
| + Sinkhorn dedup, last level | 0.6809% | **0.842** | 0.835 | **0.819** |
| + SID-AE + beam (**ours**) | **0.0039%** | 0.841 | 0.835 | 0.818 |

longer sequence, and implementation. Section F reports the complete sweep, measurement protocol, peak-memory results, and throughput figures.

**Q4: Collision-Aware Naming with Stable Catalog Insertion.** Each stage of our SID-assignment pipeline addresses a different collision scale. Across the four catalogs, balanced $k$-means alone leaves 18–31% of items sharing a tuple, deduplication brings that to 0.56–1.56%, and the autoencoder with catalog assignment brings it to at most 0.0122%. Instrument becomes exactly injective and the other three are near-injective (11 collisions in Arts, 1 in Scientific, and 2 in Game) . Table 12 gives the full cascade and shows that the refinement stage edits the level-0 code for at most 0.3% of items on any dataset.

Table 4 shows the semantic cost of injectivity. Last-level Sinkhorn on the published RQ-VAE recipe leaves 0.3095% excess collision, 79× ours, and its first code carries less of the department label ($U = 0.784$ versus 0.818), although its two-code leaf prefix (0.899 versus 0.835) is purer. Within our own cascade, level-0 department purity remains approximately 0.840 while collision falls by four orders of magnitude against a 0.140 majority-class baseline. Balancing every level removes more collision than last-level balancing alone (0.2515% versus 0.6809%), however it collapses the hierarchy, with $c_0$ purity falling to 0.621 and $U$ to 0.482 (Table 11). Our full cascade reaches 0.0039% without this cost. Purity and $U$ are descriptive probes against metadata labels the model never optimizes and never retrieves against; Table 14 reports for all four datasets.

Collision reduction alone does not establish recommendation utility. Table 5 holds the Scientific backbone, capacity, and seeds fixed, changing only the identifier table, now at the champion's own batch size (1024). Random injective codes are perfectly injective, achieving Recall@10 of .0426, close to a direct catalog softmax (.0412), despite being more unique than our vocabulary's 0.004% excess collision. Thus uniqueness alone contributes little; semantic organization matters. Last-level Sinkhorn balancing on an RQ-VAE warm start reaches .0481, while our near-injective vocabulary performs best, reaching .0499 ± .0011 Recall@10.

For catalog growth, we fit the complete naming pipeline on the oldest 67,468 (75%) Arts items and reserve their SIDs while assigning the newest 22,490. Candidate width $W \geq 64$ covers every new item, introduces no new collision at any width, and leaves every existing identifier byte-identical. Of new items, 78.1% share their first code with their nearest existing latent neighbor (chance 0.83%). Despite stable semantic placement, the frozen model obtains zero new-item Recall@10, matching random unoccupied codes. Adapting only the SID head, at the largest prespecified sweep, recovers new-item Recall@10 to 95% of the refit-and-retrain upper bound, confirming that the codes are reachable and the limitation is in the training signal. However existing-item recall decays monotonically with adaptation, and pooled recall never exceeds the frozen baseline. With the recommender frozen, the extended vocabulary reaches $0.04305 \pm 0.00027$ overall Recall@10, versus

Table 5: What does the output vocabulary contribute? Scientific; only the semantic-ID table changes.

| Output vocabulary | Excess collision | R@10 | N@10 |
|---|---|---|---|
| Direct catalog softmax | — | .0412±.0006 | .0214±.0004 |
| Random injective SIDs | **0.000%** | .0426±.0006 | .0222±.0005 |
| RQ-VAE + Sinkhorn, *all* levels | 0.058% | .0455±.0004 | .0238±.0001 |
| RQ-VAE + Sinkhorn, last level | 0.310% | .0481±.0008 | .0251±.0005 |
| Ours (bal. $k$-means + dedup + SID-AE + deconfl.) | 0.004% | **.0499±.0011** | **.0261±.0008** |

Table 6: Catalog growth on Arts. The vocabulary is fitted on the oldest 75% of items.

| Model | Vocabulary | New R@10 | Existing R@10 | Overall R@10 |
|---|---|---|---|---|
| Frozen | extended | .0000 | .0579±.0004 | .04305±.00027 |
| Head adapted | extended | .0105±.0004 | .0491±.0004 | .03921±.00035 |
| Retrained | extended | .0098±.0003 | .0575±.0007 | .04526±.00057 |
| Retrained | refitted | .0110±.0003 | .0580±.0002 | .04598±.00023 |

$0.04598 \pm 0.00023$ for the champion refitted and retrained on the full data, with the gap reflecting the missing training signal. The existing-item slice differs by only $-0.26\%$ ($0.05788$ versus $0.05803$), the previous frozen model deficit falling almost entirely ($96\%$) on items excluded from its initial training corpus. Retraining the recommender on the full data while keeping the frozen vocabulary closes most of the overall gap, reaching $0.04526 \pm 0.00057$ overall Recall@10, a 1.6% relative shortfall, with the residual concentrated among the new items ($-11.0\%$) rather than the existing ones ($-0.9\%$). Thus, on this partition, the frozen vocabulary is backward-compatible for the items it was fitted on. Section E specifies the catalog split, states which rows each arm trains on, and reports the leakage checks separating the frozen, head-adapted and retrained regimes.

## 5 Limitations and Conclusion

**Experimental scope.** Evidence covers four categories from one benchmark; the encoder factorial uses two and catalog growth one. All reported uncertainties are standard deviations over four seeds and should be interpreted descriptively. MaLaCo trails at Recall@1 on the three main datasets, and its NDCG disadvantage persists through NDCG@20 on Instruments and Games, although it leads at every cutoff on Arts. Latency was measured on one A100 at a fixed beam width and remains implementation-dependent; the matched comparison shows a throughput advantage at larger batches. No matched comparison with recent parallel or token-set SID models is available, and labeled appendix screens are single-seed.

**Vocabulary scope.** Catalog assignment does not guarantee injectivity when every candidate for an item is occupied; one excess Scientific item and two excess Game items remain. The evaluation-time random mapping is not a principled production tie-breaker, so fully injective fallback is required before claiming unique item naming. The growth experiment establishes no-churn assignment on one item-arrival partition, not universal support for catalog evolution, and new items are also systematically rarer. A popularity-matched split is needed to isolate temporal novelty from long-tail difficulty. That experiment also orders items by first interaction while retaining the benchmark's per-user leave-one-out assignment, so it tests vocabulary persistence under catalog aging rather than a deployment simulation with a single global time cutoff. A fully chronological evaluation, in which vocabulary fitting, recommender training, adaptation and testing occupy disjoint intervals, remains future work.

**Conclusion.** MaLaCo encodes history once and confines SID search to a small head, reducing measured latency while preserving recall at depth. The encoder study shows that architecture mediates how item representations affect retrieval. Collision-aware assignment enables stable extension, but naming alone cannot make a frozen decoder retrieve new items. MaLaCo is therefore a practical candidate generator; top-rank quality and large-catalog validation remain open.

### AI USE STATEMENT

Generative AI tools were used for language editing and code-development assistance. The authors reviewed all generated text and code and independently verified the reported experimental outputs. The authors take responsibility for the complete manuscript.

### ETHICS STATEMENT

This study uses public product-interaction data and does not involve human-subject experimentation. Recommendation systems may amplify popularity bias and feedback loops; our offline experiments do not establish fairness, privacy, robustness, or deployment readiness. Deployment would require dedicated evaluation and safeguards.

### REPRODUCIBILITY STATEMENT.

Institutional policy prevents us from releasing source code, checkpoints or intermediate artifacts, which is a genuine limitation of this submission. To compensate, Section A specifies the method, the data and splits, the full semantic-ID construction recipe, every architecture and optimization hyperparameter, the checkpoint-selection rule, the decoding settings and the seeds, each read directly from the configuration file of the corresponding run. The datasets and their preprocessing are public, and the primary baseline is the authors' released implementation run with its published settings, so the item representations, the evaluation protocol and the baseline can be reproduced exactly; the model and tokenizer must be reimplemented from the appendix.

## REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2025. URL https://arxiv.org/abs/2402.03216.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment, 2025. URL https://arxiv.org/abs/2502.18965.

Hanwen Du, Huanhuan Yuan, Zhen Huang, Pengpeng Zhao, and Xiaofang Zhou. Sequential recommendation with diffusion models, 2023. URL https://arxiv.org/abs/2304.04541.

Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 956–966, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3736979. URL https://doi.org/10.1145/3711896.3736979.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders, 2026. URL https://arxiv.org/abs/2403.03952.

Zheng Hu, Yuxin Chen, Yongsen Pan, Xu Yuan, Yuting Yin, Daoyuan Wang, Boyang Xia, Zefei Luo, Hongyang Wang, Songhao Ni, Dongxu Liang, Jun Wang, Shimin Cai, Tao Zhou, Fuji Ren, and Wenwu Ou. Stop treating collisions equally: Qualification-aware semantic id learning for recommendation at industrial scale, 2026. URL https://arxiv.org/abs/2603.00632.

Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. Generative recommendation with semantic ids: A practitioner's handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pp. 6420–6425, New York, NY, USA, 2025. Association

for Computing Machinery. ISBN 9798400720406. doi: 10.1145/3746252.3761612. URL https://doi.org/10.1145/3746252.3761612.

Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation . In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, Los Alamitos, CA, USA, November 2018. IEEE Computer Society. doi: 10.1109/ICDM.2018.00035. URL https://doi.ieeecomputersociety.org/10.1109/ICDM.2018.00035.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures, 2025. URL https://arxiv.org/abs/2503.01496.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11513–11522, 2022. doi: 10.1109/CVPR52688.2022.01123.

Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(3), December 2023. ISSN 1046-8188. doi: 10.1145/3631116. URL https://doi.org/10.1145/3631116.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models, 2021. URL https://arxiv.org/abs/2108.08877.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Yongsen Pan, Yuxin Chen, Zheng Hu, Xu Yuan, Daoyuan Wang, Yuting Yin, Songhao Ni, Hongyang Wang, Jun Wang, Fuji Ren, and Wenwu Ou. Beyond static collision handling: Adaptive semantic id learning for multimodal recommendation at industrial scale, 2026. URL https://arxiv.org/abs/2604.23522.

Jie Peng, Yanping Zheng, Zhewei Zhe, Bin Tong, Guan Wang, and Bo Zheng. Can generative recommendation reach cold items? a temporal perspective on semantic-id generation, 2026. URL https://arxiv.org/abs/2607.21101.

