SCALE-AWARE IDENTIFIERS FOR GENERATIVE RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative recommendation predicts the next item by autoregressively generating its semantic identifier (SID), making SID construction critical to recommendation performance. Existing hierarchical SID methods commonly use residual quantization (RQ), where each level selects a codeword and directly subtracts it from the residual. This update ignores the projection magnitude along the selected codeword and can produce non-orthogonal, non-minimum-norm residuals, leading to redundant decomposition across levels. Therefore, we propose Scale-aware Residual Quantization (SRQ), which computes projection magnitudes and subtracts the selected codeword with its corresponding scale, yielding an orthogonal, minimum-norm residual. To use the continuous magnitudes for autoregressive generation, we discretize them into quantile-based magnitude tokens. Together with codeword tokens, they form Scale-aware Identifiers for Generative Recommendation (SIGER). Moreover, SIGER can be integrated into two-stage training, unified tokenizer–recommender training, and LLM-based post-training. Analysis shows that SIGER improves residual orthogonality and codebook utilization, reducing SID collision from 5.26% to 2.99% with codeword tokens alone and to 0.04% with 128 magnitude bin size. These identifier-level improvements translate into stronger recommendation performance: SIGER improves two-stage baselines by up to 9.6% Recall@5 and 9.9% NDCG@5, improves over unified baselines by up to 2.4% NDCG@5 on Beauty and 8.0% Recall@10 on Pet, and achieves the best performance on the Office in the LLM-based setting. Further ablations on 2-level and code-only variants show that SIGER encodes semantic information more compactly and demonstrate the effectiveness of scale-aware decomposition.

1 INTRODUCTION

Generative recommendation (GR) has recently emerged as a promising paradigm for modern recommender systems (Geng et al., 2022; Rajput et al., 2023; Liu et al., 2025b; Wang et al., 2024b; Zhou et al., 2025; Li et al., 2026; Kong et al., 2025). Unlike traditional recommenders that retrieve or rank items from a large candidate set using learned user–item matching scores (Hidasi et al., 2016; Tang & Wang, 2018; Kang & McAuley, 2018; Sun et al., 2019), generative recommenders directly generate the target item in a tokenized output space. Specifically, each item is represented by a sequence of discrete semantic identifiers (SIDs) via a tokenizer, and the next item is predicted autoregressively by a recommender generating its SID conditioned on the user's interaction history.

Since generative recommenders encode interaction histories and prediction targets in the SID space, the quality of SID construction is central to their performance. Hierarchical SIDs are typically obtained by discretizing item embeddings with residual quantization (RQ) methods, including RQ-KMeans (Luo et al., 2025) and RQ-VAE (Rajput et al., 2023). RQ-KMeans is simple but often suffers from high identifier collision. Its variant, RK-Means, reduces the collision by uniformly assigning the same number of codewords in each cluster. However, this constraint can disrupt the natural semantic structure. Meanwhile, the most standard approach, RQ-VAE, learns codebooks with reconstruction objectives. Nevertheless, both methods update the residual by directly subtracting the selected codeword. This implicitly assumes that the residual contains exactly one unit of the selected direction, which generally does not hold. Specifically, given the residual $\boldsymbol{v}_l$ at level l and the selected codeword $\boldsymbol{e}_l$, standard residual quantization updates the residual as $\boldsymbol{v}_{l+1} = \boldsymbol{v}_l - \boldsymbol{e}_l$. As a result, $\boldsymbol{v}_{l+1}$ is not necessarily orthogonal to $\boldsymbol{e}_l$ and does not attain the minimum residual norm (Sec-

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.

tion 3.1). The next quantization level may therefore still need to encode information from the same semantic direction, leading to redundant decomposition across levels.

Motivated by this observation, we propose *Scale-aware Residual Quantization* (SRQ), which records not only *which* codeword but also *how much* of the selected codeword direction is present. At each level, SRQ computes the projection magnitude for each codeword, selects the codeword that minimizes the scaled residual, and stores the selected codeword index together with its corresponding magnitude. This yields an orthogonal, minimum-norm residual and reduces redundant decomposition across levels. Since SRQ modifies only the residual update, it can be applied to different RQ-based tokenizers regardless of how their codebooks are constructed. However, while the projection magnitudes are continuous, predicting them through regression would also introduce a separate optimization target beyond standard token generation. We therefore discretize the magnitudes into level-wise quantile bins and represent them as additional magnitude tokens. Together with the codeword tokens, these form our *Scale-aware Identifiers for Generative Recommendation* (SIGER), which preserve both semantic direction and magnitude while remaining fully compatible with autoregressive generation. As a tokenizer-level method, SIGER can be integrated into different GR paradigms, including two-stage training, unified tokenizer–recommender training (Li et al., 2026), and LLM-based post-training (Kong et al., 2025). Our contributions are summarized:

