THE UPDATE BOTTLENECK: RE-EXAMINING JOINT TRAINING FOR SEMANTIC-ID-BASED GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Semantic ID based Generative Retrieval models map items to discrete Semantic IDs (SIDs) and train a sequence model over the codes. Several recent methods train the tokenizer jointly with the recommender and report 9 to 14% gains over two-stage pipelines. We re-examine this comparison under matched conditions. Joint training improves Recall@10 on Beauty without a statistically significant gain against a two-stage method, while Recall@10 falls by approximately 8–16% on the other four datasets. We propose the *update bottleneck* hypothesis to explain these inconsistent benefits: tokenizer updates may leave discrete IDs unchanged, while frequent ID changes can disrupt the recommender's learning. Our theory formalizes the limits on how tokenizer updates affect a recommender through discrete IDs. Controlled experiments support the update bottleneck and show why gradient flow alone does not ensure useful representation learning in semantic-ID-based generative recommendation.

1 INTRODUCTION

In semantic-ID-based generative recommendation, a tokenizer maps item content to a short sequence of discrete tokens, called a semantic ID (SID). A sequential recommender then learns to generate the next item's SID from a user's past interactions (Rajput et al., 2023). The tokenizer therefore makes a consequential representation-learning choice: which items share tokens and how their IDs differ. These representations should capture relationships between items that help predict user interactions.

A common approach trains the tokenizer first, then trains the recommender using the resulting fixed SIDs (Rajput et al., 2023). However, IDs learned from item content alone may miss relationships useful for recommendation: products with similar descriptions may have different patterns of user interaction. Several methods therefore incorporate collaborative-filtering representations into tokenizer training (Wang et al., 2024; Zhai et al., 2025), while others use item co-occurrence in user histories (Lepage et al., 2025). These approaches allow SIDs to capture interaction patterns before the recommender is trained. Another line of work trains the tokenizer and recommender jointly, allowing signals from the recommender to guide how items are tokenized (Liu et al., 2025; Fu et al., 2026). Their reported improvements support an intuitive expectation: allowing the recommendation objective to guide tokenizer learning should produce item representations better suited to predicting user interactions, and in turn improve generative recommendation.

Jointly training the tokenizer and the recommender is challenging because semantic IDs are discrete – ordinary backpropagation cannot directly differentiate through token selection. Straight-through estimators and continuous relaxations such as Gumbel-Softmax provide approximate gradients for learning discrete representations (Bengio et al., 2013; Jang et al., 2017). However, enabling gradient flow does not ensure stable joint optimization of the tokenizer and recommender, a concern recognized in prior work (Liang et al., 2026; Fu et al., 2026). Moreover, existing training procedures combine recommendation gradients with additional modules, auxiliary losses, and stabilization mechanisms. Improvements from these complete procedures do not isolate the contribution of the recommendation gradient itself. We therefore ask: *how much does allowing recommendation gradients to update the tokenizer improve recommendation accuracy under otherwise matched training conditions?*

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.

To answer these questions, we combine a comparison of joint and two-stage training across five public datasets with controlled experiments that enable or disable recommendation gradients. Across multiple seeds, joint training offers inconsistent benefits: on Beauty, it improves NDCG@10 but provides no statistically significant Recall@10 gain; on the other four datasets, both metrics decline. These findings motivate examining what the recommendation gradient changes and whether those changes help the recommender learn.

We propose the *update bottleneck* hypothesis to explain these inconsistent benefits. It describes a tension: limited reassignment restricts the influence of tokenizer updates through discrete IDs, while frequent reassignment can disrupt the recommender's learning.

We investigate this tension through *assignment dynamics*, the evolution of item-to-SID assignments during training. Our theory establishes that, under a discrete-ID interface, unchanged assignments leave the recommendation loss and its gradient unchanged at fixed recommender parameters, regardless of changes inside the tokenizer. Our controlled experiments provide empirical support for the other aspect of the hypothesis: continued tokenizer updates can disrupt recommendation learning. Together, these findings support the update bottleneck hypothesis and show why gradient flow alone may be insufficient for useful representation learning in semantic-ID-based generative recommendation.

Our contributions are:

- We re-examine joint versus two-stage training across five datasets, distinguishing complete pipeline comparisons from controlled tests of recommendation gradients.
- We formalize how tokenizer updates reach a recommender through discrete IDs and propose the update bottleneck hypothesis to explain inconsistent joint-training gains.
- We analyze assignment dynamics through controlled experiments examining recommendation gradients, embedding displacement, codebook tying, and the timing of tokenizer updates.

2 Related Work

Learning item representations for generative recommendation. TIGER represents items with hierarchical semantic IDs obtained through residual quantization (Lee et al., 2022), then trains an autoregressive recommender on these identifiers (Rajput et al., 2023). This makes item tokenization part of the representation-learning problem: the assigned tokens determine how item information is shared and distinguished during recommendation. LETTER extends this approach by incorporating collaborative alignment and code-assignment diversity into tokenizer learning (Wang et al., 2024). These results establish that the information encoded in item IDs matters, and that recommendation-relevant information can be incorporated before training the generative recommender. Our study examines the additional step of continuing to adapt the tokenizer through recommendation gradients while the recommender learns.

Coupling tokenizer and recommender training. ETEGRec integrates item tokenization and recommendation through sequence–item and preference–semantic alignment, using alternating optimization to stabilize training (Liu et al., 2025). Its ablations associate frequent tokenizer updates with poorer performance and examine retraining a recommender on the final learned IDs. DIGER studies differentiable semantic-ID learning, using Gumbel exploration and uncertainty decay to improve code utilization and convergence (Fu et al., 2026). It also measures per-epoch and cumulative assignment changes and agreement between training-time and inference-time IDs. These studies already identify stability as an important aspect of joint learning. We build on that observation by separating tuned pipeline comparisons from experiments that enable or disable the recommendation gradient, examining their effects across datasets, and relating those effects to changes in item representations. Our experiments evaluate a specified gradient-passing implementation; they do not establish that every form of tokenizer–recommender coupling behaves identically.

Gradient estimation and discrete representations. Vector-quantized representation learning uses learned codebooks to construct discrete latent representations (van den Oord et al., 2017). Straight-through estimators (Bengio et al., 2013) and continuous categorical relaxations (Jang et al., 2017; Maddison et al., 2017) provide ways to train parameters associated with discrete choices. These techniques address how a learning signal can pass backward through an otherwise non-differentiable

operation. Our analysis asks what the resulting updates change for the downstream recommender. The distinction matters because the recommender may learn its own embeddings for discrete tokens or share the tokenizer's codebook vectors as embeddings, an arrangement known as codebook tying. These interfaces expose different aspects of tokenizer learning to the recommender. We investigate when that learning changes the representations used for recommendation and whether those changes improve performance.