Shutong Qiao, Wei Yuan, Tong Chen, Hao Wang, Quoc Viet Hung Nguyen, and Hongzhi Yin. Validrec: Variable-length llm-aligned semantic ids for generative recommendation, 2026. URL https://arxiv.org/abs/2607.25209.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, Maciej Kula, Ed Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10299–10315. Curran Associates, Inc., 2023. doi: 10.52202/075280-0452. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/20dcab0f14046a5c6b02b61da9f13229-Paper-Conference.pdf.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Teng Shi, Chenglei Shen, Weijie Yu, Shen Nie, Chongxuan Li, Xiao Zhang, Ming He, Yan Han, and Jun Xu. Llada-rec: Discrete diffusion for parallel semantic id generation in generative recommendation, 2025. URL https://arxiv.org/abs/2511.06254.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pp. 1441–1450, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357895. URL https://doi.org/10.1145/3357384.3357895.

Yi Tay, Vinh 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 S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21831–21843. Curran Associates, Inc., 2022. doi: 10.52202/068431-1587. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/892840a6123b5ec99ebaab8be1530fba-Paper-Conference.pdf.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation, 2025. URL https://arxiv.org/abs/2405.07314.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Yancheng Yuan, Xiang Wang, and Xiangnan He. Generate what you prefer: reshaping sequential recommendation via guided diffusion. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448, 2024. doi: 10.1109/ICDE60146.2024.00118.

# A Reproducibility and Complete Method Details

Institutional policy prevents us from releasing source code, checkpoints or intermediate artifacts. We therefore aim to make the method reimplementable from this appendix alone. Two components are already public and need not be reimplemented. The datasets and their preprocessing come from the public 2023 product-review benchmark, and the primary baseline is the authors' released implementation, run with their published category-specific settings. A reader can therefore rebuild the item representations, the evaluation protocol and the baseline exactly, and the remaining work is the model and tokenizer specified here. We consider the absence of a code release a genuine limitation and note it in Section 5.

**Data and splits.** Four categories of the public 2023 product-review benchmark, using the baseline's preprocessing and 5-core filtering. Sequences are built per user in timestamp order: the last interaction is reserved for the test set, the penultimate interaction is for validation, and the remainder forms the training history. Evaluation is full-catalog with one relevant item per query, and all Recall@$k$ and NDCG@$k$ metrics are computed over catalog items rather than SIDs. Beam outputs are converted through the tokenizer's fixed SID-to-item mapping. If multiple item share an SID, the mapping retains the item selected by tokenizer's random construction order; no target aware, popularity-based, or learned tie-breaking rule is applied. The mapping is fixed across all 4 seeds.

Table 7: Dataset scale and sparsity. Test rows equal the number of users, since the split is leave-one-out.

| Dataset | Users | Items | Interactions | Mean len. | Sparsity |
|---|---|---|---|---|---|
| Scientific | 50,985 | 25,848 | 412,947 | 8.1 | 99.9687% |
| Instrument | 57,439 | 24,587 | 511,836 | 8.9 | 99.9638% |
| Game | 94,762 | 25,612 | 814,586 | 8.6 | 99.9664% |
| Arts | 197,286 | 89,958 | 1,786,437 | 9.1 | 99.9899% |

All four categories are extremely sparse. Arts is the sparsest by a factor of three in density ($0.0101\%$ against $0.0313$–$0.0362\%$) as well as the largest, so it is the hardest of the four on both axes. Mean sequence length is nearly constant across categories, so they differ in catalog and user scale rather than in how much history each user provides; the 20-row canvas is therefore equally unconstrained on all four.

**Item representation.** Title, features, category, and description are concatenated and embedded by a frozen BGE-Multilingual-Gemma2, giving 3,584 dimensions. We PCA-whiten to $d = 128$: project onto the leading 128 principal components and rescale each to unit variance. For a centered embedding matrix $E_c = U\Sigma V^\top$,

$$\boldsymbol{z}_i = V_{:d}^\top(\boldsymbol{e}_i - \bar{\boldsymbol{e}}) \oslash \boldsymbol{\sigma}, \qquad \boldsymbol{\sigma} = \mathrm{diag}(\Sigma_{:d})/\sqrt{N}. \tag{4}$$

The basis is stored as `mean`, `components` and `scale`, and is reused unchanged for semantic-ID construction and for the online canvas.

**Semantic-ID construction.** Four stages, applied in order: balanced residual $k$-means; last-level Sinkhorn deduplication; the SID autoencoder; and margin-ordered catalog deconfliction (Algorithm 1). All datasets use $L = 3$ and $K = 256$.

- **Balanced residual quantization.** At level $l$, let $D_{ik}^l = \|\boldsymbol{r}_i^l - \boldsymbol{c}_k^l\|_2^2$. After distance rescaling, Sinkhorn iterations normalize

$$Q \leftarrow \exp(-\widetilde{D}^l/\epsilon_s), \qquad Q\mathbf{1} = \tfrac{1}{N}\mathbf{1}, \quad Q^\top\mathbf{1} = \tfrac{1}{K}\mathbf{1}, \tag{5}$$

  and emit $s_i^l = \arg\max_k Q_{ik}$. The same assignment rule is used both while fitting the codebook and when encoding the catalog. Residuals update as $\boldsymbol{r}_i^{l+1} = \boldsymbol{r}_i^l - \boldsymbol{c}_{s_i^l}^l$.

  Settings: $k$-means++ initialization, 25 Lloyd iterations, $\epsilon_s = 0.05$ for 100 normalization steps, fixed seed 42. Deduplication uses $\epsilon = 0.003$ for 50 rounds on the last level only.

- **SID autoencoder.** The candidate policy and reconstructor factorize as

$$\pi_\psi(\boldsymbol{s} \mid \boldsymbol{z}) = \prod_{l=1}^{L} \pi_\psi(s^l \mid \boldsymbol{z}, s^{<l}), \qquad \hat{\boldsymbol{z}} = g_\omega(\boldsymbol{s}). \tag{6}$$

  Phase A imitates the balanced codes while learning reconstruction:

$$\mathcal{L}_A = \sum_{l=1}^{L} \mathrm{CE}(\pi_\psi(\cdot \mid \boldsymbol{z}, s^{<l}), s_{\mathrm{km}}^l) + 1 - \cos(\boldsymbol{z}, \hat{\boldsymbol{z}}) - \lambda_H H(\bar{p}^{\cdot,1}), \tag{7}$$

  where $\bar{p}^{\cdot,1}$ is the batch-marginal distribution of the first code. In Phase B, sampled SIDs receive the reward $R(\boldsymbol{s}, \boldsymbol{z}) = \cos(\boldsymbol{z}, g_\omega(\boldsymbol{s})) - \lambda_{\mathrm{col}}\mathbb{1}[\boldsymbol{s} \text{ is already claimed}]$. With baseline $b$ and frozen Phase-A policy $\pi_{\mathrm{ref}}$,

$$\mathcal{L}_{\mathrm{enc}} = -\mathbb{E}\left[(R - b)\sum_l \log \pi_\psi(s^l \mid \boldsymbol{z}, s^{<l})\right] + \sum_l \beta_l \mathrm{KL}(\pi_\psi^l \| \pi_{\mathrm{ref}}^l). \tag{8}$$

  The first level carries no explicit KL anchor; reconstruction supplies an implicit coarse constraint. An exploration branch occasionally proposes an otherwise negligible first-level code.

  Encoder and decoder are each four Transformer layers, width 192, four heads, mean pooling. Phase A runs 300 epochs at learning rate $10^{-3}$ with equal cross-entropy and reconstruction weights and $\lambda_H = 0.5$. Phase B runs 400 epochs, draws four samples per item, clips advantages at 5, and uses encoder learning rate $5 \times 10^{-5}$ (cosine-annealed to $10^{-6}$) and decoder learning rate $3 \times 10^{-4}$, with $\lambda_{\mathrm{col}} = 1$, exploration probability 0.2 and KL weights $(0, 0.2, 0.2)$. Deconfliction uses candidate width $W = 8$.

**Extending an existing catalog.** Growing the catalog reuses the same four stages with two changes. First, no stage is refitted: the PCA basis, whitening statistics, balanced centroids and semantic-ID autoencoder are all kept as fitted on the items present at the time, and a new item's raw embedding is projected through the stored basis before it is encoded. Second, deconfliction runs over new items only, with every existing identifier pre-loaded into the claimed set, so an existing item can never be reassigned and identifiers are stable by construction rather than by observation. Extension needs a wider candidate list than initial naming, because the new items compete for what the existing table has left: at $W = 8$ some items find no free candidate, and full coverage requires $W \geq 64$. We report $W = 256$. Section E gives the coverage curve and the resulting recommendation quality.

**Algorithm 1** Margin-ordered catalog deconfliction

**Require:** Candidates $\boldsymbol{s}_i^{(1:W)}$ and log-probabilities for every item $i$

$m_i \leftarrow \log p_i^{(1)} - \log p_i^{(2)}$; $\mathcal{C} \leftarrow \emptyset$
**for** $i$ in decreasing order of $m_i$ **do**
  $w^* \leftarrow \min\{w : \boldsymbol{s}_i^{(w)} \notin \mathcal{C}\}$
  **if** $w^*$ exists **then**
    $\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i^{(w^*)}$
  **else**
    $\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i^{(1)}$ ▷ residual collision possible
  **end if**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{s}_i\}$
**end for**

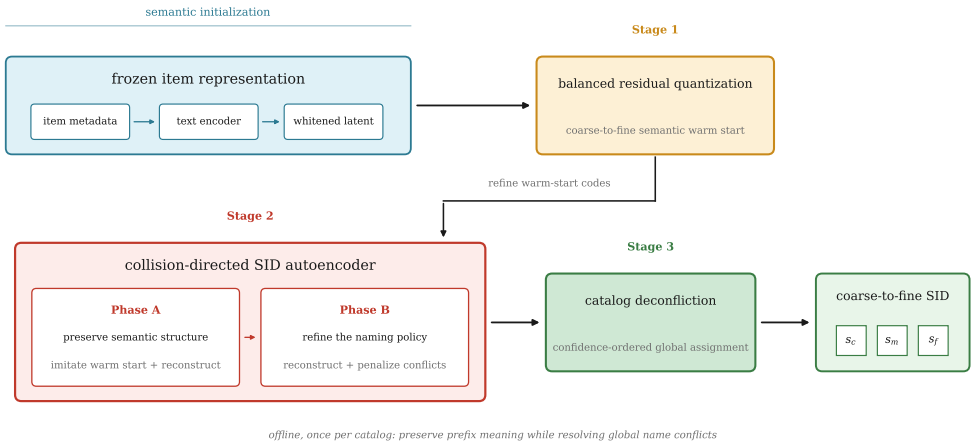



Figure 2: Offline semantic-ID construction. The learned stage proposes a ranked list of candidate IDs for each item on its own. The final stage looks at the whole catalog at once and gives each item the best candidate that no other item has taken. If every candidate for an item is already taken, that item keeps its first choice and a collision remains.

