

Beyond Semantic IDs: Finite-Beam Reachability and Continuous-Query Generation in Generative Recommendation

Yu Sun* Renkai Xiang Zuodong Xiang Junwei Zhou Yike Zhang
Jianxin Zhang Zhihao Zhu Xiaoyu Cao Hailu Xu

Abstract

Generative recommendation (GR) of social media platforms commonly retrieves items by autoregressively decoding discrete Semantic IDs (SIDs) with finite-width beam search. This work examines how this output interface constrains whether held-out targets enter finite returned lists, while distinguishing catalog-wide scoreability from empirical top- K support. We introduce two continuous-query paradigms: CQG-Single generates one retrieval query in a single pass, and CQG-AR autoregressively generates continuous residual updates, retaining step-wise decoding while removing the SID item-token vocabulary, code lookup, and discrete beam expansion. These designs thereby eliminate the SID-specific reachability constraint induced by finite-beam decoding and avoid the technical challenge of aligning a discrete SID vocabulary with the pretrained backbones’ vector space. Both methods make every encoded item directly scoreable through dense or approximate-nearest-neighbor retrieval. Under a unified protocol with valid-prefix constrained decoding, CQG-Single exceeds the TIGER-style baseline in all seven runs while using 55.3% as many trainable parameters. CQG-AR remains comparable on ML-1M and is numerically stronger on Beauty and Sports datasets. A controlled depth ablation further shows that additional continuous refinement does not monotonically improve shallow accuracy. These findings motivate decoupling output representation, autoregressive modeling, and retrieval budget when evaluating generative recommendation systems.

1 Introduction

Generative recommendation (GR) retrieves items by generating symbolic identifiers rather than directly scoring the entire catalog [Li et al., 2024]. The most influential instance of this paradigm is Semantic-ID (SID) recommendation: items are first quantized into discrete code sequences, and an autoregressive model decodes those sequences conditioned on user attributes and interaction history [Rajput et al., 2023, Zheng et al., 2024, Wang et al., 2024]. This formulation converts recommendation into a sequence-generation problem, and supports constrained decoding through a trie, ensuring that the generated token sequence corresponds to a valid item.

This marks a substantial architectural shift from conventional sequential recommenders. Models such as SASRec encode user histories, user features, and item representations, then score candidate items discriminatively [Kang and McAuley, 2018]. In SID-based generative recommendation, the output interface is instead moved to a language-style generator that is trained to emit the SID sequence. Larger pretrained language models can provide useful generic knowledge and may benefit from scaling behavior as the model size increases [Kaplan et al., 2020]. However, SID-based GR systems usually require three components: First, items must be compressed into a finite hierarchical identifier space, using vector-quantization methods such as VQ-VAE [Van Den Oord et al., 2017], residual quantization as in RQ-VAE [Lee et al., 2022], residual or balanced k-means hierarchical clustering, or industrial generative retrieval designs such as OneRec [Deng et al., 2025]. Second, the generator needs to be trained either by adapting a pretrained LLM through continued pretraining (CPT) and supervised fine-tuning (SFT), where SID tokens become a new vocabulary that must be aligned with the pretrained model’s semantic space, or by training a smaller decoder-only model from scratch, which sacrifices pretrained generalization for better SID prediction precision. Third, for inference, this generative

*Corresponding author

model decodes SID sequences from user features and sequential history, then maps the generated codes to item IDs or expands them through item-to-item retrieval.

These steps introduce two practical bottlenecks. 1. Coverage and updatability: a catalog item is useful only if its SID can be generated and mapped back to the item, but newly created items may cause semantic drift between the static clustering model labeling and the item meaning. 2. Support imbalance: a finite hierarchical SID space is a compromise for representing an open-ended catalog. Items are not distributed uniformly across SID leaves, longtail items have worse reachability, and the generated SID set may fail to recall enough valid items even when the related item catalog itself is large.

Because SID inference explores only a prefix-limited subset of the code tree, we define a *checkpoint- and budget-specific finite-beam reachability ceiling* as the target-support upper bound induced by a fixed trained model and finite beam budget, even when the held-out item has a valid code.

This support limitation motivates examining seed-stable gains and *Popularity-Dependent Crossover*. CQG-Single achieves a stable aggregate R@10 gain across seven training seeds, and it is numerically stronger than TIGER-style SID in 38 of 48 popularity–depth comparisons. We also find that TIGER and CQG exhibit different advantages across rare versus popular items and at shallow retrieval depths. Rather than claiming a universal pattern, we characterize the crossover as dependent on the checkpoint, item popularity, and retrieval depth.

To isolate this mechanism, we study two complementary continuous-query formulations. **CQG-Single** is a deliberately minimal vocabulary-free model that maps user context to one continuous query in a frozen item space. **CQG-AR** retains an autoregressive decoder but replaces SID-token prediction with continuous residual generation. Both retrieve by dense embedding scoring, and approximate nearest-neighbor (ANN) search without a code-to-item lookup. CQG-Single tests the simplest continuous interface, while CQG-AR controls for autoregressive model structure.

Our contributions are:

- **Finite-search diagnostics.** For a fixed checkpoint and search budget, we distinguish unique-code assignment, target-conditioned beam support, catalog-wide scoreability, and empirical union-of-top- K coverage, and stratify the output-based quantities by item popularity.
- **Conditional crossover pattern.** In some popularity strata and evaluated checkpoints, SID decoding can achieve higher shallow Recall@ K , while continuous retrieval can overtake at deeper K under a matched returned-list size.
- **Parallel continuous-query methods.** We introduce CQG-Single, a one-step continuous retriever, and CQG-AR, an autoregressive continuous decoder. Their comparison separates the effect of autoregression from the effect of a discrete SID vocabulary.
- **Controlled empirical study.** Across ML-1M, Amazon Beauty, and Amazon Sports, we analyze retrieval quality, finite-output coverage, objective controls, and LLM-backbone diagnostics while limiting scalability claims to the tested catalog sizes.

2 Related Work

Semantic-ID generative recommendation. TIGER [Rajput et al., 2023] introduced hierarchical Semantic IDs using residual vector quantization and autoregressive decoding. Later work improves identifier learning and alignment through collaborative semantics, textual IDs, differentiable indexing, or soft identifiers [Zheng et al., 2024, Wang et al., 2024, Tan et al., 2024, Fu et al., 2026, Li et al., 2026]. These methods differ in code construction but retain a structured output interface that typically uses constrained autoregressive decoding. We study how finite-width search allocates support over that interface, rather than proposing another discrete tokenizer. This distinction also separates successful code assignment from successful code retrieval at inference time.

Finite search support and popularity. Autoregressive tree generation can impose correlation constraints on representable item distributions [Hou et al., 2026], while training-side studies note that tail tokens receive fewer updates. We examine the complementary inference-side question: even with a valid, nearly

unique code, does a target enter a finite beam? We measure target-conditioned beam support, returned-list-size-matched catalog coverage, and popularity-stratified behavior, carefully separating these quantities. In particular, a high unique-code rate does not imply that every target enters a user’s beam, and a target-conditioned hit rate does not equal catalog coverage. Our experiments treat these as distinct diagnostics of a trained model and its search budget.

Vocabulary-free and continuous recommendation. DiffuRec, DreamRec, and ContRec use diffusion or continuous latent generation [Li et al., 2023b, Yang et al., 2023, Qu et al., 2026], while GPT4Rec generates textual retrieval queries [Li et al., 2023a]. CQG-Single provides a minimal continuous-query control, and CQG-AR adds step-wise residual refinement. Their purpose is to separate autoregression from discrete SID output, not to claim dense retrieval itself as novel. This paired design asks whether step-wise generation requires discrete identifier tokens: CQG-Single removes both the token vocabulary and serial decoding, whereas CQG-AR preserves serial generation but replaces tokens with continuous residual updates.

Adjacent efficient and multimodal model applications. Resource-efficient language modeling and multimodal adaptation have been investigated in related application settings. Energy-efficient retrieval-augmented generation with small language models addresses on-device constraints [Cheng et al., 2026]. In short-video governance, in-prompt process supervision [Liu et al., 2026] and reasoning-enhanced domain-adaptive multimodal pretraining [Wang et al., 2025] study different supervision and adaptation strategies. Audio-enhanced vision–language modeling with latent-space broadening explores multimodal representation and data expansion [Sun et al., 2025]. These studies provide broader context, but do not establish results on finite-beam Semantic-ID reachability or the continuous-query recommendation methods analyzed here.

Sequential, dense, and hybrid retrieval. GRU4Rec, SASRec, and BERT4Rec score catalog items from interaction histories [Hidasi et al., 2015, Kang and McAuley, 2018, Sun et al., 2019]; dual encoders are established in retrieval and recommendation [Huang et al., 2013, Covington et al., 2016]. Generative retrieval instead decodes identifiers [Tay et al., 2022, Wang et al., 2022], and LIGER combines dense and generative retrieval [Yang et al., 2024]. These lines establish both catalog scoring and identifier decoding as known retrieval interfaces. Our contribution is therefore a controlled comparison in a shared frozen item space, using CQG-Single and CQG-AR to distinguish finite SID output support from autoregressive modeling without attributing the comparison to a larger item encoder or an additional reranker.

