RELAX: SEMANTIC IDS NEED NOT IDENTIFY A SINGLE ITEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation with Semantic IDs commonly uses generation targets that uniquely identify the target item. We identify an *exact item generation bottleneck*: even when the correct base ID remains in the beam, exact completion can still lose the target. Conditioned on base ID success, exact completion rates are 32.25–45.08 percentage points lower for multi-item base IDs than for single-item base IDs across four benchmarks. We introduce RELAX, which relaxes exact item generation through Shared IDs with an explicit capacity bound. RELAX constructs a capacity-constrained Semantic ID hierarchy and assigns each item the shortest prefix shared by at most C items as its generation target. At inference, each generated Shared ID expands to its item bucket, and a continuous scorer ranks the union of these buckets. The generation target therefore specifies a bounded item set rather than requiring unique item identification. Relative to the strongest reproduced baseline for each metric and benchmark, RELAX improves Recall@10 and NDCG@10 by 11.4% and 19.7% on average, respectively. In controlled comparisons using the same frozen continuous scorer, RELAX outperforms the variants that retain exact item generation.

1 INTRODUCTION

Generative recommendation casts next-item prediction as sequence generation (Geng et al., 2022; Hua et al., 2023; Rajput et al., 2023). An item identifier determines both how an item is represented and what the generator must predict. TIGER constructs Semantic IDs through residual quantization (Rajput et al., 2023), and subsequent work studies the signals, objectives, and token structures used to learn identifiers (Wang et al., 2024; Zheng et al., 2024; Ju et al., 2025; Lin et al., 2025; Fu et al., 2026). Generation is also combined with dense retrieval in LIGER and with item scoring in Gryphon (Yang et al., 2025a; Tikhonovich et al., 2026). We focus on whether generation must resolve exact item identity before final ranking. When ranking is restricted to generated candidates, requiring unique item identification can exclude a target before the scorer has an opportunity to rank it.

We diagnose this failure using *Unique*, the controlled configuration trained to generate unique Semantic IDs (Appendix A.5). Following TIGER (Rajput et al., 2023), a disambiguation token distinguishes items assigned the same quantization codes, which we call the *base ID*. A single-item base ID already identifies one catalog item, whereas a multi-item base ID requires the final token to distinguish among its associated items. The generator predicts the final token for both groups. Within the same decoding run, *base ID success* means that the correct base ID remains in the beam, and *exact completion* means that the complete unique ID of the target remains after the final token. We refer to target loss between these stages as the *exact item generation bottleneck*.

At the default beam width $b = 20$, exact completion conditioned on base ID success reaches 97.00–99.68% for single-item base IDs and 54.60–65.40% for multi-item base IDs across the four benchmarks, a gap of 32.25–45.08 percentage points (Figure 1). In the two preceding base ID transitions, multi-item base IDs have higher conditional transition rates, but the ordering reverses at exact completion, after the correct base ID has already been retained. The conditional completion gap remains positive on every benchmark across all tested beam widths. Appendix B reports the complete counts and analyses across decoding stages and beam widths. The reproduced TIGER baseline also exhibits a positive conditional completion gap on all four benchmarks at $b = 20$ (Appendix B.5).

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.

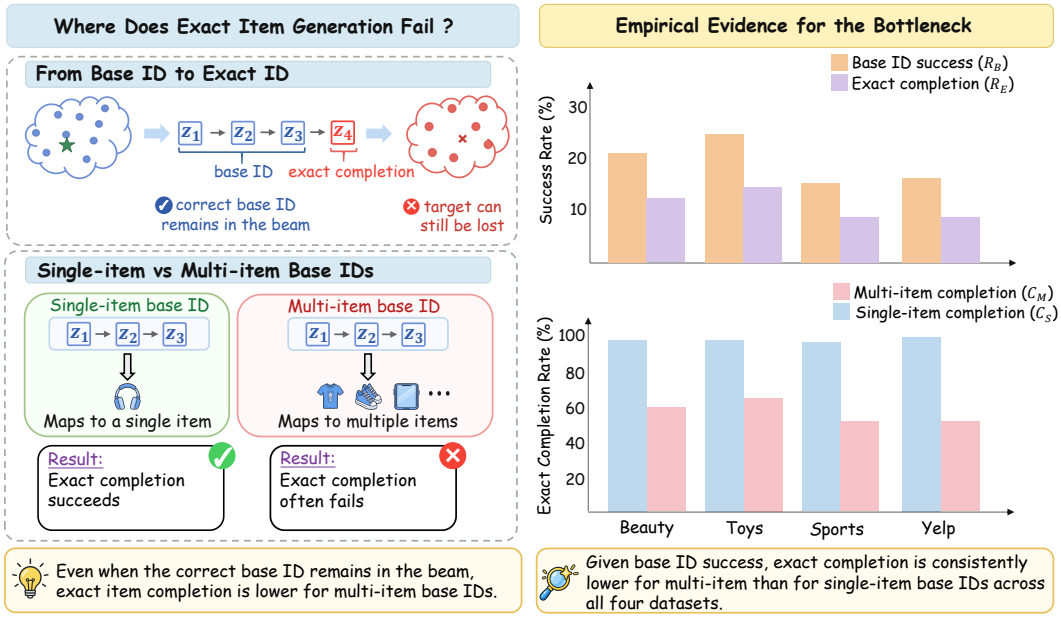

Figure 1: The exact item generation bottleneck in *Unique* at $b = 20$. Left: target loss during exact completion after the correct base ID has been retained. Right: base ID success and exact completion rates for multi-item targets (top), and exact completion conditioned on base ID success for single-item and multi-item base IDs (bottom).

These observations motivate relaxing the requirement that each generation target identify a single item. Final ranking remains item-level, whereas the generation target can represent a bounded item set. We introduce RELAX, which constructs a capacity-constrained Semantic ID hierarchy in which each complete path is shared by at most C items. This guarantees a qualifying prefix for every item. The shortest qualifying prefix becomes the *Shared ID* for that item, and the associated items form its bucket. The capacity bound controls the number of items introduced by each generated identifier, while selecting the shortest qualifying prefix avoids further hierarchy tokens once this bound is satisfied. A Shared ID can retain a complete hierarchy path while still representing multiple items. Shared IDs are fixed before generator training and used in both the discrete history and the training targets. At inference, each generated Shared ID expands to its item bucket, and a frozen continuous scorer ranks the union of these buckets.

Across four benchmarks, RELAX achieves the highest Recall@10 and NDCG@10 among the reproduced baselines under full-catalog evaluation. Controlled comparisons with the same frozen scorer show gains over unique item generation and prefix stopping applied only at inference, with similar average candidate set sizes in the latter comparison. Among targets lost by *Unique* after multi-item base ID success, RELAX returns 17.1–30.0% in the final top 10. Capacity and beam width analyses examine the trade-off between candidate coverage, candidate set size, and ranking performance. Additional beam width results and Shared ID structure are reported in Appendices C and D.

Our contributions are as follows:

- We identify and quantify the *exact item generation bottleneck* by separating base ID success from exact completion. The conditional completion gap persists across four benchmarks and all tested beam widths, and also appears in the reproduced TIGER baseline.
- We introduce RELAX, which constructs a capacity-constrained Semantic ID hierarchy and selects the shortest qualifying prefixes as Shared IDs for both histories and training targets.
- We show improvements over reproduced baselines and in controlled comparisons with the same frozen scorer. Recovery analysis connects the diagnosed failures to targets returned in the final top 10, while capacity and beam width analyses relate candidate coverage and size to ranking performance.

2 Related Work

Item identifier design and learning. Research on item identifiers in generative recommendation has explored how identifiers are constructed, what signals shape them, and what token structures they use. INDEX studies alternative indexing schemes, and TIGER represents each item with a Semantic ID obtained through residual quantization (Hua et al., 2023; Rajput et al., 2023). When multiple items receive the same quantization codes, TIGER appends an additional token so that the complete Semantic ID uniquely identifies one item (Rajput et al., 2023). LETTER incorporates semantic structure and collaborative signals while encouraging diversity in code assignments, and LC-Rec integrates language and collaborative semantics when learning item indices (Wang et al., 2024; Zheng et al., 2024). GRID further systematizes Semantic ID design choices in a modular framework for generative recommendation (Ju et al., 2025). Other work changes either the structure of the identifier or how it is optimized: SETRec represents an item with a set of order-agnostic identifier tokens, while DIGER makes Semantic ID learning differentiable so that the recommendation objective can directly shape the learned identifiers (Lin et al., 2025; Fu et al., 2026). IDGenRec learns textual identifiers, whereas ActionPiece makes action tokenization dependent on the surrounding interaction context (Tan et al., 2024; Hou et al., 2025).