**Recommender and training** The canvas holds $N = 20$ history rows. The backbone has width $256$, four heads, feed-forward expansion $4\times$ and dropout $0.1$; validation selects two layers for Scientific, Instrument and Arts, and four for Game. The SID head has four layers at width $256$. The masking rate is drawn from $(\varepsilon, 1-\varepsilon)$ with $\varepsilon = 0.001$. Completion is a single pass: the model fills every masked position at once. No Gaussian noise is added to the canvas at any point, so there is no iterative denoising schedule to specify. The head reads a sequence of $1 + L$ tokens: a linear projection of $H_n$, then a start token, then the first $L - 1$ codes of the target SID. Attention is causal, so the token that predicts level $l$ sees the state, the start token and the codes of the levels below $l$, and nothing else.

All levels share one code-embedding table. Index $0$ holds the start token and code $c$ is stored at index $c + 1$, so the same code value at two different levels maps to the same vector, and the head adds no positional embedding. Level identity therefore enters in two places. On the output side it is explicit: the $K$ logits for level $l$ come from that level's own output layer, selected by position in the sequence. On the input side it is not, and the head must recover it from how many tokens are visible: attention weights are normalized over the visible window, so the state token, which every position can see, takes a smaller share as the window grows. Depth is thus expressed as a mixing proportion rather than as a vector added to the input, and one embedding row is read differently at each level.

Optimization is AdamW at learning rate $6 \times 10^{-4}$, weight decay $0.05$, global batch $1024$, for $300$ epochs: the rate rises over $10$ warm-up epochs, then decays on a cosine to $10^{-5}$ over the next $210$ epochs, and stays at $10^{-5}$ for the final $80$. The loss has two terms: semantic-ID cross-entropy (weight

Table 8: Review-2023 benchmark statistics (Hou et al., 2026). All four categories use the same $d = 128$ whitened representation and the same $L=3$, $K=256$ SID pipeline of Section 3.2.

| Category (ours) | Baseline name | Users | Items | Backbone depth |
|---|---|---|---|---|
| Industrial & Scientific | Scientific | 50,985 | 25,848 | 2 |
| Musical Instruments | Instrument | 57,439 | 24,587 | 2 |
| Video Games | Game | 94,762 | 25,612 | 4 |
| Arts, Crafts & Sewing | Arts | 197,286 | 89,958 | 2 |

1.0) and InfoNCE (weight 1.0) over 250 popularity-sampled catalog negatives at $\tau = 0.07$. The completed latent itself carries no loss. It is supervised only indirectly, through the head that reads it and the contrastive term, so a direct reconstruction objective on the completed slot is not part of the recommender, unlike the semantic-ID autoencoder of Eq. (7), which does reconstruct. Checkpoints are selected by best validation loss and computed every 20 epochs. Decoding uses a trie-constrained beam of width 50. Seeds are 2024–2027 in every cell, and the tokenizer is held fixed across model seeds. The reproduced baseline uses the same four seeds on Scientific, Instrument, Game, and Arts.

**Baseline reproduction.** We use the short names Scientific, Instrument, Game and Arts for the benchmark categories Industrial & Scientific, Musical Instruments, Video Games, and Arts, Crafts & Sewing. User and item counts in our dataset match those reported by our primary baseline exactly, so Table 9 compares methods on identical splits.

LLaDA-Rec uses the authors' released category-specific settings: learning rate $3 \times 10^{-3}$, weight decay 0.05 on Scientific and Instrument and 0.001 on Game, 150 epochs, per-device batch 1024, and beam width 50. Our runs use one GPU rather than the authors' two-GPU setup. SASRec uses hidden width 64, dropout 0.5, full-catalog cross-entropy, and four seeds. BERT4Rec uses width 64, mask probability 0.4, dropout 0.5, full-softmax Cloze training, up to 600 epochs, and early-stopping patience 8.

We retrain RPG (Hou et al., 2025) rather than quoting it, and our setting differs from the authors' in three respects. Their configuration encodes items with a proprietary embedding API at PCA 512 and caps histories at 50; we substitute BGE-Multilingual-Gemma2 at PCA 128 and cap at 20, so that every semantic-ID row in Table 9 sees the same item text and the same history length as every other baseline. Because the encoder changed, their codebook and optimiser defaults no longer transfer, so we reselected $n_{\text{codebook}}$ per dataset on validation NDCG@10 and retuned the learning rate and temperature (64, $3 \times 10^{-3}$, 0.03, against released values of 32, $3 \times 10^{-4}$, 0.07); on Instrument the gap between 32 and 64 fell inside seed noise and we took 64 for consistency with the other categories. Third, we score the full catalogue at inference instead of using RPG's semantic-ID decoding graph. That is RPG's accuracy ceiling rather than its intended operating point, so the accuracy comparison favours RPG and we make no efficiency claim on its behalf. These deviations do not handicap it: on the three categories for which a previously published RPG number exists in Shi et al. (2025), our runs exceed it at Recall@10 by 3.1%, 4.1% and 8.4%.

## B Complete Recommendation Results

Table 9 reports every available cutoff for the locally reproduced SASRec, BERT4Rec, RPG, LLaDA-Rec, and MaLaCo models. LC-Rec values are quoted from prior work. LC-Rec values are quoted from Shi et al. (2025), and therefore have no local uncertainty estimates. For all locally reproduced methods, we report means and standard deviations over four seeds. Because the evaluation contains many metrics and cutoffs but only four runs per method, we treat these results descriptively and do not report inferential significance tests.

## C Efficiency Details

The LLaDA-Rec timing covers the full test split after discarding the first batch; at batch 20 the median batch time is 261.931 ms and the median per-query cost is 13.097 ms, indicating little tail skew. MaLaCo's batch-20 time is $10.53 \pm 0.02$ ms, or 0.526 ms per query. Position-evaluation counts

Table 9: Recommendation performance. Results are reported as mean ± standard deviation. LC-Rec results are taken from Shi et al. (2025)) and therefore have no local uncertainty estimates. The best reproduced result is bold.

**(a) Recall@$k$**

| Dataset | Method | @1 | @5 | @10 | @20 | @50 |
|---|---|---|---|---|---|---|
| Scientific | LC-Rec | 0.0091 | 0.0280 | 0.0434 | — | — |
| | SASRec (repro) | 0.0040 ±.0010 | 0.0254 ±.0004 | 0.0407 ±.0005 | 0.0610 ±.0003 | 0.0998 ±.0013 |
| | BERT4Rec (repro) | 0.0049 ±.0002 | 0.0190 ±.0003 | 0.0313 ±.0006 | 0.0491 ±.0012 | 0.0847 ±.0015 |
| | RPG (repro) | 0.0083 ±.0006 | 0.0262 ±.0005 | 0.0407 ±.0006 | 0.0603 ±.0008 | 0.0965 ±.0008 |
| | LLaDA-Rec (repro) | **0.0097** ±.0004 | 0.0302 ±.0010 | 0.0466 ±.0010 | 0.0696 ±.0009 | 0.1081 ±.0012 |
| | **MaLaCo (ours)** | 0.0089 ±.0004 | **0.0314** ±.0013 | **0.0499** ±.0011 | **0.0750** ±.0019 | **0.1173** ±.0018 |
| Instrument | LC-Rec | 0.0119 | 0.0379 | 0.0587 | — | — |
| | SASRec (repro) | 0.0071 ±.0005 | 0.0334 ±.0004 | 0.0539 ±.0015 | 0.0811 ±.0008 | 0.1322 ±.0014 |
| | BERT4Rec (repro) | 0.0076 ±.0003 | 0.0291 ±.0008 | 0.0473 ±.0001 | 0.0722 ±.0003 | 0.1200 ±.0010 |
| | RPG (repro) | 0.0118 ±.0003 | 0.0371 ±.0004 | 0.0567 ±.0006 | 0.0826 ±.0007 | 0.1310 ±.0007 |
| | LLaDA-Rec (repro) | **0.0122** ±.0002 | **0.0397** ±.0007 | 0.0615 ±.0009 | 0.0909 ±.0010 | 0.1420 ±.0010 |
| | **MaLaCo (ours)** | 0.0111 ±.0002 | 0.0391 ±.0005 | **0.0621** ±.0008 | **0.0932** ±.0008 | **0.1467** ±.0012 |
| Game | LC-Rec | 0.0165 | 0.0567 | 0.0891 | — | — |
| | SASRec (repro) | 0.0099 ±.0013 | 0.0545 ±.0003 | 0.0883 ±.0001 | 0.1334 ±.0005 | 0.2124 ±.0010 |
| | BERT4Rec (repro) | 0.0085 ±.0006 | 0.0363 ±.0008 | 0.0617 ±.0010 | 0.0986 ±.0012 | 0.1668 ±.0009 |
| | RPG (repro) | **0.0209** ±.0003 | 0.0609 ±.0005 | 0.0925 ±.0002 | 0.1358 ±.0001 | 0.2097 ±.0005 |
| | LLaDA-Rec (repro) | 0.0198 ±.0003 | **0.0613** ±.0003 | 0.0939 ±.0001 | 0.1379 ±.0012 | 0.2122 ±.0010 |
| | **MaLaCo (ours)** | 0.0163 ±.0004 | 0.0600 ±.0005 | **0.0951** ±.0004 | **0.1433** ±.0007 | **0.2244** ±.0008 |
| Arts | SASRec (repro) | 0.0051 ±.0002 | 0.0234 ±.0004 | 0.0374 ±.0003 | 0.0566 ±.0004 | 0.0927 ±.0004 |
| | BERT4Rec (repro) | 0.0043 ±.0003 | 0.0177 ±.0002 | 0.0291 ±.0003 | 0.0460 ±.0004 | 0.0783 ±.0004 |
| | RPG (repro) | **0.0097** ±.0001 | 0.0277 ±.0001 | 0.0416 ±.0002 | 0.0596 ±.0004 | 0.0910 ±.0003 |
| | LLaDA-Rec (repro) | 0.0081 ±.0001 | 0.0274 ±.0002 | 0.0433 ±.0001 | 0.0644 ±.0005 | 0.0986 ±.0012 |
| | **MaLaCo (ours)** | 0.0085 ±.0001 | **0.0292** ±.0002 | **0.0460** ±.0002 | **0.0692** ±.0005 | **0.1089** ±.0010 |