3 Method

3.1 Problem Formulation

Let $\mathcal{I}$ be a catalog of N items and let user u have a chronological interaction sequence $s_u = (i_1, \dots, i_t)$. The task is to retrieve the next item i_{t+1} . SID-based GR maps each item i to a discrete code sequence $z_i = (z_{i,1}, \dots, z_{i,L})$ and trains an autoregressive model $p(z_i | s_u)$. Inference uses beam search over valid code prefixes.

Both of our continuous methods assume a frozen item embedding matrix $E \in \mathbb{R}^{N \times d}$, where each row e_i is an L2-normalized item vector. They condition on the same user context c_u available to the SID decoder, including the sequential item history and, when available, demographic features. They differ in whether the continuous output is produced in one step or autoregressively.

3.2 CQG-Single: Single-Step Continuous Query Generation

CQG-Single learns a single-step query generator f_θ :

$$q_u = f_\theta(c_u), \quad q_u \in \mathbb{R}^d. \quad (1)$$

The recommendation score is cosine similarity:

$$\text{score}(u, i) = q_u^\top e_i. \quad (2)$$

The top- K items are retrieved by exact dense scoring or ANN search. Every item in E is scoreable by construction, although catalog-wide scoreability does not guarantee that every item will empirically appear in a finite top- K list.

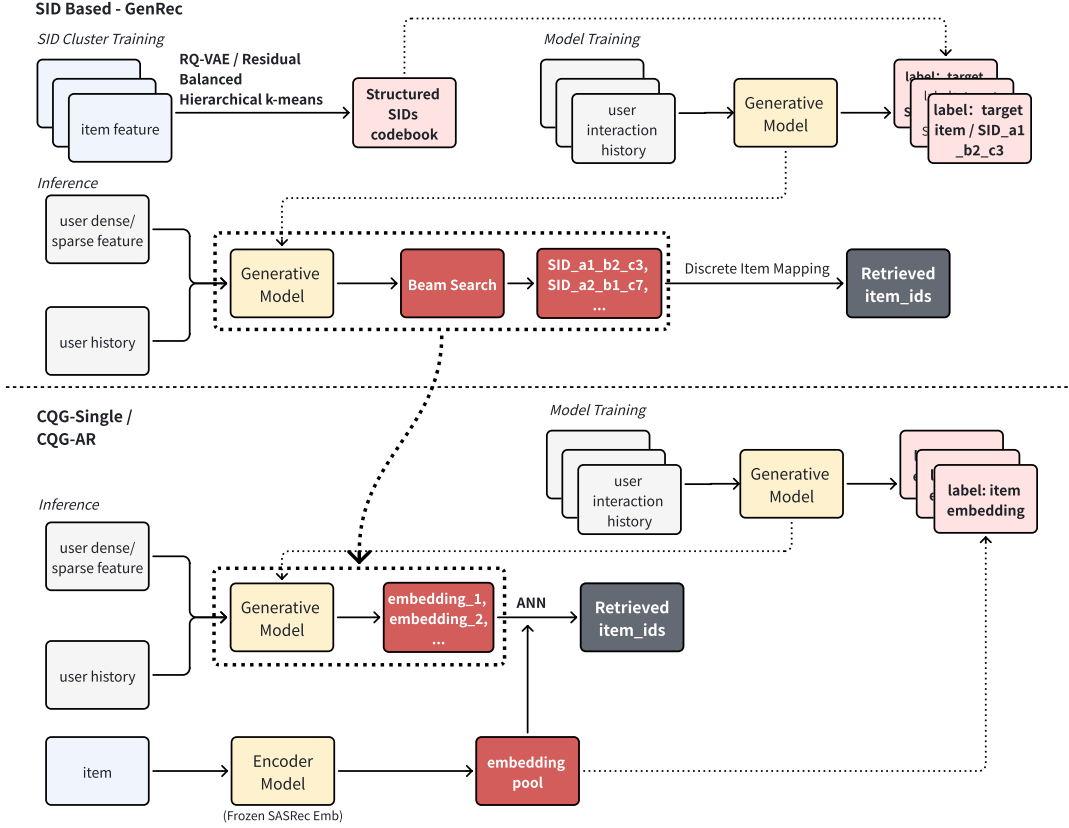


Figure 1: Training and inference pipelines for SID retrieval, CQG-Single, and CQG-AR. SID-based GR is trained to predict clustered code sequences, whereas CQG generates continuous queries in item-embedding space. SID retrieval autoregressively generates discrete code tokens and explores catalog-valid paths with finite-prefix beam search. CQG-Single produces one continuous query in a single forward pass, whereas CQG-AR autoregressively generates residual updates that refine a continuous query. Both CQG variants retrieve from the same frozen item embedding index by dense scoring or ANN, without an item-token vocabulary or code-to-item lookup.

The query generator is a causal Transformer that maps the final valid history state back into the shared item space. Unlike SID decoding, it produces the retrieval query in one forward pass without an item-token vocabulary.

3.3 CQG-AR: Autoregressive Continuous Query Generation

CQG-AR retains an autoregressive decoder but replaces discrete SID-vocabulary prediction with continuous residual generation. At decoding step ℓ , the decoder conditions on the user context and all previous continuous outputs:

$$h_\ell = D_\phi(c_u, \Delta_{<\ell}), \quad (3)$$

$$\Delta_\ell = W_{\text{AR}} h_\ell, \quad (4)$$

$$\hat{e}_u^{(\ell)} = \text{norm}(\hat{e}_u^{(\ell-1)} + \Delta_\ell). \quad (5)$$

We initialize the cumulative query as $\hat{e}_u^{(0)} = \mathbf{0}$. The generated residual Δ_ℓ is projected back into the decoder input space and appended as the next autoregressive input. After L steps, the refined query $\hat{e}_u^{(L)}$ retrieves

items from the same frozen embedding matrix E . Thus CQG-AR preserves the conditional, step-wise decoding structure of SID retrieval while removing the SID vocabulary, codebook lookup, and discrete beam expansion.

3.4 Training Objectives

We evaluate four CQG objectives: full-catalog CE, sampled CE, in-batch InfoNCE, and MSE. For CQG-Single, q_b below is the single generated query; for CQG-AR, it is the final refinement $\hat{e}_b^{(L)}$. In the primary CQG-AR configuration, only the final query receives direct CE supervision; intermediate supervision is used only in the decoding-depth ablation.

Full-catalog cross-entropy. Our primary objective treats retrieval as classification over all items:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{B} \sum_{b=1}^B \log \frac{\exp(q_b^\top e_b^+ / \tau)}{\sum_{j=1}^N \exp(q_b^\top e_j / \tau)}, \quad (6)$$

where $\tau = 0.07$. CE gives complete catalog visibility at training time but costs $O(N)$ per step.

Sampled cross-entropy. To control training-time catalog visibility, sampled CE replaces the full denominator with the positive item and a sampled set $\mathcal{S}_b$ of negatives:

$$\mathcal{L}_{\text{sCE}} = -\frac{1}{B} \sum_{b=1}^B \log \frac{\exp(q_b^\top e_b^+ / \tau)}{\exp(q_b^\top e_b^+ / \tau) + \sum_{j \in \mathcal{S}_b} \exp(q_b^\top e_j / \tau)}. \quad (7)$$

The reported control uses 256 sampled negatives.

In-batch InfoNCE. InfoNCE treats the other target items in the minibatch as negatives:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{B} \sum_{b=1}^B \log \frac{\exp(q_b^\top e_b^+ / \tau)}{\sum_{j \in \mathcal{B}} \exp(q_b^\top e_j^+ / \tau)}. \quad (8)$$

Mean-squared error. We also test direct embedding regression:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{B} \sum_{b=1}^B \|q_b - e_b^+\|_2^2. \quad (9)$$

MSE preserves geometric alignment but yields lower retrieval accuracy than CE in our experiments. BPR is used to pretrain the shared SASRec item encoder, not as one of the reported CQG objectives.

3.5 Retrieval and Contrast with SID

Standalone CQG-Single ranks items according to $q_u^\top e_i$, while CQG-AR uses $(\hat{e}_u^{(L)})^\top e_i$. Exact catalog scoring can be replaced by ANN search, such as FAISS, for large catalogs. Our CQG-AR experiments retrieve only with the final query $\hat{e}_u^{(L)}$; merging candidates from intermediate or interest-specific queries is an untested extension. In contrast, SID retrieval requires codebook learning, autoregressive discrete decoding, beam search, and code-to-item lookup; a finite beam explores only a restricted subset of valid code paths. CQG-Single instead generates one continuous query, while CQG-AR preserves step-wise autoregressive generation with continuous outputs. Both directly query the item embedding index without introducing an item-token vocabulary. (Figure 1)

4 Experiments

4.1 Experimental Setup

Datasets. We use MovieLens-1M (ML-1M), Amazon Beauty, and Amazon Sports. ML-1M is the primary controlled diagnostic because its catalog is small enough for exact full-catalog scoring. Beauty and Sports test cross-domain behavior. Our processed Sports split (21,519 users and 10,693 non-padding items) differs from

Dataset	Users	Items	Interactions	Density
ML-1M	6,040	3,416	999K	4.84%
Beauty	22,363	12,101	198K	0.07%
Sports	21,519	10,693	~178K	0.08%

Table 1: Dataset statistics. Item counts exclude the padding row. Density is defined as $|\mathcal{R}|/(|\mathcal{U}||\mathcal{I}|)$, where $\mathcal{R}$ is the set of observed interactions, $\mathcal{U}$ the users, and $\mathcal{I}$ the non-padding items. Higher density indicates that users interact with a larger fraction of the catalog, whereas lower density indicates sparser interactions and a more pronounced long tail. Bold marks the largest value in each numerical column.