- We identify a geometric limitation of residual-quantization-based SID construction. Standard RQ directly subtracts the selected codeword with a unit coefficient, which can leave a non-orthogonal and non-minimum-norm residual. Across Amazon Beauty, Toys, and Sports (Hou et al., 2024), RQ-VAE leaves clear residual-codeword correlation, with average absolute cosine values around 0.15–0.26 across quantization levels.
- We propose SIGER, a scale-aware SID framework instantiated by a scale-aware residual quantization (SRQ) tokenizer. It records both semantic direction and projection magnitude, improving codebook usage and identifier resolution. On Beauty, first-level codebook usage increases from $158/256$ to $256/256$ and entropy improves from $H = 6.97$ to $H = 7.73$; on Toys and Sports, first-level usage also improves from $117/256$ and $134/256$ to full usage. With $B = 32$ magnitude bins, raw SID collision drops from 5.26% to 0.21% on Beauty, 9.62% to 0.48% on Toys, and 9.48% to 0.69% on Sports.
- We evaluate SIGER across three GR training paradigms: two-stage training, unified tokenizer-recommender training, and LLM-based SFT-then-RL post-training. SIGER improves two-stage performance by up to 9.6% Recall@5 and 9.9% NDCG@5, improves over UniGRec by up to 2.4% NDCG@5 on Beauty and 8.0% Recall@10 on Pet, and achieves all the best metrics on the Office in the LLM-based setting, with gains of 5.9% HR@3, 4.2% NDCG@3, and 2.4% NDCG@10 over MiniOneRec.
- We provide detailed ablations showing that scale-aware identifiers capture more compact item information. A two-level SIGER variant maintains low collision rates of $\leq 3.28\%$ across datasets, while two-level RQ-VAE suffers from up to 53.47% collision rates. Code-only SIGER variants, which use the same SID length as RQ-VAE, still improve over RQ-VAE, indicating that scaled residual decomposition improves the codeword itself. Adding magnitude tokens further increases identifier resolution and leads to stronger accuracy.

2 Related Works

Sequential recommendation. Sequential recommendation predicts the next item according to a user's historical interactions. Earlier studies model user behavior with Markov chains, recurrent neural networks, or convolutional networks (Hidasi et al., 2016; Tang & Wang, 2018), while recent methods commonly adopt self-attention or Transformer architectures to capture long-range dependencies in interaction sequences (Kang & McAuley, 2018; Sun et al., 2019). These methods usually learn continuous item embeddings and rank candidates by matching them with the user representation. Although effective, they still follow the conventional discriminative recommendation paradigm, which requires scoring or retrieving items from a large candidate set. In contrast, generative recommendation predicts the target item by generating its discrete identifier.

Generative recommendation. Generative recommendation has recently emerged as a promising alternative to conventional retrieve-and-rank systems. P5 (Geng et al., 2022) and M6-Rec (Cui et al.,