**(b) NDCG@$k$**

| Dataset | Method | @5 | @10 | @20 | @50 |
|---|---|---|---|---|---|
| Scientific | LC-Rec | 0.0186 | 0.0235 | — | — |
| | SASRec (repro) | 0.0148 ±.0005 | 0.0198 ±.0005 | 0.0249 ±.0005 | 0.0325 ±.0007 |
| | BERT4Rec (repro) | 0.0119 ±.0002 | 0.0159 ±.0003 | 0.0203 ±.0004 | 0.0274 ±.0005 |
| | RPG (repro) | 0.0173 ±.0004 | 0.0220 ±.0005 | 0.0269 ±.0004 | 0.0340 ±.0004 |
| | LLaDA-Rec (repro) | 0.0200 ±.0006 | 0.0253 ±.0005 | 0.0311 ±.0005 | 0.0387 ±.0005 |
| | **MaLaCo (ours)** | **0.0201** ±.0009 | **0.0261** ±.0008 | **0.0324** ±.0011 | **0.0408** ±.0010 |
| Instrument | LC-Rec | 0.0251 | 0.0318 | — | — |
| | SASRec (repro) | 0.0203 ±.0004 | 0.0269 ±.0007 | 0.0337 ±.0006 | 0.0438 ±.0007 |
| | BERT4Rec (repro) | 0.0183 ±.0004 | 0.0241 ±.0002 | 0.0304 ±.0003 | 0.0398 ±.0004 |
| | RPG (repro) | 0.0245 ±.0003 | 0.0308 ±.0004 | 0.0373 ±.0003 | 0.0469 ±.0003 |
| | LLaDA-Rec (repro) | **0.0260** ±.0004 | **0.0330** ±.0004 | **0.0404** ±.0005 | 0.0506 ±.0004 |
| | **MaLaCo (ours)** | 0.0251 ±.0002 | 0.0325 ±.0003 | 0.0403 ±.0003 | **0.0509** ±.0003 |
| Game | LC-Rec | 0.0366 | 0.0471 | — | — |
| | SASRec (repro) | 0.0323 ±.0007 | 0.0432 ±.0007 | 0.0546 ±.0006 | 0.0702 ±.0007 |
| | BERT4Rec (repro) | 0.0223 ±.0007 | 0.0304 ±.0008 | 0.0397 ±.0008 | 0.0532 ±.0007 |
| | RPG (repro) | **0.0412** ±.0004 | **0.0513** ±.0003 | **0.0622** ±.0003 | 0.0769 ±.0003 |
| | LLaDA-Rec (repro) | 0.0407 ±.0003 | 0.0512 ±.0002 | **0.0622** ±.0002 | 0.0770 ±.0003 |
| | **MaLaCo (ours)** | 0.0381 ±.0001 | 0.0494 ±.0001 | 0.0615 ±.0002 | **0.0776** ±.0002 |
| Arts | SASRec (repro) | 0.0143 ±.0002 | 0.0189 ±.0002 | 0.0237 ±.0003 | 0.0308 ±.0003 |
| | BERT4Rec (repro) | 0.0110 ±.0003 | 0.0147 ±.0003 | 0.0189 ±.0003 | 0.0253 ±.0003 |
| | RPG (repro) | 0.0188 ±.0000 | 0.0234 ±.0001 | 0.0278 ±.0001 | 0.0340 ±.0001 |
| | LLaDA-Rec (repro) | 0.0178 ±.0002 | 0.0229 ±.0004 | 0.0282 ±.0002 | 0.0350 ±.0003 |
| | **MaLaCo (ours)** | **0.0189** ±.0001 | **0.0243** ±.0001 | **0.0301** ±.0002 | **0.0380** ±.0003 |

explain the source of the saving; they are not latency estimates, and the measured batch-20 ratio is $24.9\times$, falling to $2.4\times$ at batch 1. Section F sweeps batch size, beam width and history length, reports peak memory for both systems, and adds a matched-decoder control.

Table 10: Inference cost per query. Counts are exact from the two configurations. Latency is measured on one A100-40GB in fp32, without TF32 or key–value caching, with beam 50 on both sides, CUDA-synchronized, at batch 20 (amortized throughput, their released `test_batch_size`) and at batch 1 (single-request latency); peak memory is reported at batch 20 only and comes from a separate memory-instrumented pass. MaLaCo is timed on synthetic inputs, LLaDA-Rec over the full test split inside its own evaluation loop. LLaDA-Rec batches its beams, but still processes every beam sequence at every denoising step; MaLaCo processes one history sequence and confines beam expansion to the short SID head. Section F gives the surrounding sweep and a matched-decoder control that separates the decoding pattern from implementation differences.

| | **LLaDA-Rec** | **MaLaCo (ours)** |
|---|---|---|
| Generation mechanism | iterative masked denoising | single-pass completion |
| Sequence representation | 80 discrete SID tokens | 20 latent rows |
| Denoising steps $T$ | 4 | **1** |
| Beam width $B$ | 50 | 50 |
| *Deep backbone* (both $d_{\text{model}} = 256$; theirs 4 layers, ours 2) | | |
| batched forward calls | $T = 4$ | **1** |
| sequences processed / query | $T \times B = \mathbf{200}$ | **1** |
| positions per sequence | 84 | 20 |
| position-evaluations | 16,800 | 20 |
| *Where the beam search runs* | | |
| module searched | the full backbone | the SID head only |
| head position-evals | — | 352 |
| *Measured cost* (A100-40GB, beam 50) | | |
| ms / query, batch 20 | 13.097 | **0.526** |
| speed-up | — | **24.9×** |
| ms / query, batch 1 | 24.453 | **9.984** |
| speed-up | — | **2.4×** |
| peak MiB, batch 20 | 3,013 | **92** |
| Total parameters | 4.75M | 6.19M (2.71 + 3.49) |

Note we are the *larger* model (6.19M vs. 4.75M): the advantage is invocation structure, not parameter count.

# D Semantic-ID Analyses

## D.1 Metric definitions: excess collision, purity, and mutual information

This subsection defines every quantity reported in Tables 4, 11, 12 and 14 precisely; the main text and table captions use them only in compressed form.

**Excess collision.** For a catalog of $N$ items with codes $s_1, \ldots, s_N \in \{1, \ldots, K\}^L$, excess collision is $(N - |\{s_i\}|)/N$: the fraction of items that would have to be merged with some other item to make every code unique. It is 0 exactly when the assignment is injective.

**Purity, and why it cannot be compared across datasets.** Fix an item attribute $A$ external to the tokenizer (e.g. department), with realizations $a \in \mathcal{A}$, and partition the catalog into buckets by code value ($c_0$ alone, or the pair $(c_0, c_1)$). Writing $n_k$ for a bucket's size and $n_{k,a}$ for the number of its items with label $a$,

$$\text{purity} = \sum_k \frac{n_k}{N} \max_a \frac{n_{k,a}}{n_k},$$

the size-weighted fraction of each bucket sharing its single most common label. Purity is bounded below by the *chance purity* $\max_a n_a/N$ — the score one dummy bucket achieves by always guessing the catalog's most common department — and this floor is a property of the label distribution alone, not of the tokenizer. Chance purity ranges from 0.140 on Scientific (55 departments) to 0.466 on Instrument (31, more skewed; Table 14), so the raw purities 0.841 and 0.842 the two datasets separately achieve are a 6.0× and a 1.8× lift over chance respectively. Comparing raw purity across

datasets therefore compares numbers with different floors; we report lift, and, more usefully, the normalized quantity below.

**Entropy and mutual information.** Let $\hat{p}(a)$ be $A$'s empirical marginal over the catalog. *Entropy* $H(A) = -\sum_a \hat{p}(a) \log_2 \hat{p}(a)$ is the average number of bits needed to state a uniformly drawn item's label given only this marginal — a property of the labels alone, independent of any tokenizer, equal to $\log_2 |\mathcal{A}|$ when every label is equally likely and smaller when the distribution is skewed. *Mutual information*

$$\mathrm{MI}(c_0; A) = H(A) - H(A \mid c_0) = \sum_{k,a} \hat{p}(c_0{=}k, a)\ \log_2 \frac{\hat{p}(c_0{=}k, a)}{\hat{p}(c_0{=}k)\,\hat{p}(a)}$$

is the number of bits of that uncertainty removed, on average, once the level-0 code is known. It is symmetric in its two arguments, non-negative, and exactly $0$ when the code and the label are statistically independent — intuitively, it is what a purity-style score would look like if it accounted for how surprising each match actually is, rather than just counting matches.

**The uncertainty coefficient $U$.** Raw mutual information inherits the same cardinality sensitivity as purity: a codebook with more distinguishable buckets can trivially carry more bits about a fine-grained attribute, and MI has no fixed upper bound to compare it against. We therefore report

$$U(A \mid c_0) = \frac{\mathrm{MI}(c_0; A)}{H(A)} \in [0, 1],$$

Theil's uncertainty coefficient: the *fraction* of the attribute's own entropy that the code resolves. Normalizing by $H(A)$ itself makes $U$ comparable across attributes and datasets with different numbers of categories, which raw purity is not. On Scientific, $U(\text{department} \mid c_0) = 0.818$ against Instrument's $0.736$: despite Instrument's higher raw purity ($0.842$ versus $0.841$) and much higher chance floor ($0.466$ versus $0.140$), the level-0 code resolves *more* of Scientific's department uncertainty in relative terms. This is the reading the main text quotes; raw purity alone would suggest the opposite ordering.

**Baseline input space.** Table 4's *RQ-VAE + Sinkhorn, last level* row is the published recipe run on the raw item embedding, not on our PCA-128 input — so its collision and purity numbers reflect a different input space as well as a different quantizer. Table 13 isolates the quantizer alone by matching the input space to PCA-128 for both methods.

**Why deduplicate only the last level?** Table 11 isolates this design choice on Scientific. Running Sinkhorn deduplication on *every* level removes more collision than the shipped recipe ($0.2515\%$ versus $0.6809\%$ excess), but it does so by reassigning coarse codes to satisfy a balance constraint unrelated to semantic content: level-0 department purity falls from $0.842$ to $0.621$ and $U$ from $0.819$ to $0.482$, roughly halving how much department information the coarse code carries. Deduplicating only the last level leaves the coarse hierarchy untouched *exactly* rather than approximately: $c_0$ and $c_1$ are byte-identical before and after for all $25{,}848$ items, so every column except collision is unchanged. The SID autoencoder then closes the remaining gap to injectivity, editing the level-0 code of $0.10\%$ of items.