Method	User-weighted TargetInOutput@100	Item-balanced TargetInOutput@100				TargetItemSupport @200
	Overall	Q1 rare	Q2	Q3	Q4 popular	Distinct targets
TIGER	66.7%	25.8%	45.2%	54.0%	68.6%	78.1%
CQG-Single	67.3%	26.4%	46.9%	56.3%	70.0%	78.9%
CQG-AR	66.4%	24.4%	45.4%	54.8%	69.6%	78.8%

Table 2: ML-1M finite-output support (seed 42). Overall is user weighted; Q1–Q4 average distinct target items within training-frequency quartiles, and the final column is the fraction of distinct targets retrieved at least once by Top-200. Returned-list sizes and filtering are matched across methods. Bold denotes the best value in each column.

the commonly reported 5-core split used by TIGER, LETTER, and EAGER; consequently, we do not directly compare our Sports values with their published numbers. (We will use “TIGER” to denote our hierarchical implementation of k means in TIGER-style.)

Protocol. We use leave-one-out evaluation: the last item is held out for testing, the second-to-last item for validation, and earlier interactions for training. Historical items are excluded from ranking. We report Recall@K and NDCG@K, primarily for $K \in \{10, 30, 50, 70, 90\}$ in reachability analysis and $K \in \{5, 10, 20\}$ in cross-dataset evaluation.

Baselines. We compare against SASRec [Kang and McAuley, 2018], TIGER-style SID retrieval, and reported SID results from TIGER, LETTER, and EAGER where protocols allow contextual comparison. We also include non-learned dense retrieval baselines: random retrieval, last-item kNN, and mean-of-5 kNN.

Model training protocol. We first train a two-layer SASRec with BPR on chronological next-item pairs and select its checkpoint by full-catalog validation NDCG@10. Its 64-dimensional item embedding table is then frozen and shared by both continuous methods as the history-input table, target embedding or classifier matrix, and retrieval index. The main experiments use no textual or language-model item encoder.

CQG-Single is a four-layer, four-head causal Transformer with width 256. It directly generates one normalized query; the final configuration uses full-catalog CE, with MSE and in-batch InfoNCE evaluated as ablations. **CQG-AR** uses a four-layer, six-head causal Transformer with width 384 to autoregressively generate three residual query updates. Each residual is fed into the next decoding step, while the cumulative query is updated by adding the residual and then normalizing the result. In the final configuration, only the final query receives direct full-catalog CE supervision. The multimodal Sports diagnostic instead uses a frozen fusion of collaborative and ImageNet-pretrained ResNet-18 features. Full implementation details are provided in Appendix D.

5 Results

5.1 Budget-Specific Finite-Beam Reachability

For a fixed model and returned-list size B , target support is the fraction of test users whose held-out target occurs anywhere in the returned list. On ML-1M, the seed-42 TargetInOutput@100 values are 66.7% for TIGER, 67.3% for CQG-Single, and 66.4% for CQG-AR. This is a per-user quantity closely related to Recall@100. Figure 2 reports the matched popularity breakdown and user counts.

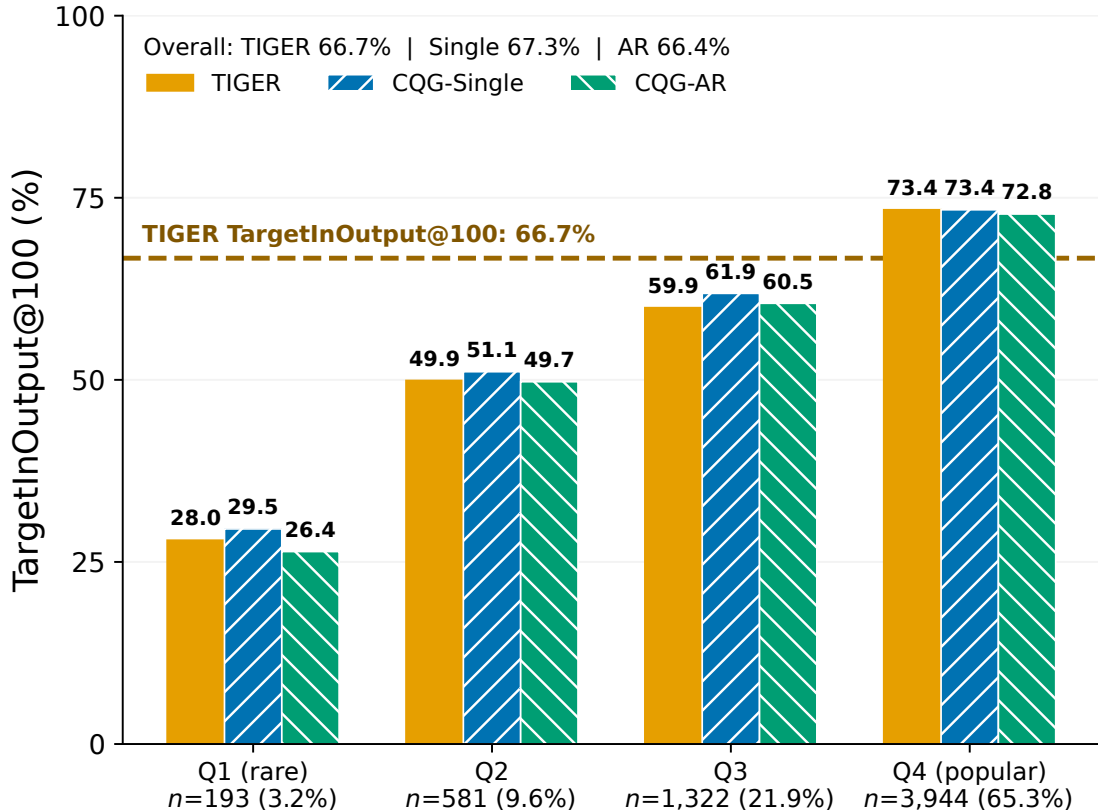


Figure 2: ML-1M TargetInOutput@100 by target-popularity quartile (seed 42). Finite returned lists exhibit strongly popularity-skewed target support, while CQG-Single matches or exceeds TIGER across all popularity quartiles at the reported precision. All methods produce matched Top-200 lists; TargetInOutput@100 is evaluated on their first 100 positions; TIGER uses valid-prefix constrained decoding. Bars are user weighted, whereas Table 2 reports item-balanced quartile values.

Method	Config.	Trainable Params	ML-1M R@10	ML-1M NDCG@10	Beauty R@10	Sports R@10
SASRec	Full-rank	0.32 / 0.88 / 0.79M	0.2797	0.1535	0.0558	0.0390
TIGER	Discrete AR	5.78M	0.2996 ± 0.0072	0.1739 ± 0.0046	0.0477 ± 0.0042	0.0236 ± 0.0028
CQG-Single	One-step	3.2M	0.3104 ± 0.0047	0.1822 ± 0.0044	0.0605 ± 0.0046	0.0363 ± 0.0015
CQG-AR	AR continuous ($L = 3$)	5.62M	0.2976 ± 0.0032	0.1735 ± 0.0016	0.0571 ± 0.0043	0.0355 ± 0.0004

Table 3: Cross-dataset primary comparison. TIGER and CQG results are mean $\pm$ sample SD over seeds 42–48 on ML-1M and seeds 42–44 on Beauty and Sports; SASRec uses one full-rank checkpoint. SASRec’s trainable parameter counts of 0.32/0.88/0.79M correspond to ML-1M, Beauty, and Sports, respectively, reflecting their different item-embedding table sizes. All methods use aligned full-user evaluation, and TIGER uses valid-prefix constrained decoding. Seed-level results and 95% t intervals are reported in Appendix B. Bold denotes the best value in each dataset/metric column.

For this fixed model-search pair, TargetSupport@ B is a realized ceiling on the Recall of any downstream reranker restricted to the decoded beam: a target that never enters the candidate set cannot be recovered later. This is a finite-beam reachability ceiling, not an architecture-invariant claim; increasing B can raise it. CQG avoids this particular beam-exclusion mechanism because every encoded item remains scoreable, although its finite Top- K Recall is still determined by query quality.