Semantic ID collisions and sharing. Complete Semantic ID collisions occur when distinct items share an entire token sequence, so a correct identifier prediction need not uniquely identify the target item. Zhang et al. (2026) distinguish identifier matching from item-level retrieval and introduce evaluation metrics that account for this ambiguity. Their Zero-Collision Reassignment preserves earlier codewords and assigns distinct final codewords within each prefix group, provided the final codebook size is at least the largest group size. QuaSID uses Hamming distance to distinguish partial and complete Semantic ID overlaps and applies corresponding repulsion margins to continuous item representations, masking collaborative-positive and same-item pairs (Hu et al., 2026). AdaSID adapts collision regularization to semantic compatibility, in-batch collision load, and training progress (Pan et al., 2026). In our diagnostic, complete Semantic IDs uniquely identify items, but targets can still be lost during exact completion after the correct base ID has been retained in the beam. RELAX instead constructs Shared IDs that each represent at most C items, leaving exact item discrimination to continuous ranking over the expanded candidate set.

Variable-length Semantic IDs. Variable-length identifiers adapt token allocation to item properties or explicit length objectives. Khrylchenko (2026) proposes adaptive-length Semantic IDs with a discrete variational autoencoder and explicit length regularization. VarLenRec uses a popularity-based information budget to guide a learned controller that selects the number of quantization levels. It resolves remaining collisions with disambiguation tokens to obtain unique identifiers (Wang et al., 2026). CapsID uses capsule routing to construct identifiers, with length determined by routing confidence and residual norm subject to a maximum depth (Cheng et al., 2026). VaLiDRec selects and prunes tokens from a pretrained language model vocabulary according to token importance and preservation of item semantics. It aggregates token probabilities predicted in parallel into item scores, without autoregressive Semantic ID decoding (Qiao et al., 2026). For each item, RELAX selects the shortest prefix of its hierarchy path such that no more than C items share that prefix. A Shared ID may be a complete path shared by multiple items.

Discrete generation and item ranking. LIGER jointly learns Semantic ID generation and dense retrieval, using dense similarity to rank generated candidates together with cold-start items (Yang et al., 2025a). COBRA conditions dense vector generation on generated Semantic IDs and combines beam scores with nearest-neighbor similarity scores for item ranking (Yang et al., 2025b). Gryphon jointly trains Semantic ID generation and item-level scoring with a shared history encoder (Tikhonovich et al., 2026). It expands generated Semantic IDs to their associated items and ranks them using item scores rather than beam scores, allowing items with the same identifier to receive different scores. Gryphon-v2 uses rollout distillation to train item ranking and generates complete Semantic IDs that may be shared by multiple items (Lipkina et al., 2026). Its index is designed to keep collision classes small, while a separate candidate budget for each request is enforced by discarding identifier hypotheses with the lowest beam scores and their associated items. RELAX instead enforces the capacity bound on each generation target through hierarchy construction and selects the shortest prefix satisfying that bound. The selected Shared IDs are used in both discrete histories and training targets. Expanding at most b Shared IDs returned by beam search therefore yields at most bC candidate items, all ranked together by the frozen continuous scorer.

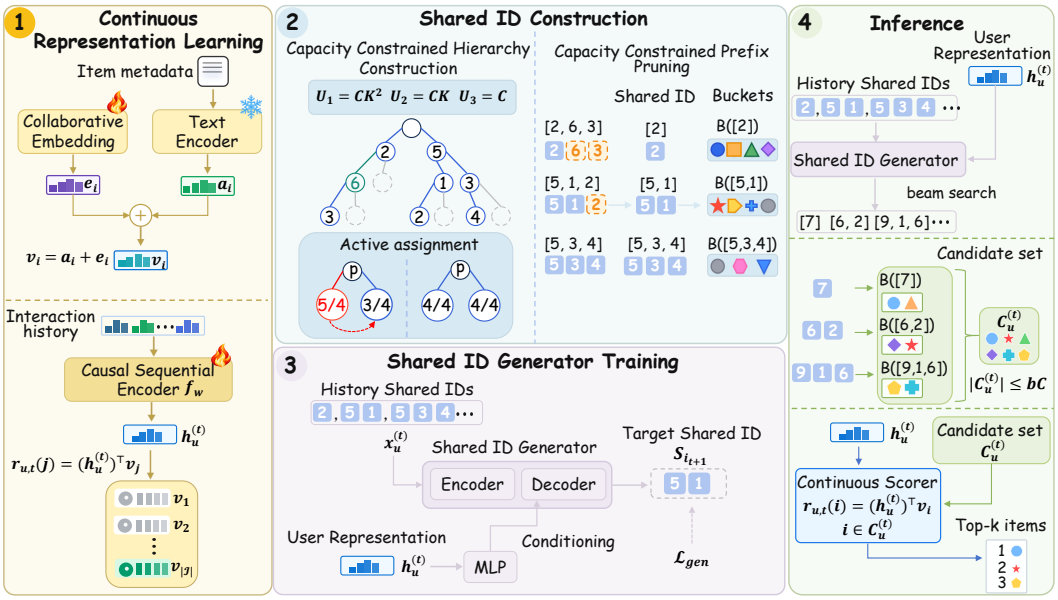

Figure 2: Overview of RELAX. A continuous recommender is trained and frozen before Shared ID construction and generator training. Each item is assigned the shortest prefix in the constructed hierarchy whose bucket contains at most C items. The generator uses Shared ID histories and continuous user guidance to predict Shared IDs. At inference, the generated buckets are expanded and their union is ranked by the frozen continuous scorer.

3 METHOD

Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets. Given the interaction history $s_u^{(t)} = (i_1, \ldots, i_t)$ of user $u \in \mathcal{U}$, the task is to rank items in $\mathcal{I}$ to predict the next item i_{t+1}. RELAX assigns each item i a Shared ID S_i that may represent multiple items. The items associated with a Shared ID S form its bucket,

$$\mathcal{B}(S) = \{i \in \mathcal{I} : S_i = S\}, \qquad |\mathcal{B}(S)| \le C, \tag{1}$$

where C is the maximum number of items represented by one Shared ID. Items in a bucket share the same generation target, while final recommendations remain item-level.

Figure 2 summarizes the training stages and inference. We first learn a continuous recommender from item metadata and interaction histories. Its item representations are then used to construct Shared IDs, and its user representation conditions the generator alongside the Shared ID history. The continuous recommender remains frozen while the generator is trained to predict the next item's Shared ID. At inference, the generated buckets define the candidate set, and the same continuous recommender scores the items in their union.

3.1 CONTINUOUS REPRESENTATION LEARNING

For item i, let $\mathbf{a}_i \in \mathbb{R}^d$ denote the fixed text embedding of its metadata and $\mathbf{e}_i \in \mathbb{R}^d$ its trainable collaborative embedding. We use $\mathbf{v}_i = \mathbf{a}_i + \mathbf{e}_i$ as the item representation. A causal sequential encoder f_ω maps the embedded interaction history to $\mathbf{h}_u^{(t)}$, whose dot product with $\mathbf{v}_i$ defines the continuous score $r_{u,i}^{(t)}$:

$$\begin{aligned} \mathbf{h}_u^{(t)} &= f_\omega\left(\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_t}\right), \\ r_{u,i}^{(t)} &= {\mathbf{h}_u^{(t)}}^\top \mathbf{v}_i. \end{aligned} \tag{2}$$

We train f_ω and the collaborative embeddings to predict the next item with cross-entropy over the full catalog $\mathcal{I}$, using the scores in Eq. (2) and keeping the text embeddings fixed. The sequential encoder and collaborative embeddings are then frozen for hierarchy construction, generator training,

and inference. The resulting item representations $\mathbf{v}_i$ define the hierarchy, the user representation $\mathbf{h}_u^{(t)}$ provides guidance, and $r_{u,i}^{(t)}$ ranks candidate items at inference. Implementation details are given in Appendix A.2.

3.2 Shared ID Construction

Shared ID construction proceeds in two steps. We first build a capacity-constrained hierarchy in which each path of length L is shared by at most C items. This guarantees that every item has at least one prefix satisfying the capacity bound. We then select the shortest qualifying prefix as its Shared ID and generation target.

Capacity-Constrained Hierarchy. We construct a residual K-means hierarchy (Ju et al., 2025) with branching factor K and depth L satisfying $CK^L \geq |\mathcal{I}|$. This ensures sufficient aggregate capacity for the catalog at depth L. Let p denote a prefix of length $\ell - 1$, and let $\mathcal{I}(p)$ denote the items sharing that prefix. A child at level ℓ has $K^{L-\ell}$ possible descendant paths of length L. Since each such path can represent at most C items, we limit the number of items assigned to that child to

$$U_\ell = CK^{L-\ell}, \qquad \ell = 1, \ldots, L, \tag{3}$$

with $U_L = C$. At the root, $|\mathcal{I}| \leq KU_1$ by the choice of L. At subsequent levels, each prefix group contains at most $U_{\ell-1} = KU_\ell$ items. The K children therefore have sufficient aggregate capacity to accommodate every item in the group.