3 Problem Setting

Recommendation with semantic IDs. In semantic-ID-based generative recommendation, each item is represented by a short sequence of discrete tokens, called a semantic ID (SID). A tokenizer constructs these IDs from item content, such as a text embedding of the item's title and description. For example, an item might receive the SID $(7, 12, 4)$, where each integer selects an entry from a learned codebook. The recommender represents a user's interaction history using the corresponding SIDs and predicts the next item's SID autoregressively (Rajput et al., 2023).

Formally, let $\mathcal{I}$ denote the item catalog and x_i the content features of item i. A tokenizer with parameters θ assigns the item an SID $z_i = (z_i^1, \ldots, z_i^L)$, where each z_i^ℓ indexes one of K entries in the codebook for level ℓ. The recommender g_ϕ maps the tokens in a user's history to embeddings and uses them to predict the tokens of the next item. Its recommendation loss is denoted by $\mathcal{L}_{\mathrm{rec}}$.

Two-stage and joint training. In two-stage training, the tokenizer is trained first and its item-to-SID assignments are then fixed. The recommender learns from these fixed assignments. This gives it a stable representation of each item, but the assignments cannot adapt to its recommendation objective during training. In contrast, joint training allows the tokenizer to continue learning alongside the recommender, with the recommendation objective also used to update tokenizer parameters. The expectation is that this feedback will produce IDs better suited to predicting user interactions.

Recent methods report gains from coupling tokenizer and recommender training, and show that stabilizing this process is important (Liu et al., 2025; Fu et al., 2026). These findings motivate a closer examination of how recommendation feedback contributes to those gains. We focus on whether allowing the recommendation gradient to update the tokenizer produces changes to item representations that the recommender can use effectively. We begin by examining whether joint training improves over a well-tuned two-stage baseline, then investigate the role of the recommendation gradient by enabling or disabling it within otherwise matched training configurations. We then examine how often items change their semantic IDs during training and whether these changes are associated with better or worse recommendation performance.

4 Experimental Setup

Our experiments examine whether joint training improves recommendation over a strong two-stage baseline and how recommendation gradients contribute to its performance. We first describe the architecture and hyperparameter search conducted on Beauty using two-stage training and the released DIGER implementation (Fu et al., 2026). We then describe our joint-training implementation, which builds on DIGER and verifies that recommendation gradients reach the tokenizer.

Datasets and evaluation. We evaluate on five categories from Amazon Reviews (Ni et al., 2019): Beauty, Musical Instruments, Sports and Outdoors, Industrial and Scientific, and Toys and Games. After 5-core filtering, we order each user's interactions chronologically and reserve the last interaction for testing and the preceding interaction for validation. The remaining interactions form the training set. We retain at most 50 items in each input history and evaluate recommendation quality using Recall@10 and NDCG@10. Dataset statistics and results at cutoff 5 are provided in Appendices B and E.

Establishing a tuned baseline. To give both training regimes a strong operating point, we searched recommender architecture and training hyperparameters for two-stage training and DIGER on Beauty. Both arms used the same released content embeddings and pretrained tokenizer. They received the same search over model width, depth, and recommender learning rate, followed by variations in dropout, weight decay, and learning-rate warmup. DIGER additionally received tuning of its Gumbel-relaxation settings and tokenizer learning rate. The search and six-seed confirmation experiments comprised 140 runs.

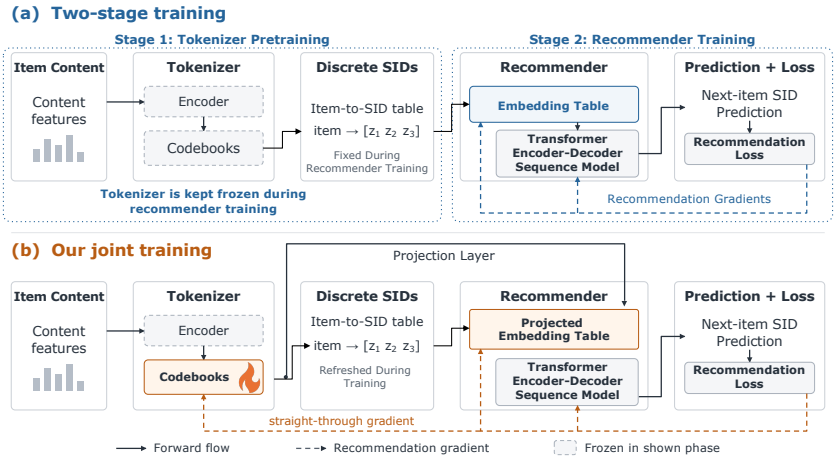


Figure 1: Training configurations used to examine two-stage versus joint training. **(a)** In two-stage training, the tokenizer is trained separately, and item-to-SID assignments remain fixed while the recommender learns its own token embeddings. **(b)** For our investigation of joint training, we modify the released DIGER implementation and verify recommendation gradients flow to the tokenizer's codebook. SID assignments are refreshed during training, and projected codebook vectors serve as semantic-token embeddings. A straight-through estimator passes recommendation gradients to the codebooks, ensuring the recommender informs the tokenizer of the downstream task.

Based on the Beauty search, we adopted a shared recommender architecture with four encoder and four decoder layers, hidden dimension 128, two attention heads, and feed-forward dimension 512. We use this architecture for both two-stage and joint training across all five datasets, keeping model capacity fixed across training regimes without repeating the architecture search for each dataset.

Tokenizer and training configuration. The tokenizer is a three-level RQ-VAE with 256 entries per codebook and codebook-vector dimension 256. A fourth token distinguishes items assigned the same three semantic tokens. The quantizer uses Sinkhorn balanced assignment, adaptive code selection, and a learnable-σ Gumbel relaxation, with shared settings across matched arms. The SIDs used by the recommender are obtained through nearest-neighbor assignment.

Our joint implementation inherits the selected Beauty training recipe rather than receiving a separate search. This recipe uses AdamW, batch size 128, weight decay 0.05, dropout 0.1, and cosine learning-rate decay with 1,500 warmup steps. The recommender learning rate is 10^{-3} and the tokenizer learning rate for joint training is 10^{-5}. In the main Beauty joint runs, the tokenizer's encoder remains frozen. Detailed training settings are provided in Appendix B.

Joint training with a modified DIGER implementation. To study the effect of recommendation gradients on the tokenizer, we modify the DIGER codebase to ensure the following. First, we tie the recommender's input and output embeddings for semantic tokens to the tokenizer's projected codebook vectors. Second, we retain discrete SIDs in the forward pass. Third, we use a straight-through estimator with Gumbel-relaxed assignment probabilities to pass recommendation gradients to the codebooks and their projection. We verify that recommendation gradients reach the tokenizer's codebook parameters; verification details are provided in Appendix C. This forms all subsequent "joint" training runs.