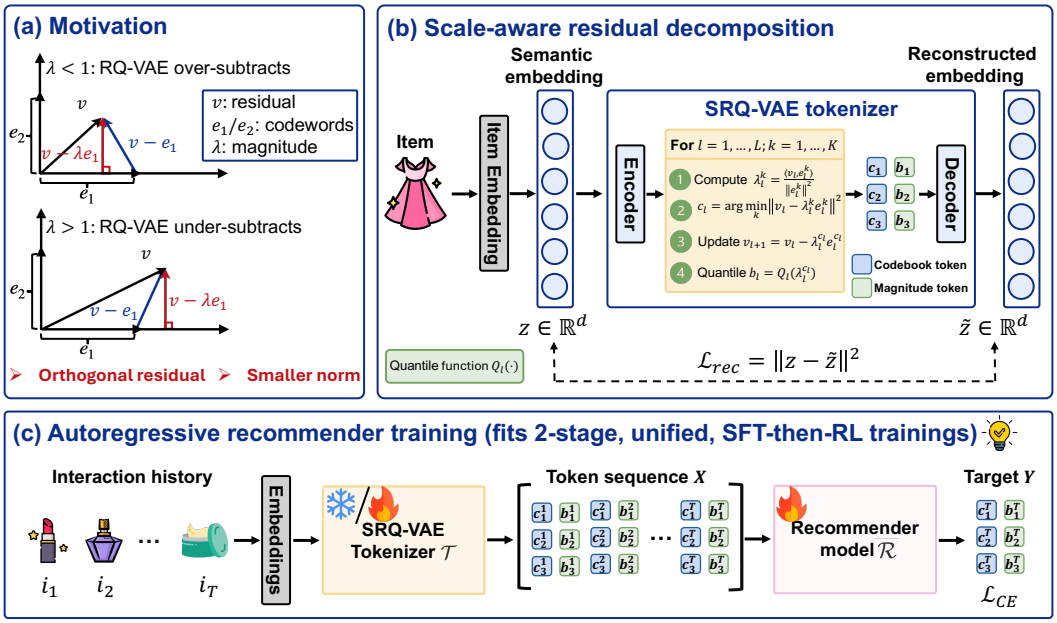


Figure 1: Overview of SIGER. (a) Standard residual quantization can leave a non-orthogonal, non-minimal residual. (b) The SRQ-VAE tokenizer constructs scale-aware identifiers by selecting codeword directions, quantizing projection magnitudes, and reconstructing item embeddings for trainig. (c) The tokenizer converts user histories into scale-aware SID sequences, and the generative recommender predicts the next target SID autoregressively.

2022) formulate multiple recommendation tasks as text generation with pretrained language models, but they usually rely on textual prompts or item IDs. TIGER (Rajput et al., 2023) introduces semantic identifiers generated by residual quantization and trains an autoregressive model to generate the next item's identifier. Following TIGER, several works improve generative recommendation from different perspectives. LETTER (Wang et al., 2024a) enhances language-model-based recommendation with learnable token representations, EAGER (Wang et al., 2024c) studies efficient and effective generative retrieval, and OneRec (Deng et al., 2025) further introduces supervised fine-tuning and reinforcement-learning-based post-training. More recent methods focus on improving the alignment between tokenizer and recommender. ETEG-Rec (Liu et al., 2025b) explores end-to-end generative recommendation, and UniGRec (Li et al., 2026) proposes soft identifiers to enable unified tokenizer–recommender optimization. MiniOneRec (Kong et al., 2025) scales this paradigm with open-source LLM backbones and recommendation-oriented post-training. These studies demonstrate the potential of generative recommendation, while also showing that the quality of item identifiers is critical to the final recommendation performance.

3 Preliminary

We formulate the next-item prediction as an autoregressive sequence generation task, which comprises two components: a tokenizer $\mathcal{T}$ and a recommender $\mathcal{R}$. Let $\mathcal{I}$ be the item set and $S = [i_1, \ldots, i_T]$ denote a user's interaction items with length T. It tokenizes all items via $\mathcal{T}$ and predicts the next item tokens via $\mathcal{R}$ conditioning on the tokenized history, rather than retrieving and then reranking from the full candidate set.

3.1 Residual-Quantization-based Item Tokenization

The tokenizer $\mathcal{T}$ maps each item's semantic embedding $\boldsymbol{z}$, obtained from pretrained text encoders such as Sentence-T5 (Ni et al., 2022) and BERT (Devlin et al., 2019), to a sequence of L discrete tokens, referred to as the item's *semantic identifier* (SID). A broad class of existing SID tokenizers constructs hierarchical identifiers using residual quantization (RQ). Let $\boldsymbol{r}$ denote the representation

to be quantized, which can be either the original item embedding or a learned transformation of it. At each level $l \in [L]$, RQ maintains a codebook $\mathcal{C}_l = \{\boldsymbol{e}_l^k\}_{k=1}^K$ with K codewords. Starting from $\boldsymbol{v}_1 = \boldsymbol{r}$, the nearest codeword is selected and directly subtracted from the current residual:

$$c_l = \arg\min_k \|\boldsymbol{v}_l - \boldsymbol{e}_l^k\|^2, \qquad \boldsymbol{v}_{l+1} = \boldsymbol{v}_l - \boldsymbol{e}_l^{c_l}. \tag{1}$$

This produces the hierarchical identifier $[c_1, \ldots, c_L]$ and the quantized representation $\tilde{\boldsymbol{r}} = \sum_{l=1}^L \boldsymbol{e}_l^{c_l}$.

Different RQ-based tokenizers instantiate this procedure differently. RQ-VAE (Rajput et al., 2023) first maps the semantic embedding to $\boldsymbol{r} = \mathrm{Encoder}_{\mathcal{T}}(\boldsymbol{z})$ and reconstructs $\tilde{\boldsymbol{z}} = \mathrm{Decoder}_{\mathcal{T}}(\tilde{\boldsymbol{r}})$. Its tokenizer is optimized with a reconstruction and quantization objective, $\mathcal{L}_{\mathrm{Token}} = \|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|^2 + \sum_{l=1}^L \Big(\|\mathrm{sg}(\boldsymbol{v}_l) - \boldsymbol{e}_l^{c_l}\|^2 + \beta \|\boldsymbol{v}_l - \mathrm{sg}(\boldsymbol{e}_l^{c_l})\|^2 \Big)$, where $\mathrm{sg}(\cdot)$ denotes stop-gradient and β balances the encoder and codebook updates. RQ-KMeans (Luo et al., 2025) constructs the codebooks through residual clustering instead, but uses the same residual update in eq. (1). Applying $\mathcal{T}$ to every item in S flattens the interaction history into a token sequence $X = [c_1^1, \ldots, c_L^1, \ldots, c_1^T, \ldots, c_L^T]$, while the target item i_{T+1} is represented by $Y = [c_1^{T+1}, \ldots, c_L^{T+1}]$.