Table 2 provides the user-weighted and returned-list-size-matched finite-output comparison. TargetInOutput@100 is user weighted. Q1–Q4 first compute the hit rate of each distinct test-target item and then average

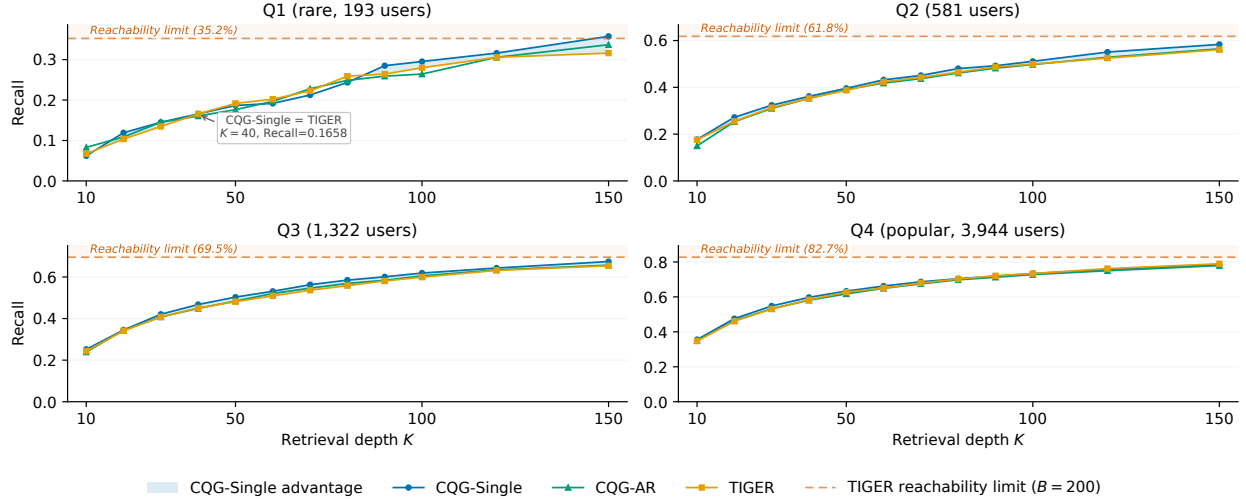


Figure 3: Measured per-quartile Recall@ K curves on ML-1M, seed 42. Dashed lines show finite-output target-support ceiling of TIGER’s user-weighted TargetSupport@200 within each quartile. All three methods return exactly 200 valid, distinct, history-filtered items per user. The X-axis represents retrieval depth K , the number of top- K items generated, while the Y-axis reports the fraction of users whose held-out target appears within the Top- K list. CQG-Single outperforms TIGER in 38 of the 48 popularity–depth comparisons, particularly on Q2–Q3. TIGER remains stronger for some rare Q1 targets at shallow depths. These results indicate that CQG’s advantage is affected by both target-item popularity and retrieval depth. This diagnostic uses a separately selected CQG-Single checkpoint (aggregate R@10 = 0.3063); the primary seed-42 checkpoint in Table 3 has R@10 = 0.3036. Full values for all 12 depths and paired-bootstrap intervals appear in Appendix C.1.

items equally within training-frequency quartiles. TargetItemSupport@200 is the fraction of distinct held-out target items retrieved for at least one associated test instance. CQG-Single retains advantage in overall user-weighted TargetInOutput@100 and in all four item-balanced quartiles.

Union-of-output catalog coverage is reported separately in Table 4. On Beauty, and more modestly at Top-100 on ML-1M, CQG achieves broader catalog coverage than TIGER, suggesting potential for improved retrieval and generalization to long-tail items. This result separates code assignment from target support: 99.4% unique codes do not imply that every held-out target enters a finite returned list. Under the corrected Top-100 protocol, 33.3% of held-out user targets do not enter the corresponding candidate list; this is not a statement that 33.3% of catalog items are never retrieved. The distinct union-of-output catalog-coverage diagnostic is reported in Table 4.

5.2 Seed-Stable Gains and Popularity-Dependent Crossover

Table 3 shows that CQG-Single outperforms TIGER-style SID in all seven ML-1M training runs, with a mean paired R@10 gain of 0.0107 (95% paired t interval [0.0021, 0.0193]; exact sign-flip $p = .0156$). For the separately selected seed-42 diagnostic checkpoint, Figure 3 shows numerically higher CQG-Single Recall in 38 of the 48 popularity–depth comparisons, including all 24 comparisons in Q2–Q3; the reversals are confined to Q1 and Q4. Full curve values and paired intervals are reported in Appendix C.1. The curve is checkpoint-specific and is not substituted for the primary seed-42 value in Table 3. Together, these results show that CQG-Single’s ML-1M gain is stable across training seeds and extends across most evaluated popularity–depth cells, without implying universal dominance.

CQG-AR retains step-wise autoregressive decoding while replacing discrete SID tokens with continuous residual outputs. It remains comparable to TIGER-style SID on ML-1M and achieves higher mean R@10 on both Amazon datasets. The largest gain occurs on Sports, where R@10 increases from 0.0236 to 0.0355 (50.4% relative); the seed-42 paired difference is 0.0146 (95% bootstrap CI [0.0119, 0.0173], $p < .0002$). Thus,

	TIGER	TIGER	CQG-Single	CQG-Single	CQG-AR	CQG-AR
Dataset	Cov.@10	Cov.@100	Cov.@10	Cov.@100	Cov.@10	Cov.@100
ML-1M	70.1%	97.0%	69.7%	97.6%	68.7%	97.6%
Beauty	74.4%	96.3%	89.5%	99.9%	87.9%	99.8%
Sports	94.0%	99.5%	63.1%	96.9%	71.8%	98.4%

Table 4: Empirical union-of-Top- K catalog coverage under matched returned-list sizes (seed 42). All methods use the same full-user sets and history filtering. Bold denotes the highest item coverage within each dataset and cutoff.

Depth	R@10	NDCG@10	R@100
$L = 1$	0.2924	0.1719	0.6550
$L = 2$	0.2909	0.1688	0.6525
$L = 3$	0.2914	0.1667	0.6619
$L = 4$	0.2944	0.1717	0.6697

Table 5: Single-seed CQG-AR decoding-depth ablation on ML-1M. All rows use seed 42, full-catalog CE, and intermediate-loss weight 0.3. Within this controlled ablation recipe, additional continuous refinements do not yield a monotonic R@10 or NDCG@10 improvement; the final primary recipe instead uses intermediate-loss weight 0.

continuous outputs can potentially remove SID-specific decoding constraints without sacrificing autoregressive generalization or retrieval accuracy.

6 Analysis

Effect of Autoregressive Refinement. Under the controlled depth-ablation recipe with intermediate-loss weight 0.3, increasing CQG-AR depth from one to four steps does not monotonically improve shallow retrieval. R@10 varies only from 0.2909 to 0.2944, and the single-step boundary ($L = 1$) is competitive with the three-step control ($L = 3$). The final primary CQG-AR recipe instead uses intermediate-loss weight 0, so this experiment isolates depth only within the ablation recipe. Additional refinement is more visible at the deeper cutoff: R@100 increases from 0.6550 at $L = 1$ to 0.6697 at $L = 4$, although the trend is not monotonic at every intermediate depth. These results suggest that serial continuous refinement may adjust deeper candidate ordering without reliably improving shallow accuracy. We therefore use $L = 3$ primarily to match the three-level SID decoding structure, rather than because three continuous outputs are intrinsically necessary.

Objective visibility. Full-catalog CE gives continuous queries access to every item at each update, whereas SID decoding receives token-level supervision. To control this difference, we train CQG-AR with 256 sampled negatives. Sampled CE reduces R@10 from 0.2982 to 0.2895 while retaining performance close to the TIGER-style baseline. Thus training-time catalog visibility explains part, but not all, of the observed behavior; our architecture-level claim is limited to eliminating discrete code lookup and finite-beam exclusion. The sampled-negative control and broader frozen-item objective comparison are reported in Appendix F.1.

Robustness across SID constructions. To examine whether the popularity skew is specific to our hierarchical- k -means baseline, we additionally evaluate a closer T5/RQ-VAE-style reproduction on Beauty. Across decoder seeds 42–44, 99.48% of generated sequences correspond to valid SIDs, yet the union of the Top-100 outputs covers only $44.0\% \pm 13.7\%$ of the catalog. Coverage increases sharply from $14.0\% \pm 7.6\%$ for Q1 rare items to $88.7\% \pm 13.0\%$ for Q4 popular items. This qualitative gradient suggests that popularity-skewed finite-output coverage is not unique to our hierarchical- k -means SID construction. Because this reproduction remains below the reference implementation in ranking accuracy, we treat it as a robustness diagnostic rather than a reproduced TIGER benchmark. Full details are reported in Appendix E.2 and Table 6.

Catalog group	Coverage@100	Std.
Q1 rare	14.0%	7.6%
Q2	26.1%	13.8%
Q3	47.4%	20.3%
Q4 popular	88.7%	13.0%
Overall	44.0%	13.7%

Table 6: Empirical union-of-top-100 catalog coverage for the closer T5/RQ-VAE Beauty reproduction, averaged over decoder seeds 42–44. Bold denotes the highest quartile coverage.