Statistical significance tests. We use six random seeds and report mean performance and standard deviation. Statistical comparisons use mean per-seed differences with 95% paired-t confidence intervals. Additional details are provided in the appendix.

5 Empirical Results

The tuning study described in Section 4 allows us to assess DIGER against a strong two-stage baseline. We first report this comparison on Beauty, then examine the performance of our joint implementation across all five datasets.

Table 1: Two-stage versus joint training. Metrics are Recall@10 / NDCG@10; relative changes refer to Recall@10. Published scores provide context only.

Dataset	Two-stage R@10 / N@10	Joint R@10 / N@10	Relative Change (joint vs disjoint)	Published SOTA
Beauty	0.0716 ± 0.0019 / 0.0394 ± 0.0008	0.0725 ± 0.0007 / 0.0405 ± 0.0004	$+1.2\%$	DIGER 0.0696 / 0.0376
Instruments	0.1128 ± 0.0008 / 0.0835 ± 0.0007	0.1000 ± 0.0029 / 0.0725 ± 0.0039	-11.3%	DIGER 0.1138 / 0.0844
Sports	0.0405 ± 0.0013 / 0.0216 ± 0.0009	0.0342 ± 0.0018 / 0.0179 ± 0.0011	-15.7%	TIGER 0.0419 / 0.0234
Scientific	0.0438 ± 0.0004 / 0.0233 ± 0.0003	0.0404 ± 0.0007 / 0.0214 ± 0.0006	-7.8%	ETEGRec 0.0455 / 0.0241
Toys & Games	0.0639 ± 0.0012 / 0.0334 ± 0.0009	0.0572 ± 0.0014 / 0.0300 ± 0.0006	-10.5%	TIGER 0.0698 / 0.0433

Table 2: Paired per-seed differences, joint minus two-stage, with 95% paired-t confidence intervals and two-sided p-values. Comparisons use six common seeds. Beauty uses the updated comparison against the two-stage baseline in Table 1. Both tables are computed from the same per-seed runs and seed subsets, so any residual discrepancy below 0.0001 between a reported difference and direct subtraction of the displayed means is display rounding (mean of per-seed differences versus difference of rounded means).

Dataset	ΔRecall@10 [95% CI] (p)	ΔNDCG@10 [95% CI] (p)
Beauty	$+0.00089$ [-0.00094, $+0.00271$] (0.267)	$+0.00108$ [$+0.00047$, $+0.00169$] (0.006)
Instruments	-0.0128 [-0.0171, -0.0084] (0.001)	-0.0110 [-0.0163, -0.0057] (0.003)
Sports	-0.0064 [-0.0084, -0.0043] (0.001)	-0.0036 [-0.0050, -0.0023] (0.001)
Scientific	-0.0035 [-0.0040, -0.0029] (< 0.001)	-0.0019 [-0.0025, -0.0013] (0.001)
Toys & Games	-0.0067 [-0.0082, -0.0052] (< 0.001)	-0.0033 [-0.0043, -0.0024] (< 0.001)

Observation 1: Two-stage training provides a strong baseline after tuning We evaluate the publicly released implementation of DIGER, a state-of-the-art joint-training recipe (Fu et al., 2026), on Beauty using the configuration selected in Section 4. Across six seeds, two-stage training achieves Recall@10 of 0.0716 ± 0.0019 and NDCG@10 of 0.0394 ± 0.0008, compared with 0.0703 ± 0.0012 and 0.0385 ± 0.0008, respectively, for DIGER as released. The released recipe therefore does not improve mean accuracy over the tuned two-stage baseline in this comparison.

Observation 2: Gradient-verified joint training does not consistently improve recommendation performance We next evaluate our joint implementation, in which recommendation gradients are verified to reach the tokenizer's codebooks. Table 1 compares its performance with two-stage training across five datasets. Joint training has lower mean Recall@10 and NDCG@10 on four datasets, with relative Recall@10 reductions ranging from 7.8% to 15.7%.

These reductions affect both retrieval and ranking quality. On Sports, Recall@10 falls from 0.0405 to 0.0342, while NDCG@10 falls from 0.0216 to 0.0179. Instruments and Toys and Games show Recall@10 reductions of 11.3% and 10.5%, respectively, accompanied by lower NDCG@10. The published scores in Table 1 provide context; our conclusions are based on comparisons between our experimental runs.

Beauty provides a meaningful exception. Joint training increases mean Recall@10 from 0.0716 to 0.0725 and mean NDCG@10 from 0.0394 to 0.0405. To assess these differences relative to seed variability, Table 2 reports mean per-seed differences and their 95% confidence intervals. The 1.2% Recall@10 improvement is not statistically significant: its interval includes zero. The 2.7% NDCG@10 improvement is statistically significant, with an interval entirely above zero and higher joint-training scores on all six seeds.

The reported paired analyses for the other four datasets favor two-stage training on both metrics, with confidence intervals entirely below zero (Table 2). Thus, the observed pattern includes both a positive ranking result on Beauty and substantial performance reductions elsewhere.

These findings challenge the expectation that allowing recommendation gradients to shape the tokenizer will produce representations that improve recommendation. Even with verified gradient flow, the joint pipeline often performs worse than training the recommender on fixed semantic IDs. Understanding this discrepancy requires examining what the tokenizer's updates actually change for the recommender. We therefore investigate how these updates alter the representations used during recommendation training, and whether changes to semantic IDs help the recommender learn or disrupt what it has already learned.

6 THE UPDATE BOTTLENECK

In this section, we raise a representation-learning question: why does allowing the recommendation objective to update the tokenizer not reliably improve recommendation? To investigate this question, we examine how tokenizer updates affect the item representations used by the recommender. This requires identifying what information passes between the two components, which we call the *interface*: the tokenizer outputs that enter the recommender.

When tokenizer updates become visible to the recommender. Consider an item assigned the SID $(7, 12, 4)$. Suppose a recommendation gradient updates the tokenizer, but the item retains this assignment. If the recommender uses its own embedding table, it still reads the same three token embeddings. Holding the recommender's parameters fixed, the tokenizer update has not changed the item's representation for recommendation. If the assignment instead changes to $(7, 18, 4)$, the recommender selects a different embedding-table entry at the second position. We call a change in an item's SID a *reassignment*, or *flip*.

This distinction separates the backward training signal from its forward effect. A straight-through estimator can provide a nonzero gradient to the tokenizer even when the discrete assignment remains unchanged. The existence of that gradient therefore does not establish that the recommender receives a different representation after the update.