Limitation from a decomposition perspective. The residual quantization above is suboptimal as shown in Figure 1(a). At each level, RQ selects a codeword direction but subtracts it with a fixed unit coefficient. This implicitly assumes that the current residual $\boldsymbol{v}_l$ contains exactly one unit of the selected codeword $\boldsymbol{e}_l^{c_l}$, which generally does not hold. As a result, the updated residual may still contain a component along the selected direction. Specifically,

$$\langle \boldsymbol{v}_{l+1}, \boldsymbol{e}_l^{c_l} \rangle = \langle \boldsymbol{v}_l - \boldsymbol{e}_l^{c_l}, \boldsymbol{e}_l^{c_l} \rangle = \langle \boldsymbol{v}_l, \boldsymbol{e}_l^{c_l} \rangle - \|\boldsymbol{e}_l^{c_l}\|^2, \tag{2}$$

which is generally nonzero unless the projection of $\boldsymbol{v}_l$ onto $\boldsymbol{e}_l^{c_l}$ has exactly unit coefficient. Therefore, information along the selected direction can remain in the residual and be re-encoded by later quantization levels. Moreover, it does not minimize the residual norm along the selected direction. For any nonzero codeword $\boldsymbol{e}$ and $\|\boldsymbol{e}\| = 1$, consider approximating $\boldsymbol{v}$ by $\lambda \boldsymbol{e}$. The optimal coefficient solves $\lambda^* = \arg\min_\lambda \|\boldsymbol{v} - \lambda \boldsymbol{e}\|^2 = \frac{\langle \boldsymbol{v}, \boldsymbol{e} \rangle}{\|\boldsymbol{e}\|^2}$ and we have

$$\|\boldsymbol{v} - \mathbf{1}\boldsymbol{e}\|^2 - \|\boldsymbol{v} - \lambda^* \boldsymbol{e}\|^2 = (1 - \langle \boldsymbol{v}, \boldsymbol{e} \rangle)^2 \geq 0. \tag{3}$$

This shows that the standard update can leave a larger-than-necessary residual. Both issues arise because the tokenizer records only the selected codeword index while discarding the magnitude of $\boldsymbol{v}_l$ along that codeword direction. Such redundancy may also make tokenizer optimization less stable in practice. This motivates our scale-aware residual quantization (SRQ) in Section 4.1.

3.2 Autoregressive Recommender

The recommender $\mathcal{R}$ generates the identifiers Y of the target item autoregressively from the tokenized history X, and is instantiated as either a Transformer encoder–decoder or a decoder-only language model. Existing generative recommenders differ in how $\mathcal{R}$ is trained, following one of three paradigms. *Two-stage* training (Rajput et al., 2023) learns the tokenizer and the recommender separately from scratch. *Unified* training (Kong et al., 2025) optimizes the tokenizer and the recommender jointly under the recommendation objective. *SFT-then-RL* training (Deng et al., 2025) finetunes $\mathcal{R}$ from a pretrained model, then sharpens its ranking ability via RL.

Across these paradigms, the supervised objective is the cross-entropy of the target identifier, $\mathcal{L}_{\mathrm{CE}} = -\sum_{l=1}^L \log P_\theta(Y_l \mid X, Y_{<l})$. The RL stage refines $\mathcal{R}$ by maximizing an expected reward r over generated identifiers, $\max_\theta \ \mathbb{E}_{Y \sim \pi_\theta(\cdot|X)}[r(X, Y)]$, where π_θ is the recommender's generation policy parameterized by θ. We study our method under all three paradigms.

4 Method

The limitation in Section 3.1 stems from discarding the magnitude of the residual along each selected direction. We therefore make this magnitude explicit that at every level, we retain not only *which* codeword is selected but also *how much* of it is present. We call the resulting method SIGER (*Scale-aware Identifiers for GEnerative Recommendation*). SIGER computes and stores the per-level SIDs plus the magnitudes in the tokenizer $\mathcal{T}$ (Section 4.1), trains the recommender $\mathcal{R}$ to predict the SIDs together with theirs quantized magnitudes (Section 4.2), and instantiates under all three training paradigms (Section 4.3). The overview of the method is shown in Figure 1.