Returned-list-size sensitivity. For the seed-42 ML-1M TIGER checkpoint selected by constrained validation, increasing the effective returned-list size from 100 to 200 raises TargetInOutput from 66.7% to 76.3%. Across seven seeds, the corresponding means are 0.6667 ± 0.0018 and 0.7593 ± 0.0025 . The evaluator increases the internal beam when deduplication and history filtering would otherwise leave too few items. Larger returned lists therefore relax the observed support limitation; our evidence supports a finite-compute trade-off, not an architecture-invariant catalog ceiling.

7 Discussion and Limitations

Practical implications. Continuous-query retrieval removes discrete code lookup and finite-beam exclusion, while supporting exact or ANN retrieval without changing the output dimensionality as encoded items are added. The controlled depth ablation suggests that CQG-Single is preferable when shallow accuracy and inference efficiency are primary, whereas CQG-AR preserves step-wise autoregressive conditioning when this structure is desired. However, full-rank SASRec remains strongest on Sports, showing that continuous outputs remove a search constraint but do not guarantee superior ranking accuracy. Production-scale latency, memory, index-update cost, and cold-item accuracy remain unmeasured.

Interpretation of the crossover. The observed crossover is a checkpoint-, popularity-, and retrieval-depth-dependent empirical pattern rather than a universal ranking theorem. On ML-1M, TIGER is numerically stronger for rare targets at some shallow and intermediate cutoffs, whereas CQG-Single is generally stronger for more popular targets and at several deeper cutoffs. Beauty does not reproduce the rare-item shallow advantage, further limiting claims of universality. These results support a conditional tradeoff between concentrated finite-beam decoding and continuous retrieval, not universal dominance by either interface.

Limitations. Our evidence is specific to the evaluated model–search–compute configurations. Wider beams substantially raise TIGER target support, and our hierarchical- k -means TIGER-style baseline is weaker than published SID systems. The closer T5/RQ-VAE Beauty reproduction also does not match the reference implementation’s ranking accuracy, so its absolute coverage effect must not be generalized to a fully reproduced TIGER checkpoint. Matched-target analysis remains observational rather than a causal intervention on popularity. Full-catalog CE costs $O(N)$ per update and has only been evaluated on the reported catalog sizes; billion-item training, ANN latency, memory, and dynamic-catalog accuracy remain open.

8 Conclusion

We study finite-beam output support in SID-based generative recommendation. Across the evaluated models, target support and catalog coverage can be incomplete and popularity-skewed, with crossover patterns depending on checkpoint, popularity, and retrieval depth. CQG-Single generates one continuous query, whereas CQG-AR replaces SID tokens with autoregressive residual vectors while retaining competitive accuracy. A controlled CQG-AR ablation finds no consistent shallow benefit from refinement. CQG-Single is not proposed as the invention of dense retrieval, but a minimal, vocabulary-free interface that enables a controlled comparison with SID under a shared item space. Together, these results decouple autoregression

from discrete output constraints and motivate matched evaluation of accuracy, target support, and catalog coverage.

A Additional Cross-Dataset Diagnostics

Under the matched Amazon protocol, CQG-Single achieves the best R@10 and NDCG@10 in all Beauty runs and the best R@10 in two of three Sports runs. The comparison is against our TIGER-style baseline, not published TIGER.

Beauty				Sports			
Seed	Method	R@10	NDCG@10	Seed	Method	R@10	NDCG@10
42	TIGER	0.0464	0.0262	42	TIGER	0.0205	0.0109
	CQG-Single	0.0552	0.0342		CQG-Single	0.0350	0.0192
	CQG-AR	0.0527	0.0321		CQG-AR	0.0351	0.0189
43	TIGER	0.0443	0.0261	43	TIGER	0.0242	0.0126
	CQG-Single	0.0632	0.0389		CQG-Single	0.0361	0.0200
	CQG-AR	0.0612	0.0363		CQG-AR	0.0353	0.0189
44	TIGER	0.0525	0.0295	44	TIGER	0.0261	0.0135
	CQG-Single	0.0631	0.0379		CQG-Single	0.0379	0.0215
	CQG-AR	0.0573	0.0333		CQG-AR	0.0359	0.0190

Table 7: Complete Beauty and Sports seed-level results underlying the means and sample standard deviations in main-paper Table 3. Beauty and Sports use 22,363 and 21,519 aligned test users, respectively. TIGER checkpoints are selected with constrained validation and evaluated with valid-prefix decoding; all three methods use the same processed split and history filtering within each dataset. Bold denotes the best value for each dataset–seed pair and metric.

The 95% t -intervals for Beauty R@10 are [0.0372, 0.0583] for TIGER, [0.0490, 0.0720] for CQG-Single, and [0.0465, 0.0676] for CQG-AR. For Sports they are [0.0165, 0.0307], [0.0327, 0.0399], and [0.0344, 0.0365], respectively. Both continuous intervals on Sports lie above the TIGER interval; the Beauty intervals overlap. These are training-run, not user-level, intervals.

B Training-Seed Stability

CQG-Single exceeds TIGER in R@10 in all seven ML-1M runs (Table 8). The mean paired difference is 0.0107 (95% t interval [0.0021, 0.0193]; exact sign-flip $p = .0156$). TIGER TargetInOutput is 0.6667 ± 0.0018 at output size 100 and 0.7593 ± 0.0025 at size 200.

All methods share the frozen item table, split, and evaluation history. Seeds change downstream training; for TIGER they also change hierarchical k -means. “Paired seed” therefore means matched replicate index, not identical random draws across architectures.

Seed	TIGER		CQG-Single		CQG-AR	
	R@10	NDCG@10	R@10	NDCG@10	R@10	NDCG@10
42	0.2992	0.1756	0.3036	0.1736	0.2972	0.1720
43	0.3038	0.1751	0.3161	0.1874	0.3028	0.1750
44	0.2841	0.1644	0.3144	0.1831	0.2945	0.1711
45	0.3028	0.1768	0.3103	0.1830	0.3010	0.1755
46	0.3045	0.1781	0.3139	0.1849	0.2969	0.1739
47	0.2995	0.1728	0.3089	0.1832	0.2970	0.1734
48	0.3036	0.1747	0.3053	0.1804	0.2940	0.1738
Mean	0.2996	0.1739	0.3104	0.1822	0.2976	0.1735
Sample SD	0.0072	0.0046	0.0047	0.0044	0.0032	0.0016

Table 8: Seven-seed ML-1M results under valid-prefix constrained TIGER decoding. Each TIGER row uses the checkpoint reselected by constrained validation and evaluates all 6,040 test users with the same history filtering. All metrics use the standard NDCG@10 notation. Bold denotes the best result for each seed/metric and for the mean row.

C Experimental Protocol and Reproducibility

C.1 MSE-trained candidate generation and reranking.

The matched pipeline retrieves 100 distinct, history-filtered candidates and reranks them with the same pretrained SASRec. It evaluates the MSE-trained retrievers, not the CE-trained primary models.

Across ML-1M, Beauty, and Sports, CQG-Single followed by SASRec reranking reaches 97.7–102% of full-rank SASRec Recall@10. CQG-AR nearly recovers full-rank accuracy on ML-1M and Sports but remains lower on Beauty. The learned retrievers also outperform random and nearest-neighbor candidate generation. These are fixed-checkpoint, not multi-seed, results.

(a) Standalone and reranked accuracy				(b) Candidate hierarchy		
Method	ML-1M	Beauty	Sports	Generator	ML-1M	Sports
<i>Recall@10</i>						
SASRec full-rank	.2797	.0558	.0390	Random	.0252	.0037
Single MSE standalone	.2425	.0524	.0307	Last-item kNN	.2349	.0311
Single MSE + SASRec	.2801	.0569	.0381	Mean-of-5 kNN	.2545	.0310
AR MSE standalone (s42)	.2414	.0466	.0294	Single + SASRec	.2801	.0381
AR MSE + SASRec (s42)	.2791	.0486	.0378	AR + SASRec	.2791	.0378
TIGER + SASRec	.2864	–	–	SASRec full-rank	.2797	.0390
<i>NDCG@10</i>						
SASRec full-rank	.1535	.0296	.0197			
Single MSE standalone	.1424	.0266	.0150			
Single MSE + SASRec	.1554	.0308	.0194			
AR MSE standalone (s42)	.1366	.0276	.0160			
AR MSE + SASRec (s42)	.1543	.0265	.0194			
TIGER + SASRec	.1571	–	–			

Table 9: Matched Top-100 MSE candidate-generation and reranking diagnostics. (a) Standalone and SASRec-reranked accuracy; CQG-AR uses the available seed-42 outputs, and TIGER reranking is available only for ML-1M. (b) Candidate-generation hierarchy under the same SASRec reranker; only SASRec full-rank bypasses candidate generation. Bold denotes the best result in each dataset and metric block.