We first formalize this observation for a recommender whose embedding tables are independent of the tokenizer. Here, z_i denotes the complete token sequence used to represent item i, and $z^{(t)} = \{z_i^{(t)}\}_{i \in \mathcal{I}}$ denotes the assignments at step t. These assignments may depend on the catalog as a whole, as under balanced transport; the argument does not require independent assignment of each item. Let $\mathcal{I}_{\mathcal{B}}$ contain all items appearing as inputs or targets in a fixed minibatch $\mathcal{B}$. We write $\mathcal{L}_{\text{rec}}(\phi; z^{(t)})$ for the recommendation loss, with the minibatch and the recommender's stochastic choices held fixed.

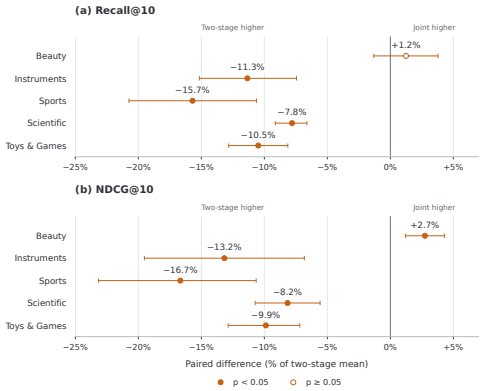


Figure 2: Joint versus two-stage training. Points show pairwise mean differences vs two-stage; bars show 95% paired-t confidence intervals. Positive values favor joint training. Filled circles indicate statistically significant difference.

Lemma 1 (Interface invariance)**.** *Suppose the recommender accesses the tokenizer only through discrete SIDs, using its own embedding tables independent of the tokenizer parameters. If*

$$z_i^{(t+1)} = z_i^{(t)} \qquad \textit{for every } i \in \mathcal{I}_{\mathcal{B}},$$

then, at any fixed recommender parameters ϕ,

$$\mathcal{L}_{\text{rec}}(\phi; z^{(t+1)}) = \mathcal{L}_{\text{rec}}(\phi; z^{(t)}), \tag{1}$$

$$\nabla_\phi \mathcal{L}_{\text{rec}}(\phi; z^{(t+1)}) = \nabla_\phi \mathcal{L}_{\text{rec}}(\phi; z^{(t)}), \tag{2}$$

regardless of the change in tokenizer parameters.

Proof. Unchanged assignments preserve every input token sequence and prediction target in $\mathcal{B}$. Since no other tokenizer output enters the recommender, the loss is the same function of ϕ before and after the tokenizer update. Its value and gradient with respect to ϕ are therefore unchanged. □

The lemma identifies a constraint on how tokenizer learning becomes visible to the recommender. A surrogate gradient may update the tokenizer even when no item changes its SID, but such an update leaves the recommendation loss and the recommender's training gradient unchanged at fixed ϕ. Under this discrete-ID interface, changes inside the tokenizer are invisible to the recommender unless they change the assigned IDs.

Quantifying the effect of reassignment. Reassignment can make a tokenizer update visible to the recommender, but the size of that change depends on which items are reassigned and how their embeddings differ. An item's *exposure* is its relative frequency in the recommender's training inputs: reassigning a frequently encountered item affects more input occurrences than reassigning a rare

(a) Beauty

Codebook Vectors	Recommendation Gradient	Recall@10	NDCG@10	$\overline{M}$
Frozen	On	0.0607 ± 0.0021	0.0337 ± 0.0012	n/m†
	Off (two-stage)	0.0696 ± 0.0018	0.0387 ± 0.0009	0.0000
Moving	On (joint)	0.0732 ± 0.0018	0.0410 ± 0.0007	0.0034
	Off	0.0693 ± 0.0021	0.0381 ± 0.0014	0.0094

(b) Toys and Games

Codebook Vectors	Recommendation Gradient	Recall@10	NDCG@10	$\overline{M}$
Frozen	On	0.0290 ± 0.0018	0.0160 ± 0.0011	n/m†
	Off (two-stage)	0.0645 ± 0.0019	0.0339 ± 0.0007	0.0000
Moving	On (joint)	0.0574 ± 0.0013	0.0301 ± 0.0006	0.0411
	Off	0.0603 ± 0.0010	0.0313 ± 0.0004	0.0276

Table 3: Controlled experiments on Beauty and Toys and Games. Accuracy is mean $\pm$ standard deviation over six seeds. $\overline{M}$ is exposure-weighted reassignment mass averaged over training epochs, then over diagnostic seeds: six for Beauty and two for Toys, using the four-layer backbone. †The diagnostics do not provide a comparable measurement for frozen/on because the deployed SID table is static; n/m does not imply zero online reassignment.

item. Even when an item receives a different SID, however, the new SID may select embeddings close to the old ones. We call the distance between these representations the *embedding displacement*. Together, item exposure and embedding displacement account for how often the recommender encounters a reassignment and how much that reassignment changes the representation it sees.

We formalize these quantities to relate reassignment to changes in the recommendation loss. Let $z_i^{(t)}$ and $z_i^{(t+1)}$ be item i's assignments before and after a tokenizer update. Let $E_\phi[z]$ denote the sequence of input embeddings representing SID z. We measure the distance between two such sequences by summing the Euclidean distances between their embeddings at corresponding token positions.

For a fixed collection of training examples, let n_i count the occurrences of item i in the recommender's inputs, including history and teacher-forced decoder inputs. Let $N_{\text{in}} = \sum_i n_i$ and define $w_i = n_i / N_{\text{in}}$. The *exposure-weighted reassignment mass* is

$$M_t = \sum_i w_i \, \mathbf{1}\left[z_i^{(t+1)} \neq z_i^{(t)}\right]. \tag{3}$$

Thus, M_t measures the fraction of item occurrences in the recommender's inputs affected by reassignment, rather than the fraction of distinct catalog items that change IDs.

To also account for the size of each representation change, define the *exposure-weighted embedding displacement*

$$D_t = \sum_i w_i \left\| E_\phi[z_i^{(t+1)}] - E_\phi[z_i^{(t)}] \right\|. \tag{4}$$

Unlike M_t, this quantity distinguishes a reassignment to nearby embeddings from one that substantially changes the item's representation. The diagnostic logs use occurrence shares in training sequences as a proxy for the exact input weights w_i.

Reassignment also changes what the recommender must predict. If a target item's SID changes from $(7, 12, 4)$ to $(7, 18, 4)$, the correct label at the second position changes from 12 to 18. We therefore account for both changes to the recommender's inputs and changes to its prediction targets.

Let $O_\phi[k]$ denote the output-classifier vector for token k, and let $O_\phi[z]$ collect these vectors for SID z, using the same sum-over-positions distance. Let m_i count occurrences of item i as a prediction target. Hold the recommender parameters, embedding tables, training examples, and stochastic choices fixed. Let $\Delta_{\text{reassign}}\mathcal{L}_{\text{rec}}$ denote the loss change caused by replacing the old SIDs with the new SIDs in both inputs and targets.