4.1 SCALE-AWARE RESIDUAL QUANTIZATION (SRQ)

SRQ generalizes residual quantization by subtracting, at each level, the selected codeword scaled by its projection coefficient rather than at a unit coefficient. It measures how much of the residual lies along each codeword and retains the most dominant direction together with its magnitude. For the residual $\boldsymbol{v}_l$ (with $\boldsymbol{v}_1 = \boldsymbol{r}$), the portion of $\boldsymbol{v}_l$ lying along a codeword $\boldsymbol{e}_l^k$ is the projection coefficient $\lambda_l^k = \langle \boldsymbol{v}_l, \boldsymbol{e}_l^k \rangle / \|\boldsymbol{e}_l^k\|^2$, whose scaled codeword $\lambda_l^k \boldsymbol{e}_l^k$ is the component of $\boldsymbol{v}_l$ along that direction. We select the codeword that accounts for the largest such component and subtract it:

$$c_l = \arg\min_{k \in [K]} \left\| \boldsymbol{v}_l - \lambda_l^k \boldsymbol{e}_l^k \right\|^2, \qquad \lambda_l = \lambda_l^{c_l}, \qquad \boldsymbol{v}_{l+1} = \boldsymbol{v}_l - \lambda_l \, \boldsymbol{e}_l^{c_l}. \tag{4}$$

The retained magnitude λ_l minimizes the level-l residual and leaves it orthogonal to the selected codeword, $\langle \boldsymbol{v}_{l+1}, \boldsymbol{e}_l^{c_l} \rangle = \langle \boldsymbol{v}_l, \boldsymbol{e}_l^{c_l} \rangle - \lambda_l \|\boldsymbol{e}_l^{c_l}\|^2 = 0$, resolving both deficiencies of Section 3.1. Besides, equation 4 recovers RQ as the special case $\lambda_l \equiv 1$.

When SRQ is applied to a VAE-based tokenizer, yielding *SRQ-VAE*, each item is thereby decomposed into L discrete directions $[c_1, \ldots, c_L]$ and L continuous magnitudes $[\lambda_1, \ldots, \lambda_L]$, with quantized latent $\tilde{\boldsymbol{r}} = \sum_{l=1}^{L} \lambda_l \boldsymbol{e}_l^{c_l}$ and reconstruction $\tilde{\boldsymbol{z}} = \text{Decoder}_{\mathcal{T}}(\tilde{\boldsymbol{r}})$. SRQ-VAE is trained with the RQ-VAE objective over the scaled codewords,

$$\mathcal{L}_{\text{Token}} = \|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|^2 + \sum_{l=1}^{L} \Big(\|\text{sg}(\boldsymbol{v}_l) - \lambda_l \boldsymbol{e}_l^{c_l}\|^2 + \beta \|\boldsymbol{v}_l - \text{sg}(\lambda_l \boldsymbol{e}_l^{c_l})\|^2 \Big), \tag{5}$$

where $\text{sg}(\cdot)$ is the stop-gradient and a straight-through estimator passes gradients through the $\arg\min$ selection. Consequently, SRQ-VAE stores each item $i \in \mathcal{I}$ as its identifier together with the corresponding magnitudes, i.e., the sequence of per-level pairs $[c_1, c_2 \ldots, c_L, \lambda_1, \lambda_2, ..., \lambda_L]$.

4.2 GENERATIVE RECOMMENDER

Under SIGER, the recommender predicts, for the target item i_{T+1}, both its codewords $[c_1, \ldots, c_L]$ and its magnitudes $[\lambda_1, \ldots, \lambda_L]$. The codewords are discrete and are generated autoregressively as in standard generative recommendation (Rajput et al., 2023). The magnitudes, however, are continuous and do not fit this discrete generation as they cannot be emitted from the identifier vocabulary. Predicting them by regression is also unsuitable, as a squared-error objective is scale-sensitive and dominated by the levels of larger magnitude.

Quantile binning. To reconcile the two codewords and magnitudes, we discretize each magnitude λ_l into B bins with a quantile function $Q(\cdot)$. Cutting the level-l magnitudes of the $N = |\mathcal{I}|$ catalogue items at their $1/B, 2/B, \ldots, (B-1)/B$ quantiles gives boundaries $\tau_{l,1} < \cdots < \tau_{l,B-1}$, and an item's bin is the interval containing its magnitude,

$$b_l = Q_l(\lambda_l^{c_l}) = \#\{\, k : \lambda_l^{c_l} \geq \tau_{l,k} \,\} \in \{0, \ldots, B-1\}. \tag{6}$$

Each bin stores the mean magnitude of its members, $\bar{\lambda}_{l,b}$, so that a bin decodes back to a scalar. By construction every bin holds about N/B items, so the boundaries are denser where λ_l is concentrated. Each item thus becomes a homogeneous sequence of $2L$ discrete tokens,

$$\text{SIGER-ID} = (c_1, \ldots, c_L, b_1, \ldots, b_L) \in [K]^L \times [B]^L, \tag{7}$$

where magnitude tokens naturally reduce items collision rate as a side-effect.