Table 11: Why deduplicate only the last level? Scientific, same cascade and metrics as Table 4 (definitions in Section D.1). Running Sinkhorn deduplication on every level removes more collision than restricting it to the last level, but only by reassigning coarse codes to satisfy a balance constraint unrelated to semantic content, which roughly halves department purity and $U$.

| Output vocabulary | Excess coll. | $c_0$ dept. pur. | $(c_0 c_1)$ leaf pur. | $U(\text{dept} \mid c_0)$ |
|---|---|---|---|---|
| Balanced residual $k$-means | $25.5455\%$ | $0.842$ | $0.835$ | $0.819$ |
| + Sinkhorn dedup, *last level* (shipped) | $\mathbf{0.6809\%}$ | $\mathbf{0.842}$ | $\mathbf{0.835}$ | $\mathbf{0.819}$ |
| + Sinkhorn dedup, *all levels* (control) | $0.2515\%$ | $0.621$ | $0.733$ | $0.482$ |

### D.2 Collision cascade across datasets

Table 12 reports the complete construction cascade on all four catalogs. Last-level Sinkhorn deduplication removes most conflicts, and the SID autoencoder plus catalog assignment closes the remaining

Table 12: Collision cascade and refinement edits, all four datasets, full catalog ($L = 3$, $K = 256$). *Upper block*: excess collision $(N - |\text{unique}|)/N$ after each cumulative stage. *Lower block*: the fraction of items whose level-$l$ code the last stage changed — monotone in depth everywhere, so the stage that buys injectivity edits the coarse codes hardly at all. The Arts column is the champion tokenizer, fitted on all 89,958 items at once; Table 16 reports the incremental construction, which fits on the 67,468 existing items and reaches two excess identifiers rather than eleven.

| | **Scientific** | **Instrument** | **Game** | **Arts** |
|---|---|---|---|---|
| Catalog size $N$ | 25,848 | 24,587 | 25,612 | 89,958 |
| *Excess collision, cumulative stages* | | | | |
| Balanced residual $k$-means | 25.55% | 18.12% | 18.34% | 31.04% |
| + Sinkhorn dedup | 0.681% | 0.557% | 0.980% | 1.557% |
| + SID-AE + assignment | **0.0039%** | **0.0000%** | **0.0078%** | **0.0122%** |
| excess items | 1 | 0 | 2 | 11 |
| items sharing a tuple | 2 | 0 | 4 | 19 |
| *Items whose code the last stage changed* | | | | |
| Level 0 | 0.10% | 0.17% | 0.28% | 0.21% |
| Level 1 | 0.21% | 0.27% | 0.31% | 0.54% |
| Level 2 | 0.94% | 0.81% | 1.29% | 2.33% |
| Any level | 1.06% | 0.99% | 1.55% | 2.63% |

Table 13: Input-matched quantizer comparison on Scientific ($K = 256$, $L = 3$). All arms use the same SID-AE and beam refinement. The middle column trains the RQ-VAE from the same PCA-128 information source used by balanced $k$-means. "Matched" does not equalize the internal dimension: RQ-VAE compresses to 64 dimensions before quantization, whereas balanced $k$-means operates directly in 128 dimensions. Sinkhorn deduplication is omitted in all three arms to isolate the quantizer.

| | **RQ-VAE + SID-AE + beam** | | **Ours** |
|---|---|---|---|
| **Metric** | *on* 3584-*d* | *on PCA*-128 *(matched)* | *bal. k-means on PCA*-128 |
| Excess collision % | 0.031 | **0.000** | 0.031 |
| $c_0$ codes used | 256 | 256 | 256 |
| *Department purity, per level* | | | |
| $c_0$ | 0.787 | 0.789 | **0.836** |
| $c_1$ | 0.206 | **0.395** | 0.376 |
| $c_2$ | 0.175 | 0.185 | **0.254** |
| *Leaf-category purity by prefix* | | | |
| $(c_0)$ | 0.413 | 0.489 | **0.544** |
| $(c_0, c_1)$ | **0.897** | 0.855 | 0.830 |
| $(c_0, c_1, c_2)$ | 1.000 | 1.000 | 1.000 |
| $U(A \mid c_0)$ — *fraction of A determined by* $c_0$ | | | |
| department | 0.768 | 0.757 | **0.804** |
| sub-category | 0.720 | 0.751 | **0.788** |
| leaf category | 0.660 | 0.698 | **0.734** |
| brand/store | 0.531 | 0.538 | 0.559 |

two orders of magnitude. The final refinement edits the level-0 code for at most 0.28% of items and concentrates its changes at the last level, supporting the claim that coarse semantic organization is preserved while collisions are removed.

### D.3 Input-matched quantizer comparison

The matched comparison fixes the PCA-128 information source, although the internal dimensions still differ: the RQ-VAE compresses to 64 dimensions, whereas balanced $k$-means acts directly in 128 dimensions. Balanced $k$-means has higher level-0 department purity (0.836 versus 0.789), while the matched RQ-VAE has zero excess collision and higher level-1 department purity. The two tokenizers are therefore complementary rather than uniformly ordered.

Table 14: Taxonomy probes for the shipped SK-hybrid tokenizer ($L = 3$, $K = 256$), all four datasets. Purity is the size-weighted fraction of each bucket sharing its dominant label; $U(A \mid c_0) = \mathrm{MI}(c_0; A)/H(A)$ is the fraction of attribute $A$ determined by the level-0 code. ***Raw purity is not comparable across columns***: chance purity, the majority department's share, ranges from $0.140$ on Scientific to $0.466$ on Instrument, so the same $0.84$ is a $6.0\times$ lift on one and $1.8\times$ on the other. The $U$ block is cardinality-fair and is what the main text quotes. Every arm here is descriptive — these labels are never a training target and never a retrieval target.

| Metric | Scientific | Instrument | Game | Arts |
|---|---|---|---|---|
| Catalog size | 25,848 | 24,587 | 25,612 | 89,958 |
| Departments | 55 | 31 | 47 | 33 |
| Chance $c_0$ purity | 0.140 | 0.466 | 0.457 | 0.205 |
| *Department purity, per level* | | | | |
| $c_0$ | 0.841 | **0.842** | 0.668 | 0.804 |
| lift over chance | **6.0×** | 1.8× | 1.5× | 3.9× |
| $c_1$ | 0.393 | 0.544 | **0.589** | 0.425 |
| $c_2$ | 0.208 | 0.484 | **0.485** | 0.257 |
| *Leaf-category purity by prefix* | | | | |
| $(c_0)$ | 0.553 | 0.590 | **0.694** | 0.601 |
| $(c_0, c_1)$ | 0.835 | 0.832 | **0.839** | 0.801 |
| $(c_0, c_1, c_2)$ | 1.000 | 1.000 | 1.000 | 1.000 |
| $U(A \mid c_0)$ — *fraction of A determined by* $c_0$ | | | | |
| department | **0.818** | 0.736 | 0.395 | 0.713 |
| sub-category | **0.800** | 0.752 | 0.457 | 0.729 |
| leaf category | **0.743** | 0.723 | 0.585 | 0.723 |
| brand/store | 0.562 | **0.567** | 0.512 | 0.499 |

### D.4 Semantic hierarchy across datasets

Raw purity cannot be compared directly across datasets because the majority department share ranges from $0.140$ on Scientific to $0.466$ on Instrument. The normalized $U(A \mid c_0) = I(c_0; A)/H(A)$ rows provide the cardinality-aware view used in the main text. These labels are descriptive probes only: they are never training or retrieval targets.

### D.5 Controlled auxiliary-loss ablation

Table 15: Loss composition on Scientific with vocabulary and capacity fixed. *Top:* given the SID head, the auxiliary continuous loss is ordered contrastive > none. *Bottom:* removing the SID objective entirely leaves only a contrastive projection to retrieve from, and costs roughly two-thirds of ranking quality — which is why the discrete read-out exists.

| Objective | @1 | @5 | @10 | @20 | @50 | n | read-out |
|---|---|---|---|---|---|---|---|
| **SID + InfoNCE (ours)** | **0.0089** ±.0004 | **0.0314** ±.0013 | **0.0499** ±.0011 | **0.0750** ±.0019 | **0.1173** ±.0018 | 4 | SID beam |
| SID only (balanced $k$-means tokenizer) | 0.0079 ±.0003 | 0.0280 ±.0007 | 0.0455 ±.0006 | 0.0690 ±.0010 | 0.1111 ±.0017 | 4 | SID beam |
| InfoNCE only, no SID head† | 0.00194 ±.00016 | 0.00925 ±.00032 | 0.01589 ±.00023 | 0.02657 ±.00038 | 0.04900 ±.00017 | 4 | cosine |

The first two rows differ only in the auxiliary objective—vocabulary, capacity, SID head, beam read-out and seeds are held fixed—so the gap isolates InfoNCE rather than any tokenizer or decoder change. It helps at every depth, but not uniformly: the absolute gain grows with $k$, from $+0.0010$ at @1 to $+0.0062$ at @50, while the relative gain falls from about 13% to 6%. The contrastive term therefore sharpens the ordering among candidates the beam already reaches more than it widens what the beam reaches at all.

The third row removes the SID objective instead of the auxiliary one. With `sid_loss_weight=0` no discrete head is constructed, so the only available read-out is full-catalog cosine retrieval over the contrastive projection; it is R@$k$ rather than SID-beam H@$k$, and the two families are not comparable cell for cell. What is comparable is the profile: retention against the full model *rises* with depth, from

22% at @1 to 32% at @10 and 42% at @50, so the continuous projection degrades worst precisely at the head of the ranking. Read together the two comparisons agree from opposite directions—rank-1 is where the discrete structure and the continuous signal each contribute proportionally most, and neither alone recovers it. This is also the complement of Table 9, where the factorized model gains at depth and concedes rank-1: the auxiliary signal is what buys back head precision.

One caveat on precision. The SID+InfoNCE row is evaluated on the full test set while the other two use the 5,120-row subsample, so the orderings and the shape of both profiles are reliable, but the individual magnitudes should not be read to the fourth decimal.

# E Catalog-Growth Details

Table 16: Where collisions are removed, Arts. Each row applies one stage of the naming pipeline to the 67,468 existing items, except the last, which additionally inserts the 22,490 new items by deconfliction. Balanced $k$-means alone leaves more than a quarter of the catalog sharing a tuple; Sinkhorn deduplication and the SID autoencoder remove almost all of it. The final stage — the one that grows the catalog by a third — introduces *no* new collision, and the two residual duplicates are the same two already present among the existing items.

| | Stage | Items | Unique tuples | Duplicates | Collision |
|---|---|---|---|---|---|
| 1 | Balanced residual $k$-means | 67,468 | 48,303 | 19,165 | 28.41% |
| 2 | + Sinkhorn deduplication | 67,468 | 66,588 | 880 | 1.30% |
| 3 | + SID autoencoder, beam-8 | 67,468 | 67,466 | 2 | 0.0030% |
| 4 | + deconflicted extension (+22,490 new items) | 89,958 | 89,956 | 2 | **0.0022%** |

Stage 4 introduces 0 collisions: no two new items share a tuple, and no new item lands on an occupied existing tuple. The 2 residual duplicates are inherited from stage 3; the existing block is byte-identical throughout.
Greedy assignment without deconfliction introduces 9,032 collisions here.