For the bound, take $\mathcal{L}_{\text{rec}}$ to be summed token cross-entropy; an averaged loss introduces the corresponding normalization. Assume that changing one input token embedding by distance δ changes the loss by at most $C_L\delta$ with labels fixed. Assume also dot-product logits $\langle h, O_\phi[k]\rangle$ over a fixed vocabulary at each position, with $\|h\| \leq H$. Then

$$\begin{aligned} |\Delta_{\text{reassign}}\mathcal{L}_{\text{rec}}| \leq C_L \sum_i n_i \left\| E_\phi[z_i^{(t+1)}] - E_\phi[z_i^{(t)}] \right\| \\ + H \sum_i m_i \left\| O_\phi[z_i^{(t+1)}] - O_\phi[z_i^{(t)}] \right\|. \end{aligned} \tag{5}$$

To obtain the bound, first change the inputs while holding labels fixed, then change the labels with the new inputs fixed. The first step is bounded by the input Lipschitz assumption. For the second,

changing a label from y to y' changes its cross-entropy by $\langle h, O_\phi[y] - O_\phi[y'] \rangle$: the softmax normalization cancels. Its magnitude is at most $H\|O_\phi[y] - O_\phi[y']\|$. Summing over input occurrences and target positions gives Equation 5. We give the complete argument in Appendix F.

The input term is $N_{\text{in}} C_L D_t$. The target term is $N_{\text{out}} H D_t^{\text{out}}$, where $N_{\text{out}} = \sum_i m_i$ and D_t^{out} replaces E_ϕ and w_i in Equation 4 with O_ϕ and m_i / N_{out}. If each complete SID embedding has norm at most B, then $D_t \leq 2BM_t$; the analogous relation holds for the target-side quantities.

The bound shows that small exposure-weighted displacement on both sides limits the immediate change in recommendation loss caused by reassignment. A nonzero recommendation gradient can therefore update the tokenizer while producing little change in what the recommender reads or predicts. Large displacement permits a larger effect, but the bound does not determine whether that effect improves or disrupts learning.

How tied embeddings change the interface. The preceding analysis holds the recommender's embedding tables fixed while assignments change. In our tied implementation, tokenizer updates can also change the codebook vectors used by the recommender, even when SIDs remain unchanged. This provides an additional continuous pathway that the reassignment bound does not cover. Consequently, low reassignment alone does not imply that tokenizer updates have little overall influence. Our empirical analysis distinguishes changes to assignments from this additional pathway.

We use the term *assignment dynamics* to describe how item-to-SID assignments evolve during training. We characterize these dynamics through reassignment rates, exposure-weighted reassignment mass, and the embedding displacement caused by reassignment.

Hypothesis 1 (The update bottleneck)**.** *Under a discrete-ID interface, small exposure-weighted displacement in both inputs and targets limits the immediate influence of tokenizer updates. Frequent reassignment, however, may disrupt learning by repeatedly updating the representations and prediction targets the recommender must learn from—a* moving-target problem *(Mnih et al., 2015; He et al., 2020). We hypothesize that this tension helps explain why joint training does not consistently outperform two-stage training.*

Frequent reassignment can still produce small embedding displacement, so reassignment mass alone does not establish harm. To examine the hypothesis, we next test how recommendation gradients affect assignment dynamics and recommendation accuracy.

7 Testing the Update Bottleneck Hypothesis

We now examine whether assignment dynamics help explain the inconsistent benefits of joint training. We first isolate the effect of recommendation gradients, then measure which items are reassigned and how their representations change. Finally, we test whether delaying tokenizer updates improves performance.

How does the recommendation gradient affect assignment dynamics? We compare four configurations that vary whether codebook vectors can move and whether recommendation gradients reach the tokenizer (Table 3). In frozen/on, codebook vectors remain fixed, but encoder updates can still change item-to-SID assignments. These controlled runs use a different training budget and a shared configuration across all rows in the Table, so their absolute scores need not match the tuned pipeline results in Table 1. With moving codebooks, recommendation gradients have opposite effects on Beauty and Toys and Games. On Beauty, enabling the gradient reduces mean exposure-weighted reassignment mass from 0.0094 to 0.0034 while increasing NDCG@10 from 0.0381 to 0.0410. On Toys and Games, enabling the gradient increases reassignment mass from 0.0276 to 0.0411 while reducing NDCG@10 from 0.0313 to 0.0301. Thus, although the gradient helps on one dataset and hurts on the other, both comparisons show the same relationship between reassignment and accuracy. **Key Takeaway:** Within each moving-codebook comparison, lower exposure-weighted reassignment mass is associated with higher accuracy.

Does this reassignment pattern extend across datasets? The association between extensive reassignment and poorer joint-training performance extends across the five main datasets (Table 4). Exposure weighting preserves this pattern, although it does not explain differences in the size of the performance loss. **Key takeaway:** Greater exposure-weighted reassignment accompanies poorer

Table 4: Reassignment across the five datasets. The reassignment columns report the fraction of items whose final SID differs from its post-warmup SID, unweighted and weighted by training exposure (five-seed means). This measure does not count repeated reassignments. NDCG@10 means are reproduced from Table 1. Beauty has the lowest exposure-weighted reassignment (20.8%) and a joint-training gain; the other four datasets have substantially greater reassignment (69.6% to at least 94.3%) and lower joint-training accuracy. The relationship does not determine loss magnitude: Scientific has the greatest reassignment but a smaller relative NDCG@10 loss than Sports.

Dataset	Changed SIDs (%)	Exposure-weighted reassignment (%)	NDCG@10 two-stage / joint
Beauty	20.5	20.8	0.0394 / 0.0405
Instruments	64.6	69.6	0.0835 / 0.0725
Toys & Games	71.4	72.9	0.0334 / 0.0300
Sports	80.0	82.6	0.0216 / 0.0179
Scientific	≥ 92.6	≥ 94.3	0.0233 / 0.0214

Table 5: Beauty representation changes (two diagnostic seeds).

Arm	Displacement $\sum_t D_t$	ARI	NMI	Neighborhood retention
Moving/on	1.52	0.851	0.932	0.959
Moving/off	3.21	0.597	0.796	0.853

Table 6: Delayed tokenizer updates on Toys and Games.

Delay (epochs)	Moving/off NDCG@10	Joint NDCG@10
0	0.0312	0.0301 ± 0.0007
25	0.0321	0.0302 ± 0.0008
50	0.0324	0.0307 ± 0.0006
100	0.0319	0.0309 ± 0.0005

Two-stage: 0.0339 ± 0.0007.

joint-training outcomes across these datasets, but its magnitude alone does not explain how much accuracy is lost.

How much do the representations change? We next measure the exposure-weighted embedding displacement introduced in Section 6. On Beauty, cumulative displacement is 1.52 with the recommendation gradient and 3.21 without it (Table 5). We also compare the item groupings induced by SIDs at warmup and the selected checkpoint, using adjusted Rand index (ARI), normalized mutual information (NMI), and retention of neighbors sharing SID prefixes. All three measures show greater preservation of item groupings with the gradient enabled. **Key Takeaway:** The better-performing joint training run on Beauty has lower cumulative embedding displacement and better-preserved item groupings.