C.2 Paired Crossover and Candidate-Generation Diagnostics

Quartile	Method	10	20	30	40	50	60	70	80	90	100	120	150
Q1 rare ($n = 193$)	Single	.0622	.1192	.1451	.1658	.1865	.1917	.2124	.2435	.2850	.2953	.3161	.3575
	AR	.0829	.1088	.1451	.1606	.1762	.1969	.2280	.2487	.2591	.2642	.3057	.3368
	TIGER	.0674	.1036	.1347	.1658	.1917	.2021	.2228	.2591	.2642	.2798	.3057	.3161
Q2 ($n = 581$)	Single	.1773	.2719	.3236	.3614	.3959	.4320	.4509	.4802	.4923	.5112	.5508	.5835
	AR	.1497	.2530	.3098	.3528	.3907	.4182	.4372	.4613	.4819	.4974	.5284	.5645
	TIGER	.1756	.2547	.3133	.3528	.3873	.4234	.4423	.4647	.4871	.4991	.5250	.5611
Q3 ($n = 1,322$)	Single	.2519	.3449	.4206	.4675	.5030	.5310	.5628	.5847	.6006	.6188	.6430	.6740
	AR	.2390	.3434	.4077	.4486	.4841	.5204	.5461	.5688	.5840	.6051	.6339	.6566
	TIGER	.2436	.3411	.4077	.4508	.4811	.5091	.5363	.5582	.5809	.5991	.6316	.6536
Q4 popular ($n = 3,944$)	Single	.3555	.4754	.5479	.5974	.6331	.6620	.6864	.7039	.7186	.7335	.7571	.7842
	AR	.3494	.4635	.5319	.5809	.6174	.6488	.6747	.6978	.7125	.7277	.7497	.7792
	TIGER	.3474	.4612	.5307	.5834	.6273	.6509	.6795	.7023	.7208	.7335	.7619	.7896

Table 10: Recall at the 12 depths in main-paper Figure 3, evaluated on the same 6,040 users using paired seed-42 checkpoints and matched Top-200 outputs. This CQG-Single checkpoint has R@10 0.3063, rather than 0.3036 in the primary seven-seed Table 8. CQG-Single-TIGER differences at $K = 10, 50, 90$ are 0.0071, 0.0093, 0.0040; all paired-bootstrap 95% CIs include zero (10,000 resamples). Bold denotes the best method; ties are bolded.

C.3 Computing Infrastructure

Experiments ran in a Linux container with one NVIDIA A800 80GB PCIe GPU (81,920 MiB device memory), 14 allocated CPU cores, and approximately 120GB of host-visible memory. The software environment uses Python 3.12.3, PyTorch 2.8.0+cu128, CUDA 12.8 as bundled with PyTorch, cuDNN 9.10.2, Transformers 5.13.1, NumPy 2.3.2, scikit-learn 1.9.0, and FAISS-CPU 1.14.3. Training uses bfloat16 automatic mixed precision where indicated; evaluation metrics are accumulated outside autocast.

C.4 Dataset Splits

All datasets are converted into chronological user sequences. For each user, we use the last interaction for testing, the penultimate interaction for validation, and all earlier interactions for training. Users with insufficient history after filtering are removed. Evaluation is full-catalog unless otherwise stated: the target item is ranked against all candidate items rather than against sampled negatives. Cross-dataset accuracy uses $K \in \{5, 10, 20\}$; the ML-1M crossover curve uses the measured depths $K \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 150\}$.

C.5 Evaluation Metrics

For a user u with held-out target item i_u^* , Recall@ K is one if i_u^* appears in the top- K list and zero otherwise. NDCG@ K discounts a hit by the target rank. We distinguish two output-based support metrics. The SID target-support rate is

$$\text{TargetSupport@B} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbf{1}\{i_u^* \in \text{SIDOutput}_B(u)\},$$

where $\text{SIDOutput}_B(u)$ is the final ranked list of at most B valid, distinct, history-filtered items produced by constrained SID decoding. When filtering would leave fewer than B items, the internal beam is expanded until the list is filled or the valid search space is exhausted. Completed sequences require an exact catalog-code match; no numeric nearest-code mapping is used. This is a per-user target-conditioned quantity and is closely related to Recall@ B ; it must not be interpreted as catalog coverage. For a common finite-output comparison

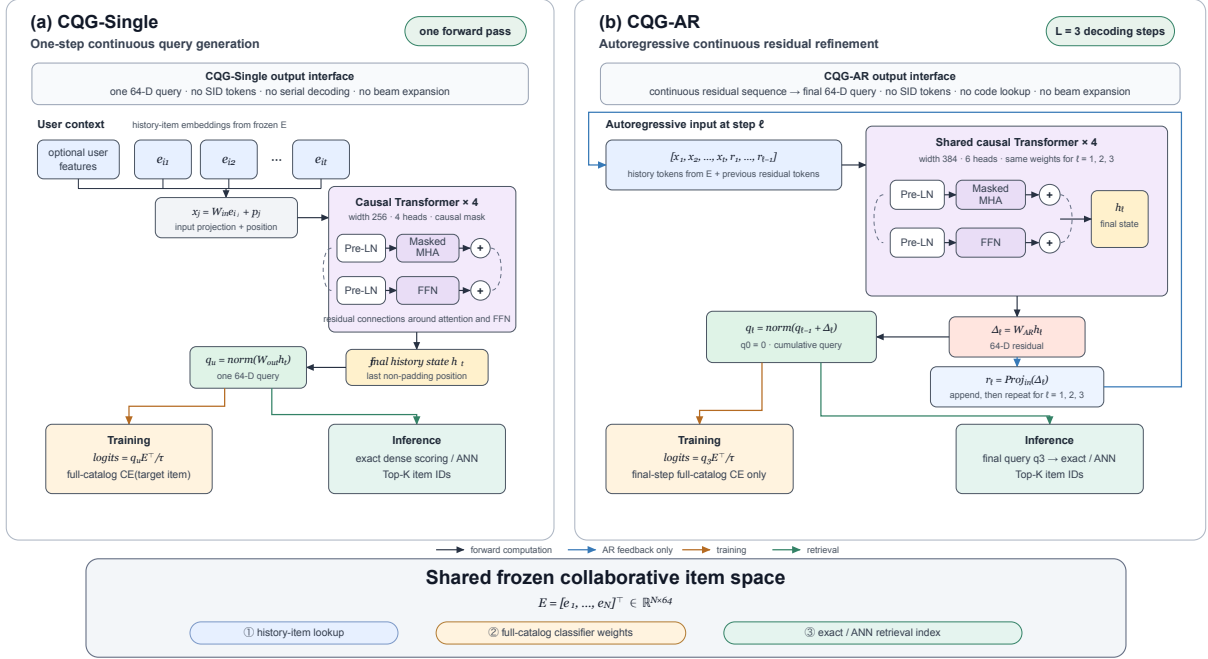


Figure 4: Transformer-level architectures of CQG-Single and CQG-AR. Both methods consume history-item embeddings from the same frozen item table E . CQG-Single produces one continuous query in a single forward pass, whereas CQG-AR autoregressively appends projected residual tokens and retrieves with the final cumulative query. Both use the frozen table as the full-catalog classifier and exact/ANN retrieval index.

across methods, we define

$$\text{TargetInOutput@K}(M) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbf{1}\{i_u^* \in \text{TopK}_M(u)\}.$$

This common finite-output measure is Recall@ K under our single-target leave-one-out protocol. Let $\mathcal{I}_{\text{test}} = \{i_u^* : u \in \mathcal{U}\}$ be the set of distinct held-out target items. We additionally define

$$\text{TargetItemSupport@K}(M) = \frac{|\{i \in \mathcal{I}_{\text{test}} : \exists u, i_u^* = i \wedge i \in \text{TopK}_M(u)\}|}{|\mathcal{I}_{\text{test}}|}.$$

Unlike user-weighted Recall, this measures how many distinct target-item types are successfully retrieved at least once. For any method M , empirical union coverage at a matched returned-list size K is

$$\text{CatalogCoverage@K}(M) = \frac{|\bigcup_{u \in \mathcal{U}} \text{TopK}_M(u)|}{|\mathcal{I}|}.$$

We report this same output-set quantity for TIGER, CQG-Single, and CQG-AR. Catalog-wide scoreability of a dense interface is a separate property and does not imply 100% CatalogCoverage@ K .

Our primary diagnostics directly report item-balanced support in Table 2 in the main paper, matched union coverage in Table 4 in the main paper, and explicit training-seed variation in Table 8. These quantities separately address target imbalance, output diversity, and training-run uncertainty; within-seed item resampling is not used as a proxy for training-seed variation.

D Model and Training Details

D.1 Frozen Collaborative Item Space

SASRec backbone and examples. The item space is learned by a SASRec encoder trained from scratch on the training portion of each dataset; no text, image, T5, or language-model features are used in the main item encoder. The model contains a trainable item lookup table $V \in \mathbb{R}^{N \times 64}$, learned positional embeddings, and two causal self-attention blocks with two heads, hidden dimension 64, and dropout 0.2. For a chronological training sequence $(i_1, \dots, i_T)$, every position $t = 2, \dots, T$ yields

$$([i_{\max(1, t-L)}, \dots, i_{t-1}], i_t),$$

where $L = 50$ on ML-1M and $L = 20$ on Beauty and Sports. Histories are left-padded with item ID 0; the padding embedding is fixed at zero and is excluded from evaluation.