Table 17: Freezing the semantic vocabulary across catalog growth, Arts. The naming pipeline is fitted on the oldest 67,468 items; the newest 22,490 are inserted by deconfliction alone. *Top:* the vocabulary extends cleanly, and does so deterministically. *Bottom:* naming an item is not the same as retrieving it — a frozen sequence model never emits the new codes, and random unoccupied codes score identically, so semantic placement buys nothing while the decoder is frozen. Values in panel (b) are four-seed means ± standard deviations evaluated on the complete test set.

*(a) Extending the vocabulary to 22,490 unseen items*

| Candidate width | Coverage | Introduced collisions | Level-0 agreement |
|---|---|---|---|
| $W=8$ | 0.9844 | 0 | 0.7796 |
| $W=16$ | 0.9989 | 0 | 0.7810 |
| $W=32$ | 0.9999 | 0 | 0.7812 |
| $W\geq 64$ | **1.0000** | **0** | **0.7812** |

Existing SIDs are byte-identical throughout (identifier churn = 0).
Level-0 agreement: fraction of new items sharing a first code with their nearest existing neighbour in latent space; chance is 0.0083.
Greedy assignment without deconfliction introduces 9,032 collisions at $W=256$.

*(b) Retrieval on the extended catalog (R@10, beam 50)*

| Sequence model | Vocabulary | New $n$=50,542 | Existing $n$=146,744 | Overall $n$=197,286 |
|---|---|---|---|---|
| frozen | extended, deconflicted | 0.0000 | 0.0579 ±0.0004 | 0.04305 ±0.00027 |
| frozen | extended, random codes | 0.0000 | 0.0579 ±0.0004 | 0.04305 ±0.00027 |
| head adapted | extended, deconflicted | 0.0105 ±0.0004 | 0.0491 ±0.0004 | 0.03921 ±0.00035 |
| retrained | extended, deconflicted | 0.0098 ±0.0003 | 0.0575 ±0.0007 | 0.04526 ±0.00057 |
| retrained | refitted on full catalog | 0.0110 ±0.0003 | 0.0580 ±0.0002 | 0.04598 ±0.00023 |

**Item-arrival partition and fitting discipline.** We partition items by the timestamp of their first recorded interaction. Timestamps are available for all $89{,}958$ Arts items. The oldest $67{,}468$ items form the existing catalog and the newest $22{,}490$ form the arriving block. This is a catalog-age partition, not a global chronological split: the benchmark's per-user leave-one-out assignment is unchanged, so the experiment isolates whether a vocabulary fitted on an older catalog can name a later block without identifier churn, and does not simulate end-to-end deployment or temporal drift in user behaviour. The PCA-128 basis, whitening statistics, balanced centroids, and SID autoencoder are fitted on existing items only; new-item embeddings are projected into this frozen space before assignment. The unchanged leave-one-out test set contains $50{,}542$ new-item targets and $146{,}744$ existing-item targets.

**What each arm trains on.** The three regimes in Table 17(b) differ in which rows the recommender sees, so we state each separately.

The *frozen* arms reuse a recommender trained only on histories from which every arriving item has been removed. Filtering drops 37,779 of the $197{,}286$ users ($19.15\%$), reducing the corpus from $912{,}046$ to $713{,}699$ rows across the four splits and the training split itself from $320{,}188$ to $235{,}178$ windows; a leakage check confirms that no arriving item occurs in any of them. Filtering applies to training sequences only: evaluation uses the original, unfiltered test rows, so test histories retain their arriving items and a user dropped from training is still evaluated. These arms measure what a deployed model does when an item has been named but never learned.

The *head-adapted* arm keeps that backbone frozen and updates only the semantic-ID head, under the semantic-ID cross-entropy alone: neither the InfoNCE term nor any reconstruction term is applied, and the backbone is asserted to remain in evaluation mode throughout. Optimization uses learning rate $10^{-4}$ and batch $512$. Its supervision comes from the *validation* split: real user histories whose held-out target is an arriving item, so the hidden states the head is tuned on match those it meets at evaluation. Each batch is a $75/25$ mixture, three quarters of it existing-item histories drawn from the same split; without this rehearsal the head trades away existing-item recall far more steeply. Because the arm consumes the validation split for training, it selects nothing on validation. Disjointness from the test rows is asserted three ways — window key, full item sequence, and per-user target identity — and any row colliding with another user's test window is removed before training. A zero-step control must reproduce the frozen numbers, otherwise the run is discarded.

Because that arm trains on the validation split, no held-out data remains on which a step count could be chosen. We therefore fixed the budget in advance rather than selecting it: the sweep ran $\{0, 500, 1000, 2000, 5000\}$ steps for every seed, and the table reports the endpoint. New-item recall rises monotonically across the sweep ($0.0000$, $0.0025$, $0.0052$, $0.0076$, $0.0105$) while existing-item recall falls monotonically ($0.0579$, $0.0564$, $0.0556$, $0.0540$, $0.0491$), so the endpoint is the most favourable point in the sweep for this arm on the new-item slice and the least favourable on the other two. Reporting it therefore states a bound rather than a tuned result: even at its most new-item-favourable setting, head-only adaptation leaves pooled Recall@10 at $0.03921$, below the frozen baseline of $0.04305$, and pooled recall exceeds that baseline at no step count in the sweep — the best pooled point, $1000$ steps, reaches $0.04271$.

The *retrained* arm differs in kind. The recommender is trained from scratch on the full, unfiltered corpus of $197{,}286$ users and $912{,}046$ rows, of which $320{,}188$ are training windows and the remaining three splits hold $197{,}286$ each. Arriving items do occur in those training windows: $813{,}901$ occurrences across $110{,}474$ of the $320{,}188$. Throughout, a *window* is one training example, history and target together, and an *occurrence* counts every arriving-item position inside such examples. Only the naming pipeline remains fitted on the $67{,}468$ existing items. Its comparison partner is a recommender trained on exactly the same rows with a refitted full-catalog vocabulary, so the two differ only in where the identifiers came from.

No arriving item occurs in the initial recommender-training corpus used by either the frozen or the head-adapted arm. The frozen arm receives no further supervision of any kind. The head-adapted arm then updates the SID head on the former validation rows whose targets are arriving items, so it *does* see arriving-item supervision during adaptation, though never during backbone training. The retrained arm trains the whole recommender from scratch on the unfiltered corpus, where arriving items appear in both histories and targets. The zero-exposure statement therefore covers the frozen arm and the head-adapted arm before adaptation, and neither the adaptation phase nor the retrained arm. Note

also that the split is over the *catalog*, not the timeline: the leave-one-out train, validation and test assignment is unchanged from the main experiments, which keeps these rows directly comparable to Table 9 and separates the effect of an item being new from any drift in user behaviour.

**Collision-free insertion relative to the existing table.** Table 16 shows that balanced residual $k$-means leaves $28.41\%$ excess collision, last-level deduplication reduces it to $1.30\%$, and the SID autoencoder reduces it to two duplicate tuples among the existing items. Inserting all $22{,}490$ new items introduces no additional collision and leaves every existing SID unchanged. The two duplicates in the extended table are inherited from the existing block, not created by insertion.

**Assignment coverage and retrieval.** Table 17(a) reaches complete assignment coverage at $W \geq 64$ with zero introduced collisions. Moreover, $78.1\%$ of new items share their first code with their nearest existing latent neighbor, compared with a $0.83\%$ chance rate. These are deterministic properties of the assignment.

Table 17(b) separates naming from recommendation. A frozen sequence model obtains zero new-item Recall@10, and random unoccupied codes produce the same result. Head-only adaptation recovers new-item recall but reduces existing-item and pooled recall. The frozen arm differs from the refitted champion by only $-0.26\%$ on the existing-item slice ($0.05788$ versus $0.05803$), despite never having trained on the arriving interactions. This pair differs in both vocabulary and training data, so it isolates neither factor. We report it only to show that withholding the arriving items leaves performance on the rest of the catalog nearly unchanged, not as evidence of statistical equivalence. The vocabulary is isolated by the retrained pair, where both arms see the same rows. After full retraining, the extended-vocabulary arm obtains $0.04526 \pm 0.00057$ overall Recall@10, compared with $0.04598 \pm 0.00023$ for the refitted-vocabulary arm, a 1.6% relative difference. Their existing-item means differ by $0.0005$ ($0.0575 \pm 0.0007$ versus $0.0580 \pm 0.0002$), while the larger remaining difference is concentrated in newly inserted items.

## F Latency: Sweep and Matched-Decoder Control

Table 3 reports a single operating point. This appendix reports the surrounding sweep, states the measurement protocol in full, and adds a control that separates two claims the headline ratio conflates.

**What the headline number is, and is not.** The headline $0.526$ ms figure is a batch of 20 queries divided by 20. It is an *amortized* cost under batching, not the latency of a single request, and the $25\times$ ratio it produces is a property of that batch size rather than of the method. Both systems were re-timed here on one device across batch sizes (Table 19), and the ratio moves with it: $2.4\times$ at batch 1, $5.8\times$ at 4, $24.9\times$ at 20, $43.1\times$ at 64 and $52.9\times$ at 128. The reason is visible in Fig. 3a. MaLaCo amortizes steeply, from $9.984$ to $0.208$ ms per query between batch 1 and 256, a factor of 48. LLaDA-Rec barely amortizes at all, from $24.453$ to $12.617$ ms, a factor of $1.9$, because its cost is dominated by re-encoding 50 hypotheses through the backbone at every denoising step and batching cannot remove that work. A single ratio therefore cannot summarise this comparison; the operating point has to be stated with it.

**Online latency.** At batch 1 — one request, nothing to amortize — MaLaCo costs $9.984$ ms per query against LLaDA-Rec's $24.453$, a reduction of $2.4\times$. This is the figure to read as latency. The $24.9\times$ at batch 20 is a throughput figure at that system's released `test_batch_size`. Both are measured; they answer different questions.

**A matched-decoder control.** A ratio between two codebases cannot separate the architecture from the implementation. We therefore add a control that holds everything fixed except the decoding pattern: the *same* backbone, the *same* weights, the *same* harness, device and precision, but invoked the way a masked diffusion decoder must invoke it — $T = 4$ forward passes, each over batch $\times$ beam rows, because every hypothesis is re-encoded at every denoising step. Any gap between MaLaCo and this control is attributable to *where beam search runs* and to nothing else. The control is deliberately conservative: it re-encodes our 20-row history rather than LLaDA-Rec's 84-position sequence, and it uses our two-layer backbone rather than their four, so it understates the cost of the pattern it models.