Let $\mathbf{q}_i^{(1)} = \mathbf{v}_i$ be the initial residual and $\bar{\mathbf{q}}_i^{(\ell)}$ the L_2-normalized residual at level ℓ. At each level, one codebook $\{\boldsymbol{\mu}_j^{(\ell)}\}_{j=1}^K$ is learned from the normalized residuals over the catalog and shared across all prefix groups. The assignment cost is $D_{ij} = \|\bar{\mathbf{q}}_i^{(\ell)} - \boldsymbol{\mu}_j^{(\ell)}\|_2^2$. Within each prefix group, nearest centroid assignments are retained if every child contains at most U_ℓ items. Otherwise, with the centroids fixed, let $\pi : \mathcal{I}(p) \to \{1, \ldots, K\}$ denote the reassignment. We solve

$$\begin{aligned} &\min_{\pi} \sum_{i \in \mathcal{I}(p)} D_{i,\pi(i)} \\ &\text{s.t. } \big|\{i \in \mathcal{I}(p) : \pi(i) = j\}\big| \leq U_\ell, \qquad j = 1, \ldots, K. \end{aligned} \tag{4}$$

We call this constrained reassignment *Active*. The transportation linear program in Appendix A.2 minimizes the total assignment cost within the current prefix group for the fixed centroids. Let $z_i^{(\ell)}$ denote the assigned centroid index. For $\ell < L$, the residual for the next level is $\mathbf{q}_i^{(\ell+1)} = \bar{\mathbf{q}}_i^{(\ell)} - \boldsymbol{\mu}_{z_i^{(\ell)}}^{(\ell)}$. The next codebook is learned after normalizing these updated residuals, so the capacity assignments also determine the inputs to subsequent levels. Together with the feasibility argument above, enforcing U_ℓ at each level ensures that every completed path represents at most C items.

Shared ID Selection. Let $Z_i = (z_i^{(1)}, \ldots, z_i^{(L)})$ denote the depth-L path assigned to item i, and let $Z_i^{1:\ell}$ denote its prefix of length ℓ. Although every depth-L path satisfies the capacity constraint, an earlier prefix may already be shared by at most C items. RELAX uses the shortest such prefix as the Shared ID generation target:

$$\ell_i^\star = \min\left\{\ell \in \{1, \ldots, L\} : \left|\mathcal{I}\big(Z_i^{1:\ell}\big)\right| \leq C\right\}, \qquad S_i = Z_i^{1:\ell_i^\star}. \tag{5}$$

Here $\mathcal{I}(Z_i^{1:\ell})$ denotes the items sharing the prefix $Z_i^{1:\ell}$. All items sharing S_i also share its shorter prefixes. By the minimality of $\ell_i^\star$, every shorter nonempty prefix is shared by more than C items, whereas S_i is shared by at most C items. Every item in $\mathcal{I}(S_i)$ therefore selects the same Shared ID S_i, giving $\mathcal{B}(S_i) = \mathcal{I}(S_i)$ and $|\mathcal{B}(S_i)| \leq C$. The Shared IDs are fixed before generator training. Hierarchy tokens beyond the selected prefix are excluded from the generation target because they are not required to satisfy the capacity bound. We call this operation *capacity-constrained prefix pruning* (*Pruning* in the controlled comparisons). A Shared ID retains length L when no shorter nonempty prefix satisfies the bound, and it can still represent multiple items. Generating a complete hierarchy path therefore does not imply unique item identification. The set of distinct Shared IDs is prefix-free. Suppose that S_i were a proper prefix of another Shared ID S_j. Item j would then

share S_i, whose associated item set contains at most C items. Thus, S_i would already satisfy the capacity criterion for item j, contradicting the selection of S_j as the shortest qualifying prefix. No Shared ID is therefore a proper prefix of another, allowing trie-constrained decoding to terminate unambiguously when a Shared ID is reached.

3.3 Shared ID Generator Training

For history $s_u^{(t)} = (i_1, \ldots, i_t)$, we replace each item i_τ with the token sequence of its Shared ID S_{i_τ}, preserving interaction order to form $x_u^{(t)}$. Items assigned to the same Shared ID have identical token sequences in this discrete history. We also condition the generator on the frozen user representation $\mathbf{h}_u^{(t)}$, computed from the original sequence of continuous item representations through Eq. (2). This supplies an additional input derived from the individual items rather than only their shared identifiers. Following TIGER (Rajput et al., 2023), the generator is an encoder–decoder Transformer (Vaswani et al., 2017). The encoder processes $x_u^{(t)}$, while an MLP maps the user representation to $\mathbf{c}_u^{(t)} = g_\eta(\mathbf{h}_u^{(t)})$. This projection supplies prefix key and value states for decoder self-attention (Li & Liang, 2021), conditioning generation on both the Shared ID history and continuous user guidance.

For target item i_{t+1}, let $\ell^\star = \ell^\star_{i_{t+1}}$ and $S^\star = S_{i_{t+1}} = (z_1^\star, \ldots, z_{\ell^\star}^\star)$. The decoder predicts this Shared ID autoregressively. During training, each prediction is conditioned on the preceding tokens of the target Shared ID, $z^\star_{<\ell} = (z_1^\star, \ldots, z^\star_{\ell-1})$. For a training pair (u, t), the loss is

$$\mathcal{L}_{\text{gen}}(u, t) = -\sum_{\ell=1}^{\ell^\star} \log p_\phi \left(z_\ell^\star \mid x_u^{(t)}, \mathbf{c}_u^{(t)}, z^\star_{<\ell} \right). \tag{6}$$

The target length $\ell^\star$ is determined by Eq. (5), so the loss covers only the identifier tokens needed to reach the selected bucket. Items in the same bucket share the same target sequence; the generator is trained to predict that sequence rather than distinguish the items within the bucket. We optimize the generator parameters ϕ and conditioning parameters η jointly, keeping the continuous recommender frozen.

3.4 Shared ID Generation and Candidate Ranking

We use beam search constrained by a trie containing all Shared IDs. Because the identifiers are prefix-free, a hypothesis terminates upon reaching a valid Shared ID. Let $\widehat{\mathcal{S}}_u^{(t)}$ contain the distinct completed Shared IDs returned with beam width b, with $|\widehat{\mathcal{S}}_u^{(t)}| \le b$. Expanding each returned identifier to its item bucket gives the candidate set

$$\mathcal{C}_u^{(t)} = \bigcup_{S \in \widehat{\mathcal{S}}_u^{(t)}} \mathcal{B}(S), \qquad |\mathcal{C}_u^{(t)}| \le bC. \tag{7}$$

The bound follows because at most b identifiers are returned and each bucket contains at most C items. The frozen continuous scorer evaluates every candidate $i \in \mathcal{C}_u^{(t)}$ using $r_{u,i}^{(t)}$ from Eq. (2). Candidates from all generated buckets are ranked together by these item scores, rather than by Shared ID likelihoods, and up to k items are returned.

4 Experiments

4.1 Experimental Setup

Datasets and evaluation. We evaluate on Amazon Beauty, Amazon Toys and Games, Amazon Sports and Outdoors, and Yelp (McAuley et al., 2015; He & McAuley, 2016; Yelp, 2026). We use chronological leave-one-out splitting: the last interaction is used for testing, the penultimate for validation, and the rest for training. Histories contain at most the 20 most recent interactions. We report item-level Recall@k (R@k) and NDCG@k (N@k) for $k \in \{5, 10\}$ under full-catalog evaluation. History items are masked, with the target item retained. For methods with an explicit candidate stage, Candidate Recall is the fraction of evaluation cases with the target in the candidate set before ranking; Avg. Candidates reports the mean number of candidate items per case. Dataset statistics and further evaluation details appear in Appendix A.1.

Baselines and settings. We compare RELAX against three sequential recommenders: GRU4Rec, BERT4Rec, and SASRec (Hidasi et al., 2016; Sun et al., 2019; Kang & McAuley, 2018). The generative and hybrid baselines are TIGER, LC-Rec, LETTER, ETEGRec, DIGER, SETRec, LIGER, Gryphon, and COBRA (Rajput et al., 2023; Zheng et al., 2024; Wang et al., 2024; Liu et al., 2025; Fu et al., 2026; Lin et al., 2025; Yang et al., 2025a; Tikhonovich et al., 2026; Yang et al., 2025b). All methods share the same data splits, history limit, and full-catalog protocol; checkpoints are selected by validation Recall@10. For methods using item metadata, we provide the same fields while retaining the representation and tokenization procedures of each method. Methods using autoregressive beam search use $b = 20$. Unless otherwise specified, RELAX uses Shared ID capacity $C = 4$. Implementation details appear in Appendices A.2 and A.3.

4.2 OVERALL PERFORMANCE

Table 1 compares Recall@10 and NDCG@10 under full-catalog evaluation. RELAX achieves the highest value on both metrics across all four datasets. For both reported metrics, the strongest reproduced baseline is LIGER on Beauty and Toys, SETRec on Sports, and ETEGRec on Yelp. Averaging the relative improvements over these baselines across the four datasets gives gains of 11.4% in Recall@10 and 19.7% in NDCG@10.

Table 1: Recommendation performance under full-catalog evaluation. R@10 and N@10 denote Recall@10 and NDCG@10. The best result in each column is bold, and the second best is underlined.