Can delaying tokenizer updates recover two-stage performance? On Toys, we train the recommender with fixed SIDs before allowing tokenizer updates after 0, 25, 50, or 100 epochs. This changes when reassignments can begin. We test both joint training and moving/off. Delaying updates increases joint training's mean NDCG@10 from 0.0301 to 0.0309. Moving/off reaches its highest mean, 0.0324, after a 50-epoch delay. Every tested configuration is below two-stage (0.0339) (Table 6). **Key Takeaway:** Delaying reassignment partially recovers the performance gap on Toys, but none of the tested schedules makes continued tokenizer updates outperform fixed SIDs.

What if we did not tie the codebook to the recommender's embeddings? We compare tied and untied joint training in a separate six-seed Beauty experiment, retaining recommendation gradients to the tokenizer in both configurations. These runs are separate from Table 3a. Tied joint training reaches NDCG@10 of 0.0409, compared with 0.0379 for two-stage training. Its paired improvement of 0.0031 is statistically significant. Untied joint training – where the only way for the recommender gradient to reach the tokenizer is through the recommendation loss updating the codebook – reaches 0.0379. Its difference from two-stage training is not statistically significant. **Key Takeaway:** Tied joint training significantly improves NDCG@10 over two-stage training on Beauty in this comparison. Passing recommendation gradients without codebook tying provides no statistically detectable improvement.

What does testing the hypothesis establish? Our experiments provide qualified support for the update bottleneck hypothesis. In both controlled comparisons, lower exposure-weighted reassignment mass accompanies higher accuracy. Beauty's improvement also accompanies lower embedding displacement and better-preserved item groupings. On Toys, delaying tokenizer updates partially restores accuracy, providing further evidence that early tokenizer updates can interfere with the recommender's learning. These findings support disruption from changing item representations as an explanation for joint training's inconsistent benefits. The tied-embedding comparison also shows that the interface matters: a statistically detectable Beauty gain appears with tying but is not reproduced by passing gradients through discrete assignments alone.

The central lesson of our work is that end-to-end gradient flow does not guarantee better item representations for generative recommenders. Its value depends on what updates reach the recommender and whether those updates help or disrupt its learning. These findings motivate evaluating when stronger two-stage tuning or additional signals during SID pretraining offer more benefit than joint training, while overcoming the update bottleneck remains a challenge for future research. We present limitations and other details in the Appendix.

Reproducibility statement

Appendices B–F describe our data preparation, model and training settings, gradient-flow checks, statistical analyses, and supporting proofs. We will open-source all code after publication, including the code used for training, evaluation, and the reported analyses.

Use of AI tools

Assistive AI tools (e.g., GitHub Copilot) were used for conceptual framing and hypothesis refinement; for feedback on experimental design and pre-specification, and for drafting, revising, and critiquing the paper. AI tools were also used to review and test our code (besides human reviewers). Under ICLR 2027 policy, these tools were assistive and are not authors; they did not originate the paper's scientific claims. All model-assisted passages were reviewed and edited by the authors; all citations were verified against sources; and all code and numerical results were produced and validated by the authors. No personal or confidential data was provided to the AI tools.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. URL https://arxiv.org/abs/1308.3432.

Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M. Jose, and Zhaochun Ren. Differentiable semantic id for generative recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 369–379, 2026. doi: 10.1145/3805712.3809641.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1611.01144. arXiv:1611.01144.

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

Simon Lepage, Jeremie Mary, and David Picard. Closing the performance gap in generative recommenders with collaborative tokenization and efficient modeling. *arXiv preprint arXiv:2508.14910*, 2025. URL https://arxiv.org/abs/2508.14910.

Yu Liang, Zhongjin Zhang, Yuxuan Zhu, Kerui Zhang, Zhiluohan Guo, Wenhang Zhou, Zonqi Yang, Kangle Wu, Yabo Ni, Anxiang Zeng, Cong Fu, Jianxin Wang, and Jiazhi Xia. Rethinking generative recommender tokenizer: Recsys-native encoding and semantic quantization beyond LLMs. *arXiv preprint arXiv:2602.02338*, 2026. URL https://arxiv.org/abs/2602.02338.

Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. Generative recommender with end-to-end learnable item tokenization. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 729–739, 2025. doi: 10.1145/3726302.3729989.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1611.00712. arXiv:1611.00712.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. doi: 10.1038/nature14236.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.05065. arXiv:2305.05065.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1711.00937. arXiv:1711.00937.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2024. URL https://arxiv.org/abs/2405.07314. arXiv:2405.07314.

Penglong Zhai et al. A simple contrastive framework of item tokenization for generative recommendation. *arXiv preprint arXiv:2506.16683*, 2025. URL https://arxiv.org/abs/2506.16683.

Table 7: Dataset sizes after 5-core filtering. The final column gives the number of training subsequences.

Dataset	Users	Items	Interactions	Subsequences
Beauty	22,363	12,101	198,502	131,413
Instruments	24,772	9,922	206,153	131,837
Sports	35,598	18,357	296,337	189,543
Scientific	50,985	25,848	409,394	259,992
Toys and Games	19,412	11,924	167,597	109,361

A Limitations

Our experiments do not establish insufficient reassignment as the cause of missed gains. Nevertheless, the theory that we present shows that tokenizer updates are invisible under unchanged SIDs at a discrete-ID interface, and bounds their immediate influence when assignments change. This provides a concrete reason to evaluate whether continued tokenizer updates offer value beyond a well-tuned two-stage pipeline. The association between lower reassignment and higher accuracy does not establish that reducing reassignment alone would improve recommender model learning. However, this pattern appears in both controlled moving-codebook comparisons and is consistent with the moving-target explanation, providing an empirical basis for further investigation. Our joint implementation ties recommender embeddings to the tokenizer's codebook, so the discrete-ID bound captures only the reassignment component of its updates. The untied comparison and completed displacement diagnostics are limited to Beauty. These controls nevertheless help distinguish the roles of assignment changes and codebook tying. Our evaluation covers five academic benchmarks with catalogs below 30,000 items, leaving generalization to larger production catalogs untested. However, these datasets are established benchmarks in semantic-ID recommendation, providing a common basis for comparison with prior work.