BPR pretraining and checkpoint selection. Let h_t be the final SASRec hidden state for a history, i_t its positive next item, and i^- one item drawn uniformly from IDs $\{1, \dots, N - 1\}$. SASRec is optimized with

$$\mathcal{L}_{\text{BPR}} = -\log \sigma(h_t^\top V_{i_t} - h_t^\top V_{i^-}).$$

We use Adam with learning rate 10^{-3} and batch size 256. Every five epochs, validation ranks the held-out validation item against the full catalog (masking padding). We retain the checkpoint with the highest validation NDCG@10. The maximum is 200 epochs for ML-1M and Beauty and 400 for Sports; checkpoint selection, rather than the last epoch, determines the exported representation.

Export, normalization, and sharing. After loading the selected checkpoint, we export the lookup parameters directly,

$$E = \text{SASRec.item_emb.weight} \in \mathbb{R}^{N \times 64}.$$

Thus e_i is a global collaborative item parameter, not a contextual SASRec hidden state. We freeze E for every downstream run. Its non-padding rows are L2-normalized for cosine retrieval, CE, BPR, and InfoNCE; the MSE-trained CQG-Single objective regresses to the raw exported vector. The same frozen table is used (i) to embed history IDs, (ii) as the MSE/InfoNCE target table or the full-catalog CE classifier, (iii) as the exact or ANN retrieval index, and (iv) as the input to the hierarchical clustering used by our TIGER-style baseline. This sharing prevents differences in the item encoder from explaining differences among the output interfaces. Each dataset uses one fixed, checkpoint-selected SASRec export for all reported downstream seeds. Seeds 42–48 on ML-1M and 42–44 on Beauty/Sports change downstream initialization, data order, and optimization; TIGER additionally reruns hierarchical clustering for each indexed seed. The ML-1M table has shape 3417×64 .

D.2 CQG-Single Architecture and Optimization

CQG-Single has no pretrained backbone and does not update E . Frozen 64-dimensional history vectors are projected to width 256 and combined with learned positional embeddings. Four pre-norm causal Transformer blocks, each with four attention heads, feed-forward dimension 1024, and dropout 0.1, produce the final valid history state. A linear head maps this state back to 64 dimensions and the query is L2-normalized before scoring the normalized catalog. The CE configuration uses Adam, learning rate 10^{-3} , batch size 256, at most 50 epochs, validation every five epochs, and early stopping with patience 15 based on validation NDCG@10. MSE and InfoNCE change only the objective and their stated learning-rate/epoch settings in Table 11; the history protocol and frozen target table remain the same.

D.3 CQG-AR Architecture and Optimization

CQG-AR is also trained from scratch with frozen E . It projects history vectors from 64 dimensions to width 384 and uses four pre-norm causal Transformer blocks with six heads, feed-forward dimension 1024, and dropout 0.1. A learned BOS vector starts continuous decoding. At step ℓ , a linear head followed by tanh

produces $\Delta_\ell \in \mathbb{R}^{64}$; a learned scalar controls its magnitude, the cumulative query is initialized as $q^{(0)} = \mathbf{0}$, and

$$q^{(\ell)} = \text{norm}(q^{(\ell-1)} + \Delta_\ell).$$

The residual is projected back to width 384, tagged with a generated-token type embedding and a position embedding, and appended as the next causal decoder input. The main configuration uses three steps, so the serial outputs are $(\Delta_1, \Delta_2, \Delta_3)$, but all three refer to the same next-item target i_t ; they are not labels for three SID levels.

For the final CE recipe, only $q^{(3)}$ receives direct catalog CE supervision (`intermediate_weight=0`); earlier steps receive gradient through their effect on later decoding. The depth ablation instead uses intermediate weight 0.3 for the non-final queries, always with the same next-item label. CQG-AR uses AdamW with learning rate 3×10^{-4} , weight decay 0.01, batch size 1024, at most 30 epochs, bfloat16 automatic mixed precision, gradient-norm clipping at 1.0, and early stopping with patience 5 on validation NDCG@10. The selected checkpoint is evaluated against the full catalog after appending the validation item to the test history and masking all previously observed items.

D.4 Hyperparameters

Method	Item d	Hidden	Layers	Heads
SASRec item encoder	64	64	2	2
CQG-Single MSE	64	256	4	4
CQG-Single InfoNCE	64	256	4	4
CQG-Single CE	64	256	4	4
CQG-AR	64	384	4	6
CQG-AR sampled CE	64	384	4	6
Qwen2.5-7B hybrid probe	64	3584	28	28
TIGER-style SID decoder	64	384	4	6

Method	LR	Batch	Epochs/Steps	Objective
SASRec item encoder	1e-3	256	200/400 max.	BPR, one uniform negative
CQG-Single MSE	1e-3	256	50 max.	MSE to target embedding
CQG-Single InfoNCE	1e-4	256	100 max.	In-batch InfoNCE, $\tau = 0.07$
CQG-Single CE	1e-3	256	50 max.	Full-catalog softmax CE
CQG-AR	3e-4	1024	30 max.	AR residuals + full-catalog CE
CQG-AR sampled CE	3e-4	1024	30 max.	AR residuals + 256 negatives
Qwen2.5-7B hybrid probe	5e-5	32×8	10	InfoNCE with projection head
TIGER-style SID decoder	1e-3	256	50K steps	Autoregressive CE over SID tokens

Table 11: Training hyperparameters for the three datasets. “Hidden” is the Transformer width; all continuous queries remain 64-dimensional. The Qwen2.5-7B probe uses its 3,584-dimensional, 28-layer, 28-attention-head backbone and a 64-dimensional projection head. SASRec uses at most 200 epochs on ML-1M/Beauty and 400 on Sports. Sequence length is 50 for ML-1M and 20 for Beauty/Sports; “max.” denotes early-stopped training.

D.5 Continuous-Query Dataloader

CQG-Single and CQG-AR use the same next-item examples derived from user histories. Each training example contains a padded history sequence and the next-item target. The history item IDs are mapped to frozen item embeddings before entering the query generator; the target ID is used either to index the target embedding for MSE/InfoNCE or to index the full-catalog label for CE. A batch therefore has the form

$$(\mathbf{H}, \mathbf{y}) = \{([i_{t-L}, \dots, i_{t-1}], i_t)\}_{b=1}^B,$$

where L is the maximum sequence length and B is the batch size. Padding positions are masked and never treated as candidate items. This dataloader is intentionally the same sequential next-item protocol used by SASRec, so the comparison isolates the output interface rather than changing the recommendation task.

For example, one actual ML-1M training pair is

$$([2758, 1069, 1445, 862, 1979], 1520).$$

The model input is the 5×64 matrix obtained by looking up the five history IDs in frozen E . Under CE, the label is the integer 1520 among 3,416 non-padding catalog items; the implementation stores these items in a 3,417-row table whose row 0 is masked padding. Under MSE or InfoNCE, the corresponding target is row E_{1520} . At test time the validation item is appended to the training history, the final item is the held-out target, and all items already present in the resulting history are masked from ranking.

D.6 Continuous Query Training Objectives

The MSE variant regresses the generated query q_u to the frozen target embedding e_{i^*} (the MSE-trained CQG-Single run uses the raw SASRec export):

$$\mathcal{L}_{\text{MSE}} = \|q_u - e_{i^*}\|_2^2.$$

The InfoNCE variant treats other targets in the batch as negatives:

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(q_u^\top e_{i^*}/\tau)}{\sum_{j \in \mathcal{B}} \exp(q_u^\top e_j/\tau)}.$$

The CE variant scores the full frozen catalog:

$$\mathcal{L}_{\text{CE}} = -\log \frac{\exp(q_u^\top e_{i^*}/\tau)}{\sum_{j=1}^N \exp(q_u^\top e_j/\tau)}.$$

For CQG-AR, $q_u = q^{(3)}$ in the final recipe. All autoregressive steps share the same next-item target; there is no separate label per residual. All variants can score every encoded item by dense similarity at inference time. Their empirical CatalogCoverage@ K is nevertheless measured from the union of finite top- K outputs.

E SID Construction and Decoder Training

E.1 Semantic-ID Construction

Our TIGER-style baseline hierarchically clusters the frozen SASRec item table into depth-3 paths. Collisions are resolved before decoder training by appending deterministic residual identifiers ordered by item ID. On ML-1M, 99.4% of items are already unique after the three cluster levels; the remaining items become one-to-one catalog codes after this tie-break.

E.2 TIGER T5/RQ-VAE Reproduction Recipe

The Beauty T5/RQ-VAE run uses fixed codes and three decoder seeds; the tokenizer is not retrained. Across these runs, 99.48% of generated sequences are valid SIDs, while the union of Top-100 outputs covers $44.0\% \pm 13.7\%$ of the catalog, rising from $14.0\% \pm 7.6\%$ in Q1 to $88.7\% \pm 13.0\%$ in Q4.

Its R@10 is 0.0408 ± 0.0016 , below the reference implementation. Thus it confirms popularity-skewed SID coverage under a second construction, but not published TIGER accuracy.

E.3 Valid-Prefix Decoding

The SID decoder uses the same history–target examples as the continuous models but predicts code tokens. At inference, a catalog trie masks invalid prefixes, completed codes use exact item lookup, historical items are removed, and the internal beam expands when necessary to fill the requested number of valid, distinct items. Checkpoint selection uses this same constrained protocol.