	Beauty		Toys		Sports		Yelp	
Method	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
Sequential recommendation								
GRU4Rec	0.0643	0.0357	0.0549	0.0311	0.0373	0.0197	0.0398	0.0207
BERT4Rec	0.0347	0.0170	0.0355	0.0178	0.0233	0.0116	0.0401	0.0210
SASRec	0.0588	0.0313	0.0442	0.0241	0.0220	0.0113	0.0296	0.0152
Generative and hybrid recommendation								
TIGER	0.0610	0.0331	0.0511	0.0267	0.0400	0.0225	0.0407	0.0213
LC-Rec	0.0642	0.0374	0.0753	0.0453	0.0451	0.0246	0.0359	0.0199
LETTER	0.0672	0.0364	0.0625	0.0373	0.0456	0.0292	0.0426	0.0231
ETEGRec	0.0870	0.0503	0.0829	0.0488	0.0484	0.0266	<u>0.0598</u>	<u>0.0326</u>
SETRec	0.0714	0.0380	0.0760	0.0423	<u>0.0533</u>	<u>0.0302</u>	0.0358	0.0177
DIGER	0.0519	0.0267	0.0522	0.0276	0.0265	0.0132	0.0199	0.0100
LIGER	<u>0.0899</u>	<u>0.0512</u>	<u>0.0895</u>	<u>0.0516</u>	0.0490	0.0279	0.0526	0.0286
COBRA	0.0352	0.0192	0.0289	0.0156	0.0186	0.0096	0.0262	0.0145
Gryphon	0.0575	0.0327	0.0499	0.0276	0.0279	0.0150	0.0377	0.0202
RELAX	**0.1000**	**0.0605**	**0.1040**	**0.0656**	**0.0573**	**0.0336**	**0.0662**	**0.0399**

4.3 RELAXING EXACT ITEM GENERATION

Table 2 examines how relaxing exact item generation affects candidate coverage and final ranking. Within each dataset, the generative variants use the same item representations, frozen user representations for guidance, generator architecture, and training settings. All configurations with continuous ranking share the same frozen scorer. *None* ranks the full catalog directly. *Unique* generates complete unique Semantic IDs, and *Unique + Scorer* applies the frozen scorer to the same generated item candidates. *Unique + Pruning + Scorer* constructs the same complete unique IDs, then selects the shortest prefixes shared by at most C items for both histories and training targets before generator training. *Active + Scorer* instead generates complete paths from the capacity-constrained hierarchy, where each path can represent multiple items. *Active + PrefixStop + Scorer* reuses this trained generator but stops at the first prefix shared by at most C items. RELAX uses the same capacity-constrained hierarchy but selects Shared IDs before generator training and uses them in both histories and training targets. For every generative variant with continuous ranking, the item sets associated with the identifiers returned by beam search form the candidate set ranked by the frozen scorer. Appendix A.4 provides the complete configuration definitions.

Table 2: Controlled comparisons at $b = 20$, with $C = 4$ where applicable. A checkmark denotes final ranking with the same frozen continuous scorer. Cand. Recall measures target coverage in the candidate set; Avg. Cand. is the mean candidate set size. The best value for each ranking metric within a dataset is bold.

Dataset	Generation	Scorer	Cand. Recall	Avg. Cand.	R@5	N@5	R@10	N@10
Beauty	None	✓	–	–	0.0570	0.0382	0.0779	0.0446
	Unique	–	–	–	0.0605	0.0417	0.0870	0.0502
	Unique	✓	0.1007	18.83	0.0666	0.0478	0.0918	0.0556
	Unique + Pruning	✓	0.1493	40.18	0.0694	0.0498	0.0964	0.0586
	Active	✓	0.1287	25.84	0.0686	0.0491	0.0952	0.0579
	Active + PrefixStop	✓	0.1526	40.35	0.0699	0.0500	0.0970	0.0588
	RELAX	✓	0.1571	39.92	**0.0718**	**0.0514**	**0.1000**	**0.0605**
Toys	None	✓	–	–	0.0696	0.0418	0.0834	0.0465
	Unique	–	–	–	0.0664	0.0458	0.0891	0.0529
	Unique	✓	0.0991	18.85	0.0725	0.0524	0.0942	0.0593
	Unique + Pruning	✓	0.1505	38.16	0.0748	0.0546	0.0991	0.0626
	Active	✓	0.1244	24.99	0.0755	0.0552	0.1000	0.0633
	Active + PrefixStop	✓	0.1492	38.27	0.0758	0.0555	0.1010	0.0638
	RELAX	✓	0.1479	37.94	**0.0777**	**0.0571**	**0.1040**	**0.0656**
Sports	None	✓	–	–	0.0262	0.0191	0.0475	0.0254
	Unique	–	–	–	0.0312	0.0213	0.0490	0.0270
	Unique	✓	0.0561	19.12	0.0362	0.0251	0.0516	0.0301
	Unique + Pruning	✓	0.0893	39.04	0.0386	0.0267	0.0545	0.0320
	Active	✓	0.0755	27.52	0.0377	0.0262	0.0537	0.0314
	Active + PrefixStop	✓	0.0879	38.86	0.0386	0.0267	0.0550	0.0319
	RELAX	✓	0.0908	38.81	**0.0409**	**0.0283**	**0.0573**	**0.0336**
Yelp	None	✓	–	–	0.0375	0.0244	0.0593	0.0321
	Unique	–	–	–	0.0315	0.0205	0.0484	0.0259
	Unique	✓	0.0624	18.86	0.0386	0.0275	0.0596	0.0342
	Unique + Pruning	✓	0.0750	19.06	0.0403	0.0290	0.0619	0.0359
	Active	✓	0.1204	54.42	0.0415	0.0305	0.0632	0.0375
	Active + PrefixStop	✓	0.1196	54.30	0.0415	0.0305	0.0633	0.0375
	RELAX	✓	0.1170	54.11	**0.0437**	**0.0326**	**0.0662**	**0.0399**

Adding continuous ranking improves *Unique* on all four ranking metrics across all datasets. Both *Unique + Pruning + Scorer* and *Active + Scorer* further improve candidate coverage and final ranking over *Unique + Scorer*. RELAX achieves the highest values on all four ranking metrics. Prefix stopping improves the Active model on Beauty, Toys, and Sports, whereas the ranking results on Yelp are nearly unchanged. Compared with *Active + PrefixStop + Scorer*, RELAX improves mean R@10 by 0.0023–0.0030 and mean N@10 by 0.0017–0.0024, with slightly smaller average candidate sets on every dataset. This comparison supports using Shared IDs in both histories and training targets beyond applying the stopping rule only at inference. On Toys and Yelp, *Active + PrefixStop + Scorer* has higher Candidate Recall than RELAX (0.1492 vs. 0.1479 and 0.1196 vs. 0.1170, respectively), yet lower values on all four ranking metrics. With the scorer fixed, these results show that the composition of the candidate set matters in addition to target coverage.

4.4 RECOVERY FROM EXACT COMPLETION FAILURES

We evaluate whether RELAX recovers targets lost by *Unique* during exact completion (Figure 1). At $b = 20$, we select cases in which the target has a multi-item base ID, the correct base ID remains in the beam, and exact completion fails. Shared ID Recovery is the fraction of these cases for which RELAX generates the Shared ID associated with the target item; Recovered gives the corresponding count. Recall@k and NDCG@k are computed over the same failure subset after candidate expansion and continuous ranking.

Across the four datasets, RELAX generates the target Shared ID for 36.6–49.4% of the diagnosed failure cases (Table 3). After continuous ranking, the target item appears in the final top 10 for 17.1–30.0% of the same cases. Generating the correct Shared ID restores the target to the candidate

set, but the target must still compete with items in all returned buckets for a position in the final list. The final ranking results therefore establish that some of the losses identified by the diagnostic can be recovered as recommendations, rather than only as matches to the target Shared ID.

Table 3: Recovery on cases with multi-item base ID success followed by exact completion failure at $b = 20$.

Dataset	Failures	Recovered	Shared ID Recovery	R@5	N@5	R@10	N@10
Beauty	437	216	0.4943	0.2010	0.1540	0.3000	0.1857
Toys	337	159	0.4718	0.1960	0.1320	0.2970	0.1643
Sports	712	272	0.3820	0.1080	0.0740	0.1910	0.1006
Yelp	533	195	0.3659	0.0990	0.0840	0.1710	0.1070

4.5 Effect of Shared ID Capacity

Shared ID capacity C bounds the number of items represented by one generated identifier, whereas beam width b limits the number of returned identifiers. We vary $C \in \{2, 4, 8\}$ at $b = 20$ with the continuous scorer fixed. Table 4 reports average bucket size, Candidate Recall, average candidate set size, and ranking performance. This analysis examines whether the additional coverage obtained with larger buckets improves final ranking at a fixed beam width.

Table 4: Effect of Shared ID capacity at fixed beam width $b = 20$. Avg. Size denotes mean bucket size; † marks the default capacity $C = 4$.