Recommender training. The SIGER-ID is homogeneous: codewords and magnitude bins are both discrete tokens, differing only in the vocabulary attached to each position. Writing it as $\boldsymbol{t} = (t_1, \ldots, t_{2L})$, this vocabulary is $\mathcal{V}_j = \{1, \ldots, K\}$ if $1 \leq j \leq L, \mathcal{V}_j = \{0, \ldots, B-1\}$ if $L < j \leq 2L$, so $\mathcal{R}$ is a K-way classifier at the codeword positions and a B-way classifier at the magnitude positions. It predicts magnitudes through the same autoregressive mechanism as codewords and without architectural change. The history X is flattened into these tokens, and $\mathcal{R}$ generates the target identifier: at step j it forms a hidden state $\boldsymbol{d}_j$ from X and $\boldsymbol{t}_{<j}$, and assigns

$$P(t_j = v \mid \boldsymbol{t}_{<j}, X) = \frac{\exp(\boldsymbol{w}_v^\top \boldsymbol{d}_j)}{\sum_{u \in \mathcal{V}_j} \exp(\boldsymbol{w}_u^\top \boldsymbol{d}_j)}, \qquad v \in \mathcal{V}_j, \tag{8}$$

where $\boldsymbol{w}_v$ is the output embedding of token v. No regression head or auxiliary magnitude loss is required: training minimizes a single cross-entropy over the $2L$ positions, $\mathcal{L}_{\text{CE}} = -\sum_{j=1}^{2L} \log P\big(t_j = t_j^* \mid \boldsymbol{t}_{<j}, X\big)$, with $\boldsymbol{t}^*$ the ground-truth identifier from the tokenizer. At inference, constrained beam search over valid identifiers retrieves the next item, matching both its codewords and its magnitudes.

Trade-offs. Quantile binning first introduces a *representation* trade-off. Discretization sacrifices the precision of λ_l, and recovering it requires a larger bin count B, which monotonically reduces both the quantization error $\varepsilon_d(B) = (\mathbb{E}[(\lambda_l - \bar{\lambda}_{l,b_l})^2])^{1/2}$ and the identifier collision rate (Figure 2). The remaining two trade-offs stem from the identifier doubling to $2L$ tokens: the history X and the generation target occupy extra positions, which incurs *overhead*. Besides, $\mathcal{R}$ must emit L additional tokens correctly, which raises the *prediction difficulty*. The former is modest, since L is short in practice ($L = 3$); for the latter, we show that the standard backbone retains sufficient capacity to predict the magnitude tokens. We analyze all three effects in Section 5.1.

4.3 Extension to Other Paradigms

SIGER can be plugged-and-played into both unified and LLM-based post-training paradigms.

Unified training. In unified training, we relax the hard codeword selection into a differentiable assignment distribution. Different from the soft identifier formulation in (Li et al., 2026), where the assignment probability is computed from the distance between the residual and each codeword, SIGER uses the distance to the corresponding scaled codeword. Specifically, for each level l, after computing the coefficient λ_l^k for each codeword $\boldsymbol{e}_l^k$, we define the assignment probability, and relaxed sclae-aware quantized residual with temperature τ as

$$p_l^k = \frac{\exp\left(-\|\boldsymbol{v}_l - \lambda_l^k \boldsymbol{e}_l^k\|^2/\tau\right)}{\sum_{j=1}^{K} \exp\left(-\|\boldsymbol{v}_l - \lambda_l^j \boldsymbol{e}_l^j\|^2/\tau\right)}; \qquad \bar{\boldsymbol{q}}_l = \sum_{k=1}^{K} p_l^k \lambda_l^k \boldsymbol{e}_l^k, \qquad \boldsymbol{v}_{l+1} = \boldsymbol{v}_l - \bar{\boldsymbol{q}}_l. \tag{9}$$

This relaxation allows the codeword assignment to be optimized jointly with the recommender with suitable recommendation objectives. For the magnitude tokens, they are obtained after the scale coefficients are computed and assigned to quantile bins using the fixed bin boundaries. Therefore, we keep the magnitude-bin indices deterministic and do not apply a soft relaxation to them.