F Additional Ablations

F.1 Objective, Catalog-Visibility, and Temperature Diagnostics

Objective and catalog visibility. Full-catalog CE yields the highest Recall@10 in the frozen item space. Replacing it with 256 sampled negatives lowers CQG-AR mean R@10 from 0.2982 to 0.2895, while remaining above MSE and InfoNCE. CE-joint is separate because it also changes the item representations.

Temperature. Lower temperatures monotonically improve shallow retrieval in the tested range. CQG-Single R@10 decreases from 0.3114 at $\tau = 0.07$ to 0.3070 at 0.10 and 0.2846 at 0.20; the aligned seed-42 CQG-AR control decreases from 0.2972 to 0.2886 and 0.2725. The MSE-trained CQG-Single reference reaches 0.2425.

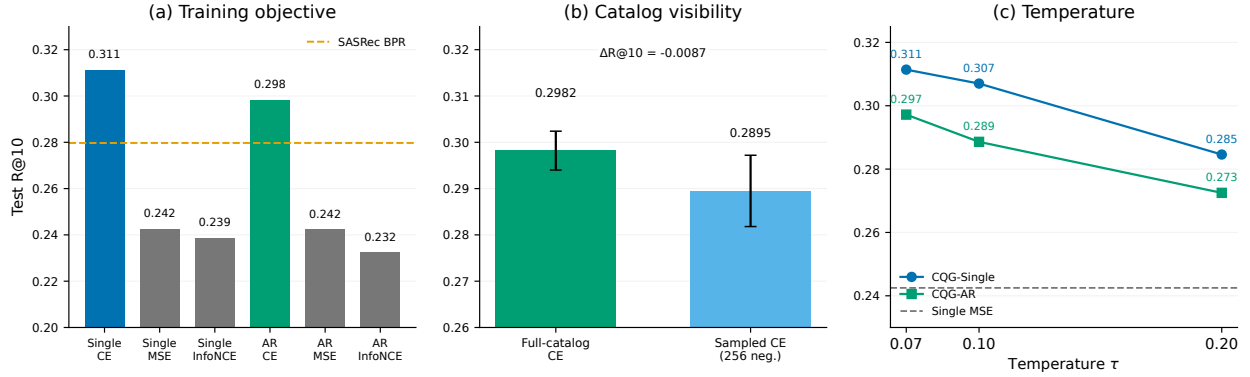


Figure 5: Unified ML-1M objective diagnostics. (a) Full-catalog CE gives the highest R@10 for both continuous decoders in the shared frozen item space. (b) Replacing full-catalog CE with 256 sampled negatives reduces CQG-AR mean R@10 by 0.0087. Error bars denote sample SD over seeds 42–44. (c) Lower softmax temperatures improve R@10 for both continuous decoders in the evaluated range; the dashed line is the CQG-Single MSE reference.

G LLM Backbone Diagnostic

No tested Qwen configuration outperforms the scratch Transformer (Figure 6). Joint training collapses the item space (item–item cosine 0.9936); freezing it avoids collapse but leaves Recall@10 below the SASRec gate. The best hybrid reaches $R@10 = 0.2118$ versus 0.248 for the single-run scratch model and trails it at every cutoff. This scratch checkpoint is distinct from the three-seed MSE result in Figure 5.

Model	R@5	R@10	R@20	R@50	R@70	R@90
Scratch Transformer (LLM diagnostic)	0.172	0.248	0.345	0.498	0.555	0.595
Qwen text-aware hybrid	0.134	0.212	0.308	0.460	0.517	0.564

Table 12: Single-run recall curve comparison between the MSE-frozen scratch Transformer used in the LLM diagnostic and the best tested LLM hybrid configuration. The scratch row is not the three-seed CQG-Single MSE aggregate reported in Figure 5. Bold denotes the best value in each column.

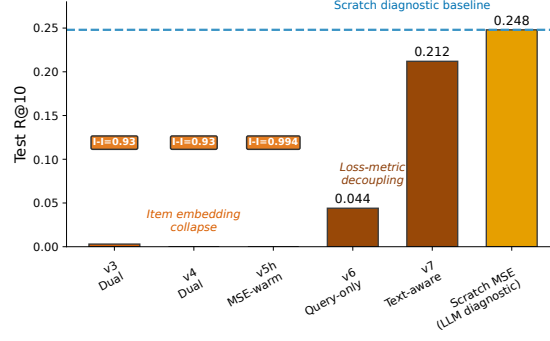


Figure 6: LLM backbone diagnostics. Pretrained language backbones do not automatically align with the frozen collaborative item space; the best text-aware Qwen hybrid still trails the single-run, MSE-frozen scratch query Transformer used in this diagnostic ($R@10 = 0.248$). This checkpoint is separate from the three-seed CQG-Single MSE result in Figure 5. In the figure, v3/v4 are jointly trained dual-tower variants, v5h is an MSE-warm-started dual tower, v6 trains only the query side against a frozen item table, and v7 is the frozen-item text-aware hybrid. The first three collapse the item space; v6/v7 instead exhibit loss-metric decoupling. These are single-run diagnostics, not primary baselines.

References

- Zhiyuan Cheng, Longying Lai, Yue Liu, and Yu Sun. Toward sustainable on-device intelligence: A survey on energy-efficient rag systems with small language models. *Available at SSRN 6698538*, 2026.
- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.
- Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.
- Junchen Fu, Xuri Ge, Alexandros Karatzoglou, Ioannis Arapakis, Suzan Verberne, Joemon M Jose, and Zhaochun Ren. Differentiable semantic id for generative recommendation. *arXiv preprint arXiv:2601.19711*, 2026.
- Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.
- Yupeng Hou, Haven Kim, Clark Mingxuan Ju, Eduardo Escoto, Neil Shah, and Julian McAuley. Expressiveness limits of autoregressive semantic id generation in generative recommendation. *arXiv preprint arXiv:2605.06331*, 2026.
- Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.
- Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- 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*, pages 11523–11532, 2022.

- Jialei Li, Yang Zhang, Yimeng Bai, Shuai Zhu, Ziqi Xue, Xiaoyan Zhao, Dingxian Wang, Frank Yang, Andrew Rabinovich, and Xiangnan He. Unigrec: Unified generative recommendation with soft identifiers for end-to-end optimization. *arXiv preprint arXiv:2601.17438*, 2026.
- Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation, 2023a. URL <https://arxiv.org/abs/2304.03879>.
- Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large language models for generative recommendation: A survey and visionary discussions. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pages 10146–10159, 2024.
- Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(3), December 2023b. ISSN 1046-8188. doi: 10.1145/3631116. URL <https://doi.org/10.1145/3631116>.
- Mingchao Liu, Yu Sun, Ruixiao Sun, Xin Dong, Xiang Shen, Hongwei Wang, Hongyu Xiong, and Yang Song. IPS: In-prompt process supervision for short video content moderation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 1277–1288, 2026. ISBN 979-8-89176-394-4. doi: 10.18653/v1/2026.acl-industry.89. URL <https://aclanthology.org/2026.acl-industry.89/>.
- Haohao Qu, Shanru Lin, Yujuan Ding, Yiqi Wang, and Wenqi Fan. Diffusion generative recommendation with continuous tokens. In *Proceedings of the ACM Web Conference 2026*, pages 7259–7270, 2026.
- Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. Recommender systems with generative retrieval. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- 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*, pages 1441–1450, 2019.
- Yu Sun, Yin Li, Ruixiao Sun, Chunhui Liu, Fangming Zhou, Ze Jin, Linjie Wang, Xiang Shen, Zhuolin Hao, and Hongyu Xiong. Audio-enhanced vision-language modeling with latent space broadening for high quality data expansion. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pages 4872–4881. Association for Computing Machinery, 2025. ISBN 9798400714542. doi: 10.1145/3711896.3737195. URL <https://doi.org/10.1145/3711896.3737195>.
- 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*, pages 355–364, 2024.
- Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*, 2022.
- Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.
- Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, et al. Eager: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3245–3254, 2024.
- Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614, 2022.

- Zixuan Wang, Yu Sun, Hongwei Wang, Baoyu Jing, Xiang Shen, Xin Dong, Zhuolin Hao, Hongyu Xiong, and Yang Song. Reasoning-enhanced domain-adaptive pretraining of multimodal large language models for short video content governance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1104–1112. Association for Computational Linguistics, 2025. ISBN 979-8-89176-333-3. doi: 10.18653/v1/2025.emnlp-industry.77. URL <https://aclanthology.org/2025.emnlp-industry.77/>.
- Liu Yang, Fabian Paischer, Kaveh Hassani, Jiacheng Li, Shuai Shao, Zhang Gabriel Li, Yun He, Xue Feng, Nima Noorshams, Sem Park, et al. Unifying generative and dense retrieval for sequential recommendation. *arXiv preprint arXiv:2411.18814*, 2024.
- Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *Advances in Neural Information Processing Systems*, 36:24247–24261, 2023.
- 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)*, pages 1435–1448. IEEE, 2024.