Dataset	C	Avg. Size	Cand. Recall	Avg. Cand.	R@5	N@5	R@10	N@10
Beauty	2	1.17	0.1324	25.77	0.0696	0.0503	0.0966	0.0590
	$4^{\dagger}$	1.38	0.1571	39.92	0.0718	0.0514	0.1000	0.0605
	8	1.56	0.1823	65.84	0.0716	0.0513	0.0998	0.0604
Toys	2	1.17	0.1299	25.47	0.0773	0.0563	0.1025	0.0644
	$4^{\dagger}$	1.37	0.1479	37.94	0.0777	0.0571	0.1040	0.0656
	8	1.52	0.1694	58.43	0.0776	0.0570	0.1038	0.0655
Sports	2	1.16	0.0768	26.66	0.0385	0.0267	0.0546	0.0319
	$4^{\dagger}$	1.42	0.0908	38.81	0.0409	0.0283	0.0573	0.0336
	8	1.63	0.1146	70.62	0.0408	0.0282	0.0574	0.0337
Yelp	2	1.39	0.0934	33.27	0.0431	0.0317	0.0610	0.0375
	$4^{\dagger}$	1.60	0.1170	54.11	0.0437	0.0326	0.0662	0.0399
	8	1.81	0.1436	83.40	0.0438	0.0326	0.0658	0.0397

Increasing C enlarges the average bucket and candidate set and raises Candidate Recall on all four datasets. Moving from $C = 2$ to $C = 4$ also improves all four ranking metrics. Increasing C from 4 to 8 further improves coverage but raises the average candidate count by 54.0%–82.0%, while each reported ranking metric changes by at most 0.0004 in absolute value. Thus, the extra coverage at $C = 8$ does not yield a corresponding improvement in final ranking. Appendix C examines the complementary effect of varying the number of returned Shared IDs through beam width.

5 Conclusion

We identify an exact item generation bottleneck: targets can be lost during exact completion even after the correct base ID remains in the beam. RELAX relaxes this requirement by constructing capacity-bounded Shared IDs from the shortest qualifying prefixes, followed by item-level ranking with a frozen continuous scorer. Across four benchmarks, RELAX improves over reproduced baselines and controlled variants using the same frozen scorer, while recovery analysis shows that some exact-completion failures return to the final top 10. These results support bounding the item set represented by each generation target rather than requiring unique item identification. Our evaluation does not cover strict cold-start settings for users or items with little or no collaborative signal.

AI USE STATEMENT

Generative AI tools were used for language editing, literature search, and software development, including code completion, debugging, and refactoring. They also provided feedback on experimental design and comparisons, interpretation of results, and the checking and presentation of mathematical arguments. The authors made the final research decisions and reviewed the AI-assisted text, references, and mathematical arguments. AI-assisted code used in the reported experiments was reviewed and tested by the authors. The authors take responsibility for the final manuscript, implementations, and reported results.

REPRODUCIBILITY STATEMENT

The experimental protocol and checkpoint selection are described in Section 4.1 and Appendix A.1. Appendix A.2 describes continuous representation learning, Shared ID construction, generator training, and inference. Baseline implementations and controlled variants are documented in Appendices A.3 and A.4, respectively. The bottleneck setup and diagnostic metrics are specified in Appendices A.5 and B.1. An anonymized implementation is available at `https://github.com/Sanfeixiao/RELAX`.

REFERENCES

Wenzhuo Cheng, Menghang Gong, Qixin Guo, Hang Zheng, Zhaobin Yang, Jianguo Lou, and Zhengwei Zheng. CapsID: Soft-routed variable-length semantic IDs for generative recommendation. *arXiv preprint arXiv:2605.05096*, 2026. URL https://arxiv.org/abs/2605.05096.

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.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 299–315, 2022. doi: 10.1145/3523227.3546767.

Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 507–517, 2016. doi: 10.1145/2872427.2883037.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06939.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H. Chi, Julian McAuley, and Derek Zhiyuan Cheng. ActionPiece: Contextually tokenizing action sequences for generative recommendation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24004–24024. PMLR, 2025. URL https://proceedings.mlr.press/v267/hou25f.html.

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. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 7401–7411, 2026. doi: 10.1145/3770855.3818486.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item IDs for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 195–204, 2023. doi: 10.1145/3624918.3625339.

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*, pp. 6420–6425, 2025. doi: 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, 2018. doi: 10.1109/ICDM.2018.00035.

Kirill Khrylchenko. Variable-length semantic IDs for recommender systems. *arXiv preprint arXiv:2602.16375*, 2026. URL https://arxiv.org/abs/2602.16375.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. Order-agnostic identifier for large language model-based generative recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1923–1933, 2025. doi: 10.1145/3726302.3730053.

Anna Lipkina, Daria Tikhonovich, Viktor Yanush, Mariia Ulianova, Oleg Sorokin, Vladislav Dodonov, Ilya Murzin, Denis Burshtein, and Nikolay Savushkin. Gryphon-v2: One model in place of a cascade - generate-and-rank recommender with rollout distillation. *arXiv preprint arXiv:2608.06213*, 2026. URL https://arxiv.org/abs/2608.06213.

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.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, 2015. doi: 10.1145/2766462.2767755.

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. *arXiv preprint arXiv:2604.23522*, 2026. URL https://arxiv.org/abs/2604.23522.

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. *arXiv preprint arXiv:2607.25209*, 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 Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10299–10315, 2023. doi: 10.52202/075280-0452.

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*, pp. 1441–1450, 2019. doi: 10.1145/3357384.3357895.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. IDGenRec: LLM-RecSys alignment with textual ID learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 355–364, 2024. doi: 10.1145/3626772.3657821.

Daria Tikhonovich, Oleg Sorokin, Vladislav Dodonov, Mariia Ulianova, and Ilya Murzin. Gryphon: A unified architecture for semantic-ID generation and item-level scoring in industrial recommendations. *arXiv preprint arXiv:2606.08604*, 2026. URL https://arxiv.org/abs/2606.08604.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Minhao Wang, Bowen Wu, and Wei Zhang. Learning variable-length tokenization for generative recommendation. *arXiv preprint arXiv:2605.17779*, 2026. URL https://arxiv.org/abs/2605.17779.

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 33rd ACM International Conference on Information and Knowledge Management*, pp. 2400–2409, 2024. doi: 10.1145/3627673.3679569.

Liu Yang, Fabian Paischer, Kaveh Hassani, Jiacheng Li, Shuai Shao, Zhang Gabriel Li, Yun He, Xue Feng, Nima Noorshams, Sem Park, Bo Long, Robert D. Nowak, Xiaoli Gao, and Hamid Eghbalzadeh. Unifying generative and dense retrieval for sequential recommendation. *Transactions on Machine Learning Research*, 2025a. URL https://openreview.net/forum?id=jxdnFIsjCb.

Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, and Lin Liu. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. In *Advances in Neural Information Processing Systems*, volume 38, pp. 104045–104069, 2025b. doi: 10.52202/085713-3135.

Yelp. Yelp open dataset. https://business.yelp.com/data/resources/open-dataset/, 2026. Accessed 2026-08-22.

Qian Zhang, Lech Szymanski, Haibo Zhang, and Jeremiah D. Deng. How reliable are semantic-ID tokenizer comparisons in generative recommendation? *arXiv preprint arXiv:2605.25330*, 2026. URL https://arxiv.org/abs/2605.25330.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pp. 4653–4664, 2021. doi: 10.1145/3459637.3482016.

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 Experimental Details

A.1 Datasets and Evaluation

We evaluate on four sequential recommendation benchmarks: Amazon Beauty, Amazon Toys and Games, Amazon Sports and Outdoors, and Yelp (McAuley et al., 2015; He & McAuley, 2016; Yelp, 2026). We use the 5-core versions of the three Amazon datasets.

Table 5: Dataset statistics before leave-one-out splitting.

Dataset	#Users	#Items	#Interactions	Avg. Seq. Len.
Beauty	22,363	12,101	198,502	8.88
Toys	19,412	11,924	167,597	8.63
Sports	35,598	18,357	296,337	8.32
Yelp	30,431	20,033	316,354	10.40

We use chronological leave-one-out splitting: each user's last interaction is used for testing, the penultimate interaction for validation, and all preceding interactions for training. Histories contain at most the 20 most recent interactions. Evaluation uses the full item catalog, with history items masked and the target item retained. We report Recall@k (R@k) and NDCG@k (N@k) for $k \in \{5, 10\}$. For methods with an explicit candidate stage, Candidate Recall measures target coverage and Avg. Candidates reports the mean candidate set size before final ranking. Checkpoints for RELAX and all baselines are selected by validation Recall@10. Unless otherwise stated, recommendation metrics are averaged over three random seeds.

A.2 RELAX Implementation

Continuous representation learning. For Beauty, Toys, and Sports, we form each item's text from its title, brand, category, and description; for Yelp, we use the business name, categories, and city. Fields are concatenated in the listed order, with each value preceded by its field name. We encode the resulting text with Qwen3-Embedding-0.6B (Zhang et al., 2025). The text embedding $\mathbf{a}_i$ is obtained by retaining the first 256 dimensions of the final hidden state at the last nonpadding token, followed by L_2 normalization. These text embeddings are precomputed and kept fixed thereafter. The continuous recommender is a two-layer SASRec-style causal Transformer (Kang & McAuley, 2018). We train it with Adam using a learning rate of 10^{-3} and batch size 128, and freeze the trained model before Shared ID construction and generator training.