B Data and Training Details

Dataset Selection and Evaluation Protocol We evaluate on these five Amazon review categories because they constitute the canonical evaluation setting for Semantic-ID-based generative recommendation. They are used in prior work including TIGER (Rajput et al., 2023), LETTER (Wang et al., 2024), ETEGRec (Liu et al., 2025), and DIGER (Fu et al., 2026), and are necessary to ensure reproducibility, alignment with established experimental protocols, and direct comparability with previously published results. Using the same datasets, splits, and evaluation metrics enables reviewers to verify our claims by comparing against reported baselines rather than re-implementations. The datasets are used solely for research benchmarking and evaluation; no models, derived datasets, or artifacts incorporating this data are released beyond aggregate experimental results, consistent with prevailing academic practice, and we cite the dataset sources as their distributors request.

Data preparation. We apply 5-core filtering to the five Amazon review categories in Table 7. For each user, interactions are ordered chronologically: the last interaction is used for testing, the preceding interaction for validation, and the remaining interactions for training. Each user contributes one validation and one test instance. Input histories contain at most 50 items.

Tokenizer and recommender. The tokenizer uses frozen content embeddings: 4,096-dimensional LLaMA embeddings for Beauty and 1,024-dimensional Qwen embeddings for the other four datasets. The RQ-VAE maps these features to a 256-dimensional latent space and uses three codebooks, each with 256 entries. A fourth token distinguishes items with identical three-token semantic codes. The recommender is a T5-style encoder–decoder Transformer with four encoder layers, four decoder layers, hidden dimension 128, two attention heads, and feed-forward dimension 512. This architecture is shared across the five datasets.

Training settings. Table 8 gives the selected Beauty training recipe inherited by our joint implementation. The main Beauty joint runs keep the tokenizer encoder frozen while updating the

Table 8: Selected Beauty training settings. These are adopted values, not the ranges explored during tuning.

Setting	Value
Optimizer	AdamW
Batch size	128
Recommender learning rate	10^{-3}
Joint tokenizer learning rate	10^{-5}
Dropout	0.1
Weight decay	0.05
Learning-rate schedule	Cosine decay
Learning-rate warmup	1,500 steps

codebooks and their projection through the recommendation objective. Quantizer settings shared by matched arms are usage momentum 0.99, hot-code threshold ratio 1.5, and a learnable-σ Gumbel relaxation with initial $\sigma = 1$, σ learning rate 10^{-3}, regularization weight 2.0, and temperature $\tau = 2$.

Tuning and comparisons. On Beauty, two-stage training and released DIGER (Fu et al., 2026) use the same content embeddings and pretrained tokenizer. Both receive the same search over model width, depth, and recommender learning rate, followed by variations in dropout, weight decay, and learning-rate warmup. DIGER additionally receives tuning of its Gumbel settings and tokenizer learning rate. The search and six-seed confirmation experiments comprise 140 runs in total. Our modified joint implementation inherits the selected recipe; it does not receive a separate search. The architecture is then fixed across datasets. Both arms receive the same learning-rate grid on the remaining datasets. The controlled factorials use a shared configuration and a different training budget from the main pipeline comparison. Their scores therefore represent separate comparisons.

C Implementation and Gradient Verification

Released DIGER and our joint implementation. The audit of `junchen-fu/DIGER`, commit `fccf122`, finds that the straight-through indicator returned by `VectorQuantizer.forward` is discarded in `trainer.py`. The recommender consumes detached SIDs through a separate embedding table. In this path, the recommendation loss is disconnected from the RQ-VAE codebooks and encoder, although tokenizer self-losses can still update the tokenizer. The released-DIGER comparison in Section 5 is therefore distinct from the subsequent gradient-verified joint experiments.

Our implementation retains hard assignments in the forward pass and uses the straight-through indicator

$$a_{\mathrm{ST}} = a_{\mathrm{hard}} - \mathrm{stopgrad}(a_{\mathrm{soft}}) + a_{\mathrm{soft}}$$

in the backward path. Projected codebook vectors supply the recommender's semantic-token input, decoder, and output embeddings through `code_embed_tables`. The corresponding configuration uses `sid_grad_mode=soft_embed` and `use_indicator_ste`.

Connectivity check. We use `torch.autograd.grad` on $\mathcal{L}_{\mathrm{rec}}$ to test its connection to tokenizer parameters. This isolates the recommendation objective from reconstruction and other tokenizer losses. The released path has no connection; the modified path produces nonzero codebook gradients. The supporting verification script, `verify_fix.py`, also checks encoder gradients when the encoder is trainable. This does not imply encoder updates in the main Beauty runs, where it is frozen. Nonzero gradients establish connectivity, not improved accuracy or a change in discrete SIDs.

Controlled arms. Frozen/off uses fixed SIDs and the recommender's own embedding table. In the factorials, moving/off continues tokenizer updates without recommendation gradients, while moving/on enables them. Frozen/on holds codebook vectors fixed using `freeze_rq`, but its trainable encoder can change assignments through the straight-through path. Thus, frozen codebook vectors

Table 9: Seed counts for supporting experiments. Accuracy and diagnostic measurements need not use the same number of seeds.

Measurement	Dataset or arm	Seeds
Factorial accuracy	Beauty and Toys, each	6
Mean reassignment mass	Beauty	6
Mean reassignment mass	Toys (2021, 2022)	2
Endpoint reassignment	Each of the five datasets	5
Displacement and structure	Beauty	2
Tied/untied comparison	Beauty	6
Delayed updates	Toys, joint	6
Delayed updates	Toys, moving/off	5

do not imply fixed online SIDs. Matched arms share their per-seed initialization, data order, tokenizer self-losses, and training schedule. The separate untied comparison retains the recommender's own embedding table and passes recommendation gradients through the assignment indicator, removing the direct codebook-vector path.

C.1 Assignment and export

Inside RQ-VAE training, the quantizer uses Sinkhorn balanced assignment with per-level $\varepsilon = 0.003$, 50 iterations, unit row marginals, and equal column quotas. The exported SIDs consumed by the recommender use nearest-neighbor assignment through `get_indices`. These are different assignment maps: balanced quantizer assignments should not be identified with the exported SID table. In particular, a static deployed table does not establish that online assignments are unchanged.

D Statistics and Diagnostic Aggregation

Seed pairing and intervals. The seed pool is $\{2020, 2021, 2022, 2023, 2024, 2025\}$. Reported accuracy summaries are means and standard deviations over seeds. For each comparison, we pair runs by seed within the same experiment. Let d_s be joint minus two-stage performance for common seed s, and let n be the number of common seeds. With sample standard deviation s_d, the paired interval is

$$\bar{d} \pm t_{n-1,0.975} \frac{s_d}{\sqrt{n}}, \qquad \bar{d} = \frac{1}{n} \sum_{s=1}^{n} d_s.$$

Two-sided paired-t tests use $t = \bar{d}/(s_d/\sqrt{n})$. The standard deviation of an arm's scores is distinct from this confidence interval for the paired difference. Table 2 uses the main pipeline comparison; the factorials and untied experiment have their own baselines and are not pooled with it. Published scores provide context only. Table 9 distinguishes the seed counts for supporting experiments and diagnostics.