Table 18: Measurement protocol and environment for Table 19 and Fig. 3.

| Setting | Value |
|---|---|
| GPU | A100-SXM4-40GB |
| Driver | 595.91.07 |
| PyTorch / CUDA / cuDNN | 2.9.1+cu128 / 12.8 / 91002 |
| Python | 3.11.0 |
| Precision | fp32, TF32 matmul disabled, no autocast |
| KV caching | none (all position-evaluations counted) |
| Warmup | 5 calls, discarded |
| Timed calls | 30 per run, 3 runs |
| Reduction | median within run, median across runs |
| Synchronization | `torch.cuda.synchronize` around every timed region |
| Inputs | synthetic histories of the benchmarked shape |
| Denoising steps (control) | 4 |

Table 19: Generation cost across batch size, beam width and history length on Scientific. *MaLaCo* runs the backbone once and confines beam search to the SID head; the *matched control* is the same backbone and the same weights re-entered once per hypothesis per denoising step, so the two differ only in where search runs. LLaDA-Rec was re-timed on the same device, but only along the batch axis, and the two reasons differ: its `test_batch_size` is *derived* from the beam width ($\lfloor \text{train_batch_size}/\text{num_beams} \rfloor$), so a beam sweep would move batch size with it and vary two factors at once; and its history length is a model dimension fixed at training time (the checkpoint is `Seq-20`), so shorter histories would require retraining rather than re-timing. Both of our arms share one backbone that accepts either at inference. Hence — means *not measured*, and is distinct from OOM, which was measured and did not fit. $\star$ marks the operating point reported in Table 3. Medians over 3 runs of 30 timed calls for the first two arms, and over the full test split for LLaDA-Rec.

| | | | MaLaCo | | | Matched control | | LLaDA-Rec | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Batch | Beam | Hist | batch ms | ms/query | MiB | ms/query | ratio | ms/query | MiB | ratio |
| *varying batch size* | | | | | | | | | | |
| 1 | 50 | 20 | 9.98 | 9.984 | 51 | 9.093 | 0.91 | 24.453 | 176 | 2.4 |
| 4 | 50 | 20 | 9.83 | 2.458 | 57 | 2.699 | 1.10 | 14.245 | 626 | 5.8 |
| 20 | 50 | 20 $\star$ | 10.53 | 0.526 | 92 | 1.715 | 3.26 | 13.097 | 3013 | 24.9 |
| 64 | 50 | 20 | 18.83 | 0.294 | 187 | 1.602 | 5.44 | 12.685 | 9578 | 43.1 |
| 128 | 50 | 20 | 30.54 | 0.239 | 326 | 1.564 | 6.56 | 12.617 | 19131 | 52.9 |
| 256 | 50 | 20 | 53.29 | 0.208 | 599 | 1.547 | 7.43 | OOM | | |
| *varying beam width* | | | | | | | | | | |
| 20 | 10 | 20 | 10.05 | 0.503 | 58 | 0.542 | 1.08 | — | — | — |
| 20 | 20 | 20 | 9.97 | 0.498 | 67 | 0.823 | 1.65 | — | — | — |
| 20 | 50 | 20 $\star$ | 10.53 | 0.526 | 92 | 1.715 | 3.26 | 13.097 | 3013 | 24.9 |
| *varying history length* | | | | | | | | | | |
| 20 | 50 | 5 | 10.49 | 0.524 | 92 | 0.558 | 1.06 | — | — | — |
| 20 | 50 | 10 | 10.43 | 0.522 | 92 | 0.888 | 1.70 | — | — | — |
| 20 | 50 | 20 $\star$ | 10.53 | 0.526 | 92 | 1.715 | 3.26 | 13.097 | 3013 | 24.9 |

**Decomposing the $25\times$.** At the reported batch-20 operating point, MaLaCo costs $0.526$ ms per query, compared with $1.715$ ms for the matched control and $13.097$ ms for LLaDA-Rec. Thus, moving beam-dependent computation out of the backbone provides a $3.26\times$ throughput improvement in the matched implementation at this operating point. The additional $7.6\times$ difference between the matched control and LLaDA-Rec reflects backbone depth, sequence length, and implementation differences. The matched advantage is not a constant: it is $0.91\times$ at batch $1$, $1.10\times$ at batch $4$, $3.26\times$ at batch $20$, and $7.43\times$ at batch $256$. Factorization therefore primarily improves amortized throughput under batching; it does not provide a matched single-request latency advantage.

**The advantage is structural, and it grows.** The position-evaluation model predicts that MaLaCo's advantage should widen with anything that multiplies the work done per hypothesis. It does, on both axes we can vary. Against the control the ratio rises from $1.08$ to $1.65$ to $3.26$ as the beam widens from $10$ to $20$ to $50$, and from $1.06$ to $1.70$ to $3.26$ as the history grows from $5$ to $10$ to $20$ rows (Fig. 3c). Both are consequences of the same fact: the control's cost scales with beam $\times$ history while MaLaCo's backbone cost does not scale with the beam at all.

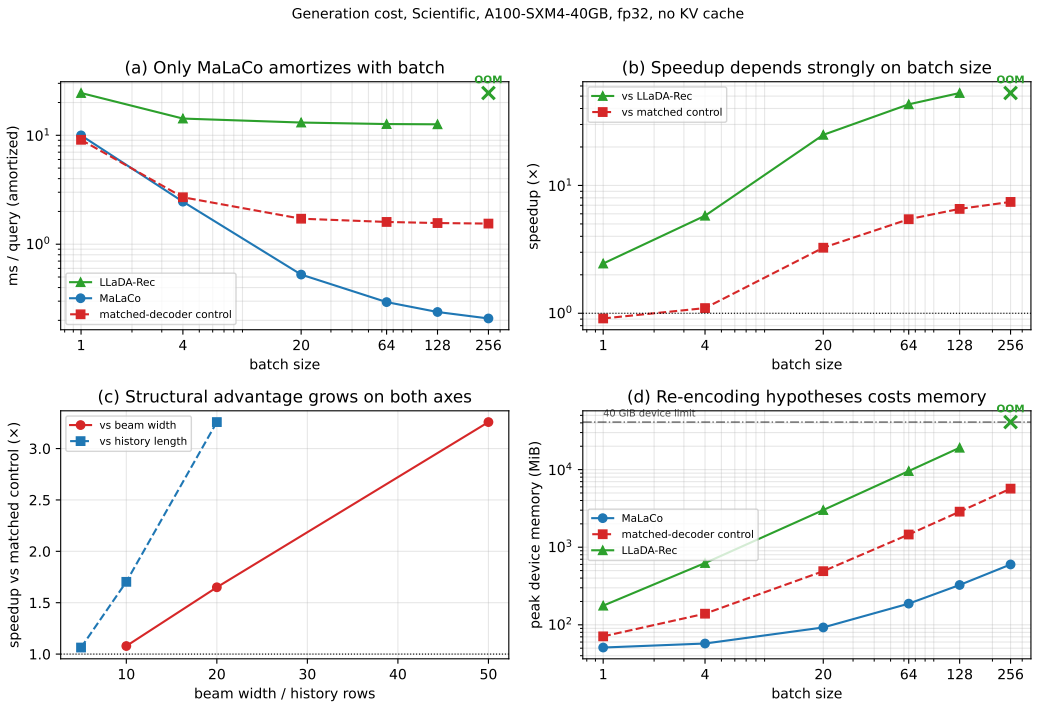



Figure 3: Generation cost on Scientific. (a) Amortized cost per query against batch size; MaLaCo falls by $48\times$ across the range while LLaDA-Rec falls by $1.9\times$. (b) The resulting speedup, which grows from $2.4\times$ to $52.9\times$ with batch size alone. (c) Speedup over the matched-decoder control as beam width and history length grow. (d) Peak device memory; LLaDA-Rec reaches 19.1 GB at batch 128 and exhausts the 40 GB device at 256, where MaLaCo uses 599 MiB. All MaLaCo and control points are medians over 3 runs of 30 timed calls on one A100-SXM4-40GB in fp32 without KV caching; LLaDA-Rec points are medians over the full test split on the same device.

**Where it disappears.** At batch 1 the ratio against the control inverts: MaLaCo costs 9.984 ms against the control's 9.093, a factor of 0.91. Neither arm is compute-bound at that size, and MaLaCo's beam search runs three sequential levels through the head, so it pays more kernel-launch latency than four wide backbone passes do. Against the real baseline MaLaCo is still $2.4\times$ faster there, because LLaDA-Rec's deeper backbone and longer sequences cost more than either of our arms. The factorization buys throughput under batching; against a matched decoder it does not buy single-request latency. We report this because it bounds the claim, not because it is favourable.

**Memory, and where the baseline stops.** Re-encoding hypotheses is expensive in memory as well as time, and this is the clearest separation in the study. LLaDA-Rec's peak allocation grows almost linearly with batch — 176, 626, 3,013, 9,578 and 19,131 MiB at batches 1 to 128 — while MaLaCo's grows from 51 to 326 MiB over the same range. The gap widens from $3\times$ to $59\times$ (Fig. 3d). At batch 256 LLaDA-Rec does not run at all: it exhausts the 39.67 GB device, holding 37.54 GB when a further 4.10 GB allocation fails, because a single denoising step must materialize $256 \times 50 = 12{,}800$ sequences of 84 positions. Linear extrapolation from batch 128 predicts 37.4 GB, matching the observed failure point, so this is a genuine capacity limit rather than an allocator artifact. MaLaCo runs the same workload in 599 MiB. We did not bisect the exact cutoff, so all we establish is that it lies between 128, which fits, and 256, which does not. The throughput ceiling for LLaDA-Rec is therefore set by memory rather than by time, and it is reached at a batch size well below MaLaCo's.

**Reproducibility and limits.** Table 18 records the device, driver, library versions, precision, warmup, repeat counts and reduction. Run-to-run variability is small — across every configuration the standard deviation of the batch time over the three runs stays below 0.13 ms, against batch times of 10 to

53 ms — so the ordering is not an artifact of noise, though repeated whole-sweep reruns move the operating point by up to 3%, which is the practical repeatability of these measurements. The re-timing reproduces the previously archived numbers on this device to 2.6% for MaLaCo (0.526 against 0.540) and 0.06% for LLaDA-Rec (13.097 against 13.089). Three limits remain. First, LLaDA-Rec is swept over batch size only, and only to batch 128: its `test_batch_size` is derived from the beam width in the released pipeline, so beam and batch cannot be varied independently on that side, and batch 256 exceeds device memory. The beam and history rows of Table 19 are therefore MaLaCo-versus-control only. Second, history length cannot exceed the trained 20 rows, because the positional embedding is a learned table of that size. Third, every number is from one device, one precision and one implementation of each system, and neither implementation is written for serving; the ratios are specific to that setting.

## G Masking Policy: A Control for the Training Corruption

The recipe masks a random subset of history positions and asks the model to recover them. A natural question is whether that corruption does any work, or whether the results are carried by the semantic-ID vocabulary and the factored decoder alone. This section reports a masking-policy control that tests its contribution.