Shared ID construction. We use branching factor $K = 256$, hierarchy depth $L = 3$, and default capacity $C = 4$. Let $\mathbf{q}_i^{(1)} = \mathbf{v}_i$ denote the initial residual. At level ℓ, we L_2-normalize the current residual $\mathbf{q}_i^{(\ell)}$ to obtain $\bar{\mathbf{q}}_i^{(\ell)}$. A single codebook $\{\boldsymbol{\mu}_j^{(\ell)}\}_{j=1}^{K}$ is learned from the normalized residuals over the catalog using minibatch K-means with K-means++ initialization. The codebook is shared across all prefix groups at that level, with assignment cost $D_{ij} = \|\bar{\mathbf{q}}_i^{(\ell)} - \boldsymbol{\mu}_j^{(\ell)}\|_2^2$. Within each prefix group, items are first assigned to their nearest centroids. If every child contains at most U_ℓ items, these assignments are retained. Otherwise, Active reassigns the items while keeping the centroids fixed. We implement the constrained assignment in Eq. (4) as the transportation linear program

$$
\begin{aligned}
\min_{\{y_{ij}\}} \quad & \sum_{i \in \mathcal{I}(p)} \sum_{j=1}^{K} D_{ij} y_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{K} y_{ij} = 1, && i \in \mathcal{I}(p), \\
& \sum_{i \in \mathcal{I}(p)} y_{ij} \le U_\ell, && j \in \{1, \dots, K\}, \\
& 0 \le y_{ij} \le 1.
\end{aligned}
$$

The constraint matrix of this transportation problem is totally unimodular, and its right-hand side is integral. The linear program therefore admits an integral optimal extreme point. We solve it with the HiGHS dual simplex solver through SciPy and verify integrality and capacity feasibility. The integral solution defines the capacity-feasible assignment $z_i^{(\ell)}$, which determines the residual passed to the next level:

$$\mathbf{q}_i^{(\ell+1)} = \bar{\mathbf{q}}_i^{(\ell)} - \boldsymbol{\mu}^{(\ell)}_{z_i^{(\ell)}}, \qquad \ell < L. \tag{8}$$

The resulting residuals are used to learn the codebook at level $\ell + 1$. After constructing all L levels, we select for each item the shortest qualifying prefix according to Eq. (5). The resulting Shared IDs are fixed before generator training and are used for both the discrete interaction histories and generation targets.

Generator training and inference. The Shared ID generator follows the encoder–decoder Transformer architecture used by TIGER (Rajput et al., 2023), with four layers in both the encoder and decoder. A one-layer MLP projects the frozen continuous user representation into prefix key and value states for decoder self-attention, providing continuous user guidance alongside the Shared ID history. During generator training, the continuous recommender remains frozen, while the generator and conditioning projection are optimized jointly. We use Adam with a learning rate of 10^{-2} and batch size 128. At inference, we follow the trie-constrained decoding and candidate ranking procedure in Section 3.4, using beam width $b = 20$ unless otherwise specified. All experiments were run on eight NVIDIA A100-SXM4 40 GB GPUs.

A.3 Baseline Implementations

All baselines follow the data and evaluation protocol in Appendix A.1. For methods that use item metadata, we provide the fields described in Appendix A.2 while retaining the representation and tokenization procedures of each method. We use the RecBole implementations of GRU4Rec, BERT4Rec, and SASRec (Zhao et al., 2021). For LC-Rec, LETTER, ETEGRec, DIGER, SETRec, and LIGER, we use the released implementations and adapt data loading and evaluation to the common protocol. For LETTER, we use the released instantiation based on TIGER. TIGER, Gryphon, and COBRA are implemented according to the corresponding papers. Training follows the released configurations when available and otherwise the settings reported in the corresponding papers. Checkpoints are selected by validation Recall@10. For methods using autoregressive beam search, the main comparison uses $b = 20$. Methods without beam search retain their respective inference procedures.

A.4 Controlled Variant Definitions

The variants in Table 2 differ in identifier construction, the IDs used in discrete histories and training targets, the decoding stopping rule, and final continuous ranking. Within each dataset, all generative variants use the same interaction sequences, item representations, and frozen user representations. The generator architecture, conditioning mechanism, and training settings follow Appendix A.2, with checkpoint selection as described in Appendix A.1. Within each trained generator, histories and training targets use the same identifier representation. For variants with continuous ranking, beam search returns identifiers that are mapped to item candidates. A complete unique ID contributes one item, while a shared path or prefix contributes all items in its bucket. The same frozen scorer ranks the union of these item sets. *None* omits generation and applies this scorer directly to the full catalog.

Unique constructs complete unique Semantic IDs by appending a disambiguation token to each base ID of three tokens, following TIGER (Rajput et al., 2023). The additional token distinguishes items sharing a base ID, so every complete identifier maps to a single catalog item. Complete unique IDs are used in both the discrete history and the training targets. Beam search generates all four identifier tokens, and *Unique* returns the associated item candidates without final continuous ranking. *Unique* + *Scorer* reuses the same trained generator and ranks the same candidate set with the frozen scorer. Candidate statistics for both configurations are reported in the *Unique* + *Scorer* row. *Unique* + *Pruning* + *Scorer* starts from the same complete unique IDs as *Unique* + *Scorer*, then applies prefix pruning to this fully disambiguated hierarchy before generator training. For each item, the shortest prefix shared by at most C items is selected according to the capacity criterion in

Section 3.2. Prefix selection considers all four identifier levels, including the disambiguation level; the complete unique ID is retained when no shorter prefix satisfies the capacity bound. The selected prefixes replace the complete unique IDs in both the discrete history and the training targets. At inference, beam search generates these prefixes directly, and their associated item buckets form the candidate set for continuous ranking.

Active + Scorer uses the capacity-constrained hierarchy from Section 3.2 without prefix pruning. Complete paths of length L are used in both the discrete history and the training targets. Beam search generates these paths to depth L, and their associated item buckets form the candidate set. Complete paths can represent multiple items, with bucket size bounded by C; no disambiguation token is appended. *Active + PrefixStop + Scorer* reuses the trained generator and conditioning projection from *Active + Scorer*, with the same history encoding, hierarchy, and frozen scorer. Only the decoding stopping rule changes: a hypothesis terminates at the first prefix shared by at most C items, and all items sharing that prefix enter the candidate set. RELAX uses the same capacity-constrained hierarchy but selects the shortest qualifying prefix for each item before generator training, following Eq. (5). The selected prefixes are fixed as Shared IDs and used in both the discrete history and the training targets. Decoding terminates upon reaching a valid Shared ID, and the associated item buckets form the candidate set for continuous ranking. The comparison between *Active + Scorer* and *Active + PrefixStop + Scorer* measures the effect of the stopping rule for a fixed trained generator. The comparison between *Active + PrefixStop + Scorer* and RELAX evaluates the joint change to history encoding and training targets under the same hierarchy, capacity criterion, and frozen scorer.

A.5 BOTTLENECK DIAGNOSTIC SETUP

The main bottleneck diagnostic uses the *Unique* configuration defined in Appendix A.4. Within each dataset, it shares the interaction sequences, item representations, frozen user representations for guidance, generator architecture, and training settings with the other generative variants in Table 2. Complete unique Semantic IDs are used in both the discrete history and the training targets. Following the disambiguation scheme in TIGER (Rajput et al., 2023), an additional token is appended to each base ID so that distinct items sharing the same base ID receive distinct complete identifiers. We group targets according to the number of catalog items associated with the base ID: single-item base IDs correspond to one item, whereas multi-item base IDs correspond to more than one. Every complete identifier contains three base ID tokens followed by one disambiguation token. During beam search, we record *base ID success* when the correct base ID remains in the beam after the third identifier token, and *exact completion* when the complete unique Semantic ID of the target item remains in the beam after the fourth. Both events are recorded from the same decoding run for each target, with the two groups evaluated at the same respective decoding depths. We compute exact completion rates conditioned on base ID success using the definitions in Appendix B.1. The main diagnostic uses beam width $b = 20$, and we evaluate $b \in \{10, 20, 50, 100\}$ to examine the conditional completion gap across beam widths. At $b = 20$, targets with multi-item base ID success followed by exact completion failure form the subset used for the recovery analysis in Section 4.4. The additional diagnostic on the reproduced TIGER baseline is reported separately in Appendix B.5.

B EXACT ITEM GENERATION BOTTLENECK

The analyses in Sections B.1–B.4 use the *Unique* configuration with the diagnostic setup in Appendix A.5. Appendix B.5 reports the same diagnostic for the reproduced TIGER baseline used in the main comparison.

B.1 DEFINITIONS AND METRICS

Let M and S denote the events that the target has a multi-item or single-item base ID, respectively. Let B denote *base ID success*, meaning that the correct base ID remains in the beam after its three tokens, and let E denote *exact completion*, meaning that the complete unique ID of the target remains after the disambiguation token. All probabilities below are empirical frequencies over evaluation cases. We define

$$R_B = \Pr(B \mid M), \qquad R_E = \Pr(E \mid M),$$
$$C_M = \Pr(E \mid B, M), \qquad C_S = \Pr(E \mid B, S).$$

The rates R_B and R_E measure base ID success and exact completion among all cases with multi-item base IDs. The rates C_M and C_S measure exact completion among cases with base ID success in the multi-item and single-item groups, respectively. The conditional completion gap is $\Delta C = C_S - C_M$, so a positive gap indicates a lower conditional completion rate for the multi-item group. Because both events are recorded along the same decoding run, exact completion requires prior base ID success and $E \subseteq B$. Consequently,

$$R_E = \Pr(B \mid M) \Pr(E \mid B, M) = R_B C_M,$$
$$R_B - R_E = R_B(1 - C_M) = \Pr(B \cap E^c \mid M).$$

Thus, $R_B - R_E$ measures the fraction of multi-item cases in which the correct base ID is retained but exact completion fails. The tables and figures below display rates as percentages and differences between rates in percentage points.