SFT-then-RL training. The SFT-then-RL paradigm finetunes pre-trained models, and we replace the original SID tokenizer with our scale-aware tokenizer while keeping the LLM post-training procedure unchanged. In the SFT stage, the LLM is trained with the standard next-token prediction objective over the target SIGER identifier. In the RL stage, given a user prompt x, the policy generates a group of G valid candidate identifiers $\{y^{(i)}\}_{i=1}^{G}$ with constrained decoding. Each candidate is mapped to an item and assigned a reward R_i, and the group-normalized advantage is $\hat{A}_i = \frac{R_i - \mu_{1:G}}{\sigma_{1:G}}$, where $\mu_{1:G}$ and $\sigma_{1:G}$ are the mean and standard deviation of rewards in the group. The model is optimized with the clipped GRPO objective Guo et al. (2025): $\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\left(w_i \hat{A}_i, \text{clip}(w_i, 1-\epsilon, 1+\epsilon)\hat{A}_i\right) - \beta\, \text{KL}(\pi_\theta \| \pi_{\text{ref}})\right]$, where $w_i = \frac{\pi_\theta(y^{(i)}|x)}{\pi_{\theta_{\text{old}}}(y^{(i)}|x)}$. We use the same hybrid reward as MiniOneRec (Kong et al., 2025), combining a rule-based reward for generating the ground-truth item and a rank-aware reward that penalizes high-ranked negative candidates. The complete formulation is provided in Appendix A.2:

$$R(i_k, i_t) = R_{\text{rule}}(i_k, i_t) + R_{\text{rank}}(i_k, i_t), \tag{10}$$

where i_k and i_t are the generated item and ground-truth, respectively. In current method, we have not introduced additional rewards design for magnitude-bin tokens, and leave it for future work.

5 Experiments

Evaluation protocols and datasets. We evaluate SIGER under three representative generative recommendation settings, using the corresponding public sequential recommendation benchmarks from prior work. Following GRID (Ju et al., 2025), we first conduct a modular SID-tokenizer comparison on three Amazon Review categories, Beauty, Toys, and Sports (Hou et al., 2026), where the downstream generative recommender is fixed and only the identifier construction method is

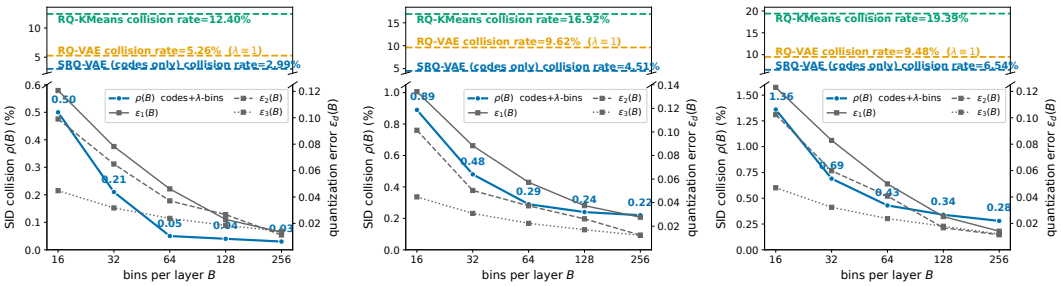


Figure 2: SID collision rate $\rho(B)$ and magnitude quantization error $\epsilon_d(B)$ under different magnitude-bin budgets on Beauty, Toys, and Sports, comparing RQ-KMeans, RQ-VAE, and code-only SRQ-VAE. Magnitude-bin tokens further reduce $\rho(B)$ and $\epsilon_d(B)$.

changed. Following UniGRec (Li et al., 2026), we then evaluate SIGER in the unified tokenizer–recommender optimization setting with the same preprocessing and data splits. Following MiniOneRec (Kong et al., 2025), we further evaluate SIGER in the LLM-based generative recommendation pipeline on the Industrial and Office datasets from Amazon Reviews, with supervised fine-tuning and recommendation-oriented post-training. Across all settings, user interactions are ordered chronologically, and the task is next-item prediction from historical interaction sequences.

Baselines. We compare SIGER with both traditional sequential recommenders and recent generative recommenders. Traditional baselines include Caser (Tang & Wang, 2018), GRU4Rec (Hidasi et al., 2016), SASRec (Kang & McAuley, 2018), and BERT4Rec (Sun et al., 2019). For generative recommendation, we compare with TIGER (Rajput et al., 2023), LETTER (Wang et al., 2024a), EAGER (Wang et al., 2024c), OneRec (Deng et al., 2025) DiscRec (Liu et al., 2025a), UniGRec (Li et al., 2026), and MiniOneRec (Kong et al., 2025). In the tokenizer-level comparison, we additionally compare different identifier construction methods, including RK-Means, R-VQ, and RQ-VAE (Zeghidour et al., 2022). All baselines are evaluated under their corresponding original protocols when available, and SIGER is inserted into the same pipeline by replacing the standard SID tokenizer with our scale-aware tokenizer.