Reassignment measures. For diagnostic seed s, let $M_{s,t}$ be the exposure-weighted reassignment mass between consecutive recorded epochs. The factorial summary first averages within each seed and then across diagnostic seeds:

$$\overline{M} = \frac{1}{S} \sum_{s=1}^{S} \frac{1}{T_s} \sum_{t=1}^{T_s} M_{s,t},$$

where T_s is the number of recorded epoch transitions. Exposure weights are occurrence shares in training sequences, used as a proxy for the exact input-occurrence weights in Section 6. Accuracy uses six seeds in both factorials. Reassignment diagnostics use six seeds for Beauty and two for Toys (2021 and 2022), all with the four-layer backbone. The frozen/on entry is not measurable with the deployed-table diagnostic; it is not a zero-reassignment result.

The endpoint measure in Table 4 instead compares each item's final SID with its post-warmup SID, with and without exposure weighting, and averages over five seeds. It does not count repeated changes or changes that later reverse. The Scientific values remain lower bounds, as reported in that table.

Table 10: Published cutoff-5 reference results corresponding to the published cutoff-10 entries in Table 1.

Dataset	Published model	Recall@5	NDCG@5
Beauty	DIGER (SDUD+FrqUD)	0.0439	0.0293
Instruments	DIGER (FrqUD)	0.0915	0.0772
Sports	TIGER	0.0278	0.0189
Scientific	ETEGRec	0.0294	0.0190
Toys and Games	TIGER	0.0518	0.0375

Table 11: Factorial accuracy at cutoff 5, mean $\pm$ standard deviation over six seeds.

Dataset	Arm	Recall@5	NDCG@5
Beauty	Moving/on	0.0489 ± 0.0009	0.0332 ± 0.0008
	Frozen/off	0.0456 ± 0.0010	0.0310 ± 0.0008
	Moving/off	0.0452 ± 0.0024	0.0304 ± 0.0015
	Frozen/on	0.0399 ± 0.0019	0.0271 ± 0.0012
Toys and Games	Moving/on	0.0359 ± 0.0009	0.0232 ± 0.0006
	Frozen/off	0.0407 ± 0.0012	0.0263 ± 0.0005
	Moving/off	0.0370 ± 0.0011	0.0238 ± 0.0007
	Frozen/on	0.0190 ± 0.0017	0.0128 ± 0.0013

Representation diagnostics. Table 5 uses two diagnostic seeds. It reports cumulative displacement $\sum_t D_t$, together with ARI, NMI, and retention of neighbors sharing SID prefixes between the post-warmup and later representations. Moving/on has displacement 1.52, compared with 3.21 for moving/off, and higher ARI, NMI, and neighborhood retention. These measures describe different aspects of representation change; neither the endpoint reassignment rate nor $\overline{M}$ equals cumulative displacement.

E Supporting Experiments

Published reference results at cutoff 5. Table 10 gives cutoff-5 scores for the published configurations cited in Table 1. The DIGER values come from Table 2 of Fu et al. (2026); the TIGER values are the three-run means in Table 9 of Rajput et al. (2023); and the ETEGRec values come from Table 3 of Liu et al. (2025). These results follow each source's experimental protocol and provide context for our comparisons.

Cutoff-5 metrics for the controlled comparisons. Table 11 reports cutoff-5 results for the factorials in Table 3. These belong to the controlled experiments, with their own training configuration and baselines.

Untied comparison. The separate six-seed Beauty comparison gives NDCG@10 of 0.0409 for tied joint training and 0.0379 for both untied joint and two-stage training. The paired tied-minus-two-stage difference is $+0.0031$, with 95% interval $[+0.0018, +0.0044]$. The untied-minus-two-stage interval is $[-0.0012, +0.0013]$. The latter includes zero.

Delayed updates. Table 6 delays tokenizer updates by 0, 25, 50, or 100 epochs while initially training the recommender on fixed SIDs. Joint results use six seeds and moving/off results use five. The largest mean NDCG@10 is 0.0309 for joint training and 0.0324 for moving/off. Both means remain below the main two-stage reference, 0.0339 ± 0.0007.

F Proof Details

Interface invariance. For Lemma 1, fix the examples, recommender parameters, and stochastic choices. Unchanged SIDs for all input and target items preserve the encoder inputs, teacher-forced

decoder inputs, and labels. With embedding tables independent of the tokenizer, the loss is consequently the same function of ϕ, so its value and gradient with respect to ϕ agree. Catalog-level coupling in the assignment rule does not affect this argument.

Proposition 1 (Input and target effects of reassignment)**.** *Under the assumptions of Equation 5, write U, U' for the old and new input embeddings and Y, Y' for the old and new labels. Define*

$$d_i^{\text{in}} = \sum_{\ell} \|E_\phi[z_i'^{\ell}] - E_\phi[z_i^{\ell}]\|_2,$$
$$d_i^{\text{out}} = \sum_{\ell} \|O_\phi[z_i'^{\ell}] - O_\phi[z_i^{\ell}]\|_2.$$

Then the summed token cross-entropy satisfies

$$|\mathcal{L}(U', Y') - \mathcal{L}(U, Y)| \le C_L \sum_i n_i d_i^{\text{in}} + H \sum_i m_i d_i^{\text{out}}.$$

Proof. Insert $\mathcal{L}(U', Y)$ and apply the triangle inequality. For $|\mathcal{L}(U', Y) - \mathcal{L}(U, Y)|$, replace the input embeddings one position at a time. The coordinatewise Lipschitz assumption bounds the sum of these changes by $C_L \sum_i n_i d_i^{\text{in}}$.

For $|\mathcal{L}(U', Y') - \mathcal{L}(U', Y)|$, hold all inputs, including teacher-forced decoder inputs, fixed at U'. At each output position the hidden state h is therefore fixed. With dot-product logits over the same vocabulary,

$$\ell(h, y') - \ell(h, y) = \langle h, O_\phi[y] - O_\phi[y'] \rangle,$$

because the log-softmax normalization cancels. Cauchy–Schwarz and $\|h\|_2 \le H$ give $|\ell(h, y') - \ell(h, y)| \le H\|O_\phi[y] - O_\phi[y']\|_2$. Summing over target occurrences and token positions proves the second term and hence the result. □

Normalization and scope. The SID norm here is the sum of token-vector Euclidean norms, including every token used by the consumer. If this norm is at most B, the triangle inequality gives $d_i^{\text{in}} \le 2B\mathbf{1}[z_i' \ne z_i]$ and hence $D_t \le 2BM_t$. The same argument applies to output embeddings with their own norm bound and target-occurrence weights. An averaged token loss divides both sides of the bound by the fixed number of loss terms. The bound concerns reassignment at fixed embedding tables. It does not bound continuous changes to tied codebook vectors, long-term training outcomes, or the sign of an accuracy change.