B.2 Base ID Success and Exact Completion

At $b = 20$, Table 6 reports base ID success and exact completion in *Unique* for cases with multi-item base IDs. Targets gives the number of evaluated cases, while $\#B$ and $\#E$ count cases satisfying the corresponding events. Across the four datasets, 337–712 cases retain the correct base ID but fail exact completion, giving $R_B - R_E = 6.88$–8.63 percentage points. These failure cases form the subset evaluated in the recovery analysis in Section 4.4.

Table 6: Base ID success and exact completion for multi-item base IDs at $b = 20$. Failures is $\#B - \#E$; $R_B - R_E$ is computed from unrounded rates and reported in percentage points.

Dataset	Targets	$\#B$	$\#E$	Failures	R_B	R_E	$R_B - R_E$
Beauty	5,461	1,168	731	437	21.39	13.39	8.00
Toys	3,903	974	637	337	24.96	16.32	8.63
Sports	10,355	1,570	858	712	15.16	8.29	6.88
Yelp	7,251	1,174	641	533	16.19	8.84	7.35

Among cases with base ID success, Table 7 compares conditional exact completion for multi-item and single-item base IDs. The rates are 54.60–65.40% for the multi-item group and 97.00–99.68% for the single-item group, giving $\Delta C = 32.25$–45.08 percentage points. Equivalently, 34.60–45.40% of the multi-item cases with base ID success fail exact completion. Section B.3 compares this transition with the preceding transitions within the base ID.

Table 7: Exact completion conditioned on base ID success at $b = 20$. C_M and C_S are percentages; $\Delta C = C_S - C_M$ is computed from unrounded rates and reported in percentage points.

	Multi-item			Single-item			
Dataset	$\#B$	$\#E$	C_M	$\#B$	$\#E$	C_S	ΔC
Beauty	1,168	731	62.59	1,569	1,522	97.00	34.42
Toys	974	637	65.40	1,317	1,286	97.65	32.25
Sports	1,570	858	54.65	1,150	1,140	99.13	44.48
Yelp	1,174	641	54.60	1,262	1,258	99.68	45.08

B.3 Analysis Across Decoding Stages

To compare exact completion with the preceding base ID transitions, let B_ℓ denote the event that the correct base ID prefix remains in the beam after ℓ identifier tokens. The base ID contains three tokens, so $B = B_3$. For each group $G \in \{M, S\}$, we measure

$$\Pr(B_2 \mid B_1, G), \qquad \Pr(B_3 \mid B_2, G), \qquad \Pr(E \mid B_3, G).$$

The final transition gives $\Pr(E \mid B_3, M) = C_M$ and $\Pr(E \mid B_3, S) = C_S$. Figure 3 reports the three conditional transition rates at $b = 20$. Across all four datasets, the multi-item group has

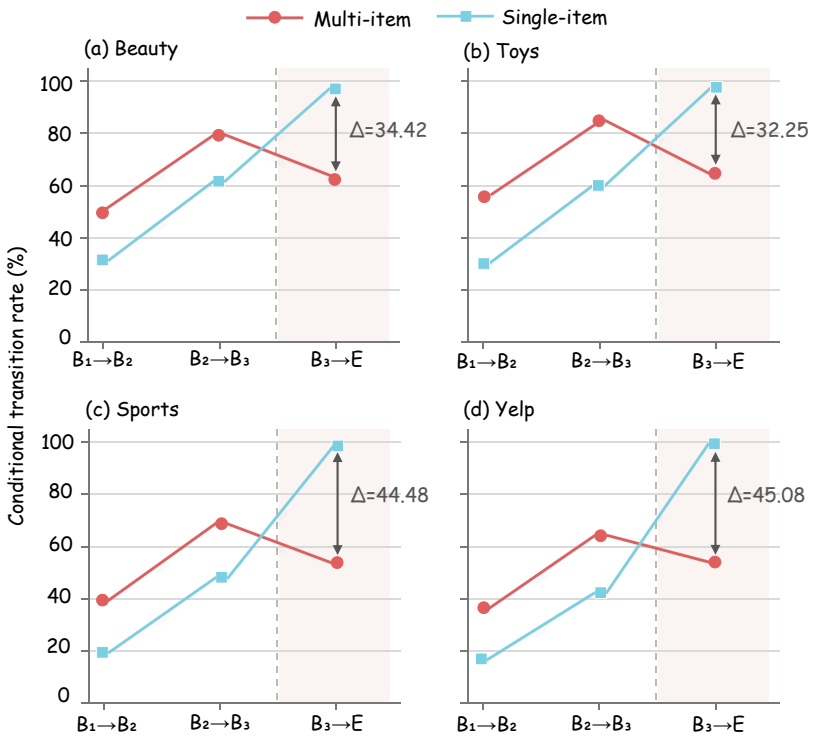

Figure 3: Conditional transition rates in *Unique* at $b = 20$. Each rate is conditioned on retaining the correct prefix at the preceding stage. The first two transitions extend the base ID, and $B_3 \to E$ completes the unique identifier. Panels (a)–(d) correspond to Beauty, Toys, Sports, and Yelp.

higher conditional transition rates by 17.04–25.34 percentage points at $B_1 \to B_2$ and 18.40–25.31 percentage points at $B_2 \to B_3$. At $B_3 \to E$, the ordering reverses: the multi-item rate is lower by 32.25–45.08 percentage points. The lower conditional retention of the multi-item group therefore appears at exact completion rather than in the two preceding base ID transitions.

B.4 Effect of Beam Width on the Bottleneck

We evaluate $b \in \{10, 20, 50, 100\}$ to examine whether the conditional completion gap persists as the beam widens. Figure 4 reports the conditional completion rates for multi-item and single-item base IDs across beam widths. The conditional completion gap remains positive in all 16 dataset–beam combinations, with ΔC ranging from 22.93 to 47.17 percentage points. At $b = 100$, the gap decreases to 26.71 and 22.93 percentage points on Beauty and Toys, respectively. On Sports and Yelp, it remains at least 37.04 and 44.20 percentage points, respectively, throughout the tested range. We next examine whether the transition-level reversal observed at $b = 20$ also persists across beam widths. Table 8 reports, for each decoding transition, the multi-item rate minus the single-item rate. Across all 16 settings, the multi-item rate exceeds the single-item rate by 15.68–28.82 percentage points at $B_1 \to B_2$ and by 11.16–29.61 percentage points at $B_2 \to B_3$. The ordering reverses at $B_3 \to E$, where the multi-item rate is lower by 22.93–47.17 percentage points. Thus, the transition-level reversal observed at $b = 20$ persists across all tested beam widths, although the magnitude of the conditional completion gap varies with b.

B.5 Bottleneck Diagnostic in TIGER

The preceding analyses use the controlled *Unique* diagnostic described in Appendix A.5. To examine whether the observed completion disparity is specific to this controlled configuration, we apply the same event definitions and metrics to the TIGER baseline used in the main comparison (Rajput et al., 2023). TIGER retains its own identifier construction, model architecture, and training proce-

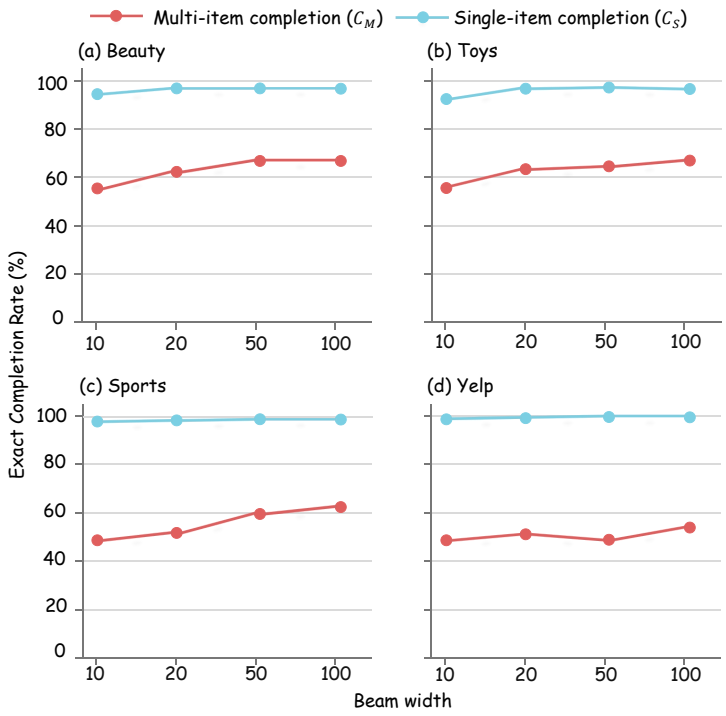

Figure 4: Exact completion conditioned on base ID success across beam widths. C_M and C_S denote the conditional completion rates for multi-item and single-item base IDs, respectively.