Models and metrics. For the two-stage experiments, we use Flan-T5-XL to extract item semantic embeddings and an encoder-decoder Transformer as the generative recommender, following the default GRID configuration with 4 encoder layers, 4 decoder layers, 6 attention heads, 128 hidden dimensions, and 1024 MLP dimensions (Ju et al., 2025). For the unified experiments, we use the same backbone and training configuration as UniGRec (Li et al., 2026). For the SFT-RL experiments, we use Qwen-1.5B as the LLM backbone following MiniOneRec (Kong et al., 2025). We report Recall@3/5/10 and NDCG@3/5/10 and trainings are conducted on 8 NVIDIA A100 GPUs.

5.1 Codebook Analysis

Decomposition Quality and Coverage. Figure 4(a) in the appendix directly verifies the geometric limitation discussed in Section 3.1. Across Beauty, Toys, and Sports, RQ-VAE leaves non-negligible correlation between the selected codeword and the next residual, with mean absolute cosine values around 0.15–0.26 across layers and maximum values up to 0.89. In contrast, SRQ-VAE subtracts the projected component $\lambda_l e_l^{c_l}$, making the residual orthogonal to the selected codeword for every item by construction. Figure 4(b) further shows that SRQ-VAE improves codebook coverage. At the first level, RQ-VAE uses only $158/256$, $117/256$, and $134/256$ codewords on Beauty, Toys, and Sports, respectively, while SRQ-VAE uses all 256 codewords at every level on all three datasets. Finally, Figure 2 shows that better decomposition and magnitude tokens lead to lower identifier collision. With codeword tokens only, SRQ-VAE reduces the collision rate: $5.26\% \rightarrow 2.99\%$ on Beauty, $9.62\% \rightarrow 4.51\%$ on Toys, and $9.48\% \rightarrow 6.54\%$ on Sports. Adding magnitude-bin tokens further reduces the collision rate consistently: at $B = 32$, the collision rates drop to 0.207%, 0.478%, and 0.686% on Beauty, Toys, and Sports, respectively. The collision rate continues to decrease as B increases, while the magnitude quantization error also becomes smaller, suggesting that scale-aware identifiers improve item resolution with a moderate bin budget.

Table 1: Performance of GR models with SIDs generated by different tokenization algorithms. The top-1/2/3 results are highlighted in red, yellow, and green, respectively.

Method	Codebook level	Recommendation Performance (Recall@5/Recall@10/NDCG@5/NDCG@10)											
		Beauty				Toys				Sports			
RK-Means	3	0.0422	0.0639	0.0277	0.0347	0.0376	0.0577	0.0243	0.0308	0.0236	0.0353	0.0153	0.0191
R-VQ		0.0422	0.0638	0.0282	0.0351	0.0327	0.0493	0.0209	0.0262	0.0234	0.0352	0.0151	0.0189
RQ-VAE		0.0404	0.0593	0.0268	0.0329	0.0342	0.0514	0.0224	0.0280	0.0205	0.0312	0.0132	0.0166
SRQ-VAE		0.0426	0.0642	0.0295	0.0365	0.0383	0.0552	0.0252	0.0308	0.0240	0.0359	0.0155	0.0194
SRQ-VAE (code-only)		0.0433	0.0632	0.0291	0.0355	0.0390	0.0581	**0.0265**	**0.0326**	0.0237	0.0353	0.0160	**0.0198**
RQ-VAE	2	0.0365	0.0527	0.0252	0.0304	0.0335	0.0489	0.0226	0.0276	0.0218	0.0325	0.0142	0.0177
SRQ-VAE		**0.0444**	**0.0647**	**0.0310**	**0.0369**	**0.0412**	**0.0600**	0.0263	0.0323	**0.0241**	**0.0363**	**0.0162**	**0.0198**
SRQ-VAE (code-only)		0.0415	0.0593	0.0287	0.0344	0.0365	0.0528	0.0250	0.0302	0.0220	0.0341	0.0148	0.0187

Geometry and Usage. Moreover, we inspect the geometry and usage of the learned codebooks in Figure 3. The left-column projects the ℓ_2-normalized codewords of each level residual into two dimensions with PCA, which highlights directional diversity. At the first level, RQ-VAE forms a compact cluster, indicating that many codewords represent similar directions, while SRQ-VAE spreads more broadly across the projection space. The right-column usage curves show the same pattern quantitatively: RQ-VAE exhibits a sharp usage fraction cliff at rank 158, leaving 98 of 256 codewords unused and reaching entropy $H = 6.97$ bits, whereas SRQ-VAE has a flatter usage distribution with entropy $H = 7.73$ bits. This improved coverage helps explain why SRQ-VAE already reduces identifier collisions with codeword tokens alone in Figure 2. At deeper levels, the two tokenizers show more similar geometry and usage, suggesting that the main utilization gap is concentrated at the first residual level.