**What was varied, and how each arm was selected.** We train two additional arms that differ from the reported model only in the masking policy. The first, *terminal only*, hides just the final item and predicts it. The second, *random one*, hides a single item drawn uniformly at random from the history. Between terminal only and random one, everything else is held fixed: the same tokenizer, backbone and head sizes, SID and InfoNCE terms with the same coefficients, optimizer, batch size, epoch budget, beam width, and seeds. Both arms score exactly one position per window and omit $1/t$ reweighting. Their comparison therefore isolates where the prediction target falls rather than how many targets are scored. The random-subset arm retains the same loss terms and nominal coefficients but scores multiple positions and applies $1/t$ reweighting, so its comparison with random one is not fully factorial. All three arms are evaluated on the complete 50,985-row test set, and all are selected by the same rule: the checkpoint with the highest validation Recall@10. The rule is worth stating, because the alternative — minimum validation loss — lands on epoch 99 for the terminal-only arm, after it has begun to memorise, and scores it at 0.0135 instead of 0.0263. We report each arm under the rule that is more favourable to the ablated baselines. Applying that rule to the reported model moves its Scientific Recall@10 from 0.0499 to 0.0500, which is why the figure here differs marginally from Table 1: both of its checkpoints are late (epochs 279 and 299) and score almost identically.

**The corruption contributes.** On Scientific the three policies order cleanly. Terminal-only training reaches Recall@10 of $0.0263 \pm 0.0006$, random-one $0.0393 \pm 0.0009$, and random subset masking $0.0500 \pm 0.0011$; Recall@50 follows the same order at $0.0714 \pm 0.0007$, $0.0959 \pm 0.0009$ and $0.1173 \pm 0.0009$. The ordering holds in each of the four matched seeds, and the per-seed ranges of the three arms do not overlap at either depth.

**The same pattern with greater per-item training coverage.** Scientific provides 2.62 training windows per item, whereas Musical Instruments provides 3.52. The control could therefore be sensitive to the amount of training supervision available per item. Repeating all three arms on Musical Instruments (3.52 windows per item) reproduces the result closely: Recall@10 runs $0.0349 \pm 0.0011$, $0.0508 \pm 0.0019$, $0.0627 \pm 0.0009$, again monotone, again with disjoint per-seed ranges. The spacing is reproduced as well as the ordering. Terminal only retains 55.6% of the full recipe's Recall@10 on Instrument against 52.6% on Scientific, and the single-position arm accounts for 57% of the gap against 55%. The best-checkpoint epochs follow the same progression on both datasets: 20 for terminal only, 40–50 for random one, and a median of 260 for random-subset masking.

**How exact match is defined.** Training exact match is the fraction of scored positions at which the SID head's per-level argmax, under teacher forcing, matches all three ground-truth codes. It is not a beam-search or ranking metric. Every arm is rescored under a single protocol — the same windows, the same single terminal position masked, the same teacher-forced decoder, in evaluation mode — because the value logged during training is not comparable across policies: the positions it scores

Table 20: Masking policy, four matched seeds per arm. All arms share the tokenizer, model capacity, optimizer, batch size, training budget, beam width, and seeds. Terminal only and random one each score one position per window without $1/t$ reweighting, whereas random subset scores several positions and applies $1/t$ reweighting; each is selected by highest validation Recall@10 and evaluated on the full test set. Exact match is rescored under one protocol: the terminal position masked, teacher-forced, in evaluation mode. *Random one* masks a single position chosen uniformly at random, so it carries the same supervision budget as *terminal only* and separates *where* the mask falls from *how many* masks there are. Recall increases along the ladder, and the collapse at the end of the budget shrinks as corruption increases: by epoch 300 terminal only has lost $45\%$ of its Recall@10 on Scientific and $58\%$ on Instrument, random one $15\%$ and $13\%$, and random subset none on either. Musical Instruments provides greater per-item training coverage (3.52 versus 2.62 training windows per item) and reproduces the ordering, its spacing, and the memorization signature.

| | **Best** | **Exact match** | | | |
|---|---|---|---|---|---|
| **Masking policy** | **epoch** | train | val | **R@10** | **R@50** |
| *Scientific* (50,985-row test set): | | | | | |
| Terminal only | 20 | 0.0458 | 0.0045 | $0.0263 \pm 0.0006$ | $0.0714 \pm 0.0007$ |
| Random one | 50 | 0.0217 | 0.0070 | $0.0393 \pm 0.0009$ | $0.0959 \pm 0.0009$ |
| Random subset | 260 | 0.0349 | 0.0083 | $\mathbf{0.0500} \pm 0.0011$ | $\mathbf{0.1173} \pm 0.0009$ |
| *same runs at epoch* 300*:* | | | | | |
| Terminal only | 300 | 0.9980 | 0.0032 | $0.0144 \pm 0.0004$ | $0.0405 \pm 0.0010$ |
| Random one | 300 | 0.3613 | 0.0058 | $0.0333 \pm 0.0003$ | $0.0824 \pm 0.0006$ |
| Random subset | 300 | 0.0360 | 0.0084 | $0.0502 \pm 0.0007$ | $0.1179 \pm 0.0011$ |
| *Musical Instruments*, same protocol: | | | | | |
| Terminal only | 20 | 0.0419 | 0.0047 | $0.0349 \pm 0.0011$ | $0.0929 \pm 0.0024$ |
| Random one | 40 | 0.0099 | 0.0062 | $0.0508 \pm 0.0019$ | $0.1243 \pm 0.0033$ |
| Random subset | 260 | 0.0300 | 0.0081 | $\mathbf{0.0627} \pm 0.0009$ | $\mathbf{0.1478} \pm 0.0007$ |
| *same runs at epoch* 300*:* | | | | | |
| Terminal only | 300 | 0.9988 | 0.0028 | $0.0148 \pm 0.0002$ | $0.0448 \pm 0.0002$ |
| Random one | 300 | 0.2517 | 0.0064 | $0.0444 \pm 0.0007$ | $0.1119 \pm 0.0013$ |
| Random subset | 300 | 0.0310 | 0.0082 | $0.0626 \pm 0.0006$ | $0.1478 \pm 0.0008$ |

are chosen by the policy under test, so the terminal-only arm is measured on one final slot and the randomly masked arms mostly on interior ones.

**What the exact-match columns show.** They answer two different questions, and separating them matters.

At the checkpoint each arm actually reports, none has memorised, but they already differ in how well they generalise: validation exact match rises along the ladder ($0.0045$, $0.0070$, $0.0083$). We do not read the training column as a fit comparison, because the three arms peak at very different epochs (20, 50 and 260) and it is not ordered; the train-to-validation ratios at those peaks are $10.3$, $3.1$ and $4.2$.

Continuing training separates the arms entirely, and the separation is graded by how much corruption each one sees. By epoch 300 the terminal-only arm reproduces $99.8\%$ of training positions against $0.3\%$ of validation positions, a ratio of 315; random-one reproduces $36.1\%$ against $0.6\%$, a ratio of 62; and random subset masking barely moves, $0.0360$ against $0.0084$, a ratio of $4.3$. The reported metric degrades in the same order: carried to the end of the budget, terminal-only falls from Recall@10 $0.0263$ to $0.0144$, a loss of $45\%$, random-one from $0.0393$ to $0.0333$, a loss of $15\%$, and random subset masking is unchanged at $0.0502$. This is why the arms use their budgets so differently: terminal-only peaks at the first evaluation point we record and never improves over the remaining 280 epochs, whereas random subset masking is still improving near epoch 260. Memorisation therefore explains why the extra budget is wasted; it does not by itself explain the lower peak.

Instrument reproduces this too. The train-to-validation ratios at epoch 300 are 361, 39 and $3.8$ across the same three arms, and Recall@10 falls by $58\%$, $13\%$ and not at all. The ordering of the memorisation signature, not just of the recall, is therefore stable across both datasets we tested.

The masking pattern is resampled each epoch, so a history is not deterministically paired with the same fixed completion target throughout training. That is what prevents the collapse, and it makes the corruption closer to a structured augmentation over the history than to a fixed prediction target. It also means the ablated arm is not undertrained but over-trained, so reporting it at its best validation checkpoint is the generous reading rather than a penalising one.

**What the third arm shows.** Random one holds the number of scored positions fixed at one per window and varies only the location of that position. It recovers $55\%$ of the terminal-to-random-subset Recall@10 gap on Scientific and $57\%$ on Musical Instruments. This establishes that random target location contributes substantially and that its benefit cannot be attributed to additional scored positions. The remaining difference is associated with moving to the full random-subset policy, but that step jointly changes the number and variability of masked positions, loss aggregation, and $1/t$ reweighting. It therefore cannot be attributed specifically to masking several positions or to any one of these changes. We describe the approximate split only as half associated with random target location and half associated with the complete subset policy, not as separable causal effects.

**Relation to masked sequential recommenders.** That cloze-style corruption generalises better than next-item-only training is not itself a new observation; it is the motivation behind masked sequential recommenders such as BERT4Rec, which appears among our baselines. We report this control not as a finding about masking in general but because the claim under test is specific: our corruption is applied to *continuous item latents* rather than to discrete identifiers. Within that setting, the control shows that randomizing the target location contributes substantially and that the complete random-subset policy provides an additional gain. Because the second step also changes loss aggregation and reweighting, the experiment does not identify which component of the complete policy produces that remaining gain. Readers who already accept the masked-training argument may treat this section as a confound check.

**Three qualifications.** First, the training losses of the three arms are not directly comparable in absolute value. The random subset arm aggregates the combined SID and InfoNCE loss over multiple masked positions per window, including interior positions whose prediction context differs from terminal next-item prediction, and applies the $1/t$ reweighting factor. The terminal-only and random-one arms predict a single position and do not use this reweighting. The exact-match rates reported above are computed under the same evaluation protocol and are therefore more directly comparable than the weighted training losses. The strongest evidence, however, is the terminal-only arm's own divergence between training fit and validation recall, which does not depend on a cross-arm comparison.

Second, neither randomly masked arm is trained predominantly on the terminal position in isolation — random one draws that position in roughly one window in eight, and random subset masking seldom isolates it — which is the protocol under which we rescore them. Part of their low exact-match scores may reflect that mismatch rather than an absence of memorization. The within-arm trajectories, measured under each arm's own training conditions, are unaffected by this.

Third, the replication covers two datasets, not four. Scientific and Musical Instruments differ in per-item training coverage ($2.62$ versus $3.52$ training windows per item), but Game and Arts remain untested. The experiment therefore establishes transfer beyond one dataset, but not across the full range of catalog scale or interaction-matrix sparsity. Before running Musical Instruments, we expected the gap to narrow with greater per-item supervision, and it did slightly: terminal-only retains $55.6\%$ of the full recipe's Recall@10 there, compared with $52.6\%$ on Scientific.