Table 8: Multi-item minus single-item conditional transition rates across beam widths, in percentage points. The final column equals $-\Delta C$; † marks $b = 20$.

Dataset	b	$B_1 \to B_2$	$B_2 \to B_3$	$B_3 \to E$
Beauty	10	+18.71	+24.02	-39.23
	20†	+17.40	+18.40	-34.42
	50	+16.79	+12.56	-29.50
	100	+15.68	+11.16	-26.71
Toys	10	+28.82	+29.61	-37.15
	20†	+25.34	+25.31	-32.25
	50	+22.66	+17.85	-28.04
	100	+20.93	+14.75	-22.93
Sports	10	+19.56	+24.42	-45.00
	20†	+20.18	+22.00	-44.48
	50	+20.78	+20.24	-39.75
	100	+21.72	+20.04	-37.04
Yelp	10	+16.47	+20.84	-44.20
	20†	+17.04	+22.12	-45.08
	50	+19.13	+21.14	-47.17
	100	+20.65	+17.70	-46.34

dure and is evaluated under the common data and evaluation protocol. For TIGER, the base ID is the three-token quantization code sequence preceding the disambiguation token. We use the definitions and rates from Appendix B.1 and evaluate the diagnostic at beam width $b = 20$. For targets with multi-item base IDs, $R_B - R_E$ ranges from 7.70 to 9.98 percentage points across the four datasets. Conditioned on base ID success, exact completion reaches 46.47–58.88% for multi-item base IDs and 96.65–99.15% for single-item base IDs, yielding $\Delta C = 38.46$–52.68 percentage points. The conditional completion gap is positive on every dataset, matching the direction observed in the con-

trolled *Unique* diagnostic. The TIGER diagnostic therefore provides independent evidence that the exact item generation bottleneck is not specific to the controlled *Unique* configuration.

Table 9: Bottleneck diagnostic for TIGER at $b = 20$, using the definitions in Appendix B.1. Rates are percentages and differences are percentage points.

Dataset	R_B	R_E	$R_B - R_E$	C_M	C_S	ΔC
Beauty	20.57	11.46	9.11	55.71	96.65	40.94
Toys	24.27	14.29	9.98	58.88	97.34	38.46
Sports	14.76	7.06	7.70	47.85	98.64	50.79
Yelp	15.73	7.31	8.42	46.47	99.15	52.68

C Effect of Beam Width

With $C = 4$, we evaluate RELAX at $b \in \{10, 20, 50\}$ using the same trained generator, frozen continuous recommender, and Shared ID hierarchy within each dataset. At each beam width, we report Candidate Recall, average candidate set size, ranking performance, and inference latency (Table 10). Increasing b from 10 to 20 improves R@10 by 0.0042–0.0091 across the four datasets. Increasing b further from 20 to 50 increases the average candidate set size by a factor of 2.41–2.55 and inference latency by a factor of 1.87–1.88, with an additional R@10 gain of 0.0024–0.0085. The incremental R@10 gain from $b = 20$ to $b = 50$ is smaller than that from $b = 10$ to $b = 20$ on all four datasets. The default beam width is $b = 20$; together with the capacity analysis in Section 4.5, the default configuration is $(C, b) = (4, 20)$.

For comparison, RELAX at $b = 10$ achieves higher mean R@10 and N@10 than *Unique* + *Scorer* at $b = 20$ on Toys and Sports (Table 2). The average candidate counts for RELAX and *Unique* + *Scorer* are 19.20 and 18.85 on Toys, and 19.40 and 19.12 on Sports, respectively.

Table 10: Effect of beam width with Shared ID capacity fixed at $C = 4$. Latency is end-to-end inference time in milliseconds; † marks the default beam width $b = 20$.

		Candidate set			Final ranking			
Dataset	b	Cand. Recall	Avg. Cand.	Latency (ms)	R@5	N@5	R@10	N@10
Beauty	10	0.1124	21.00	0.660	0.0702	0.0511	0.0943	0.0589
	20†	0.1571	39.92	0.891	0.0718	0.0514	0.1000	0.0605
	50	0.2329	101.67	1.676	0.0752	0.0535	0.1025	0.0623
Toys	10	0.1068	19.20	0.650	0.0741	0.0548	0.0949	0.0615
	20†	0.1479	37.94	0.881	0.0777	0.0571	0.1040	0.0656
	50	0.2214	94.26	1.644	0.0836	0.0605	0.1125	0.0698
Sports	10	0.0638	19.40	0.660	0.0384	0.0271	0.0531	0.0318
	20†	0.0908	38.81	0.889	0.0409	0.0283	0.0573	0.0336
	50	0.1501	99.03	1.665	0.0420	0.0291	0.0607	0.0351
Yelp	10	0.0808	27.50	0.650	0.0411	0.0302	0.0612	0.0367
	20†	0.1170	54.11	0.884	0.0437	0.0326	0.0662	0.0399
	50	0.2056	130.28	1.656	0.0474	0.0357	0.0686	0.0425

D Shared ID Structure

We examine the Shared IDs constructed by RELAX under the default capacity $C = 4$. For a Shared ID S, let $\mathcal{B}(S)$ denote its associated item bucket. Across the four datasets, multi-item Shared IDs constitute 25.40–34.86% of the constructed Shared IDs and represent 45.48–59.39% of items. Figure 5 further reports the distributions of bucket size and Shared ID length. Figure 5(a) shows that 65.14–74.60% of Shared IDs represent one item and 16.73–18.95% represent two, while 2.78–9.62% reach capacity $C = 4$. No bucket exceeds the capacity bound. The length distributions differ across datasets (Figure 5(b)): 73.11–81.68% of Shared IDs on Beauty, Toys, and Sports have length

two, whereas 97.31% on Yelp retain length three. These lengths reflect the first depth at which the capacity bound is satisfied. A Shared ID of length L can still represent multiple items; retaining the complete path does not imply unique item identification.

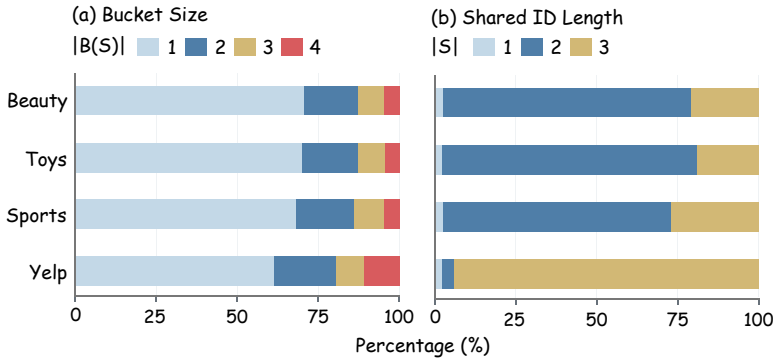

Figure 5: Shared ID structure at $C = 4$. (a) Distribution of bucket size $|\mathcal{B}(S)|$. (b) Distribution of Shared ID length $|S|$ after prefix pruning. Percentages are computed over distinct Shared IDs. The full hierarchy depth is $L = 3$.

E Guidance and Item Representation Ablations

We examine continuous user guidance to the generator and the item representation used for Shared ID construction. Both ablations use $C = 4$ and $b = 20$, with the remaining hyperparameters and evaluation settings fixed. For the guidance comparison, item representations are fixed to *Fused*; *No Guidance* removes the projected continuous user representation while retaining the Shared ID history. For the representation comparison, continuous user guidance is retained. *Text*, *Collaborative*, and *Fused* use $\mathbf{a}_i$, $\mathbf{e}_i$, and $\mathbf{v}_i = \mathbf{a}_i + \mathbf{e}_i$, respectively, to construct Shared IDs. RELAX is retrained with each resulting hierarchy.

Table 11: Guidance and item-representation ablations at $C = 4$ and $b = 20$. The guidance block uses Fused representations; the representation block retains continuous user guidance. Best results within each block are bold.

	Beauty		Toys		Sports		Yelp	
Variant	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
Continuous user guidance								
No Guidance	0.0928	0.0560	0.0935	0.0588	0.0514	0.0296	0.0592	0.0352
Guidance	**0.1000**	**0.0605**	**0.1040**	**0.0656**	**0.0573**	**0.0336**	**0.0662**	**0.0399**
Item representation								
Text	0.0843	0.0513	0.0854	0.0556	0.0445	0.0261	0.0506	0.0317
Collaborative	0.0791	0.0475	0.0825	0.0471	0.0397	0.0218	0.0351	0.0189
Fused	**0.1000**	**0.0605**	**0.1040**	**0.0656**	**0.0573**	**0.0336**	**0.0662**	**0.0399**

Continuous user guidance improves R@10 by 0.0059–0.0105 and N@10 by 0.0040–0.0068 across the four datasets. For item representation, *Fused* consistently outperforms *Text*, which in turn outperforms *Collaborative*, on both metrics across all datasets. Relative to *Text*, *Fused* improves R@10 by 0.0128–0.0186 and N@10 by 0.0075–0.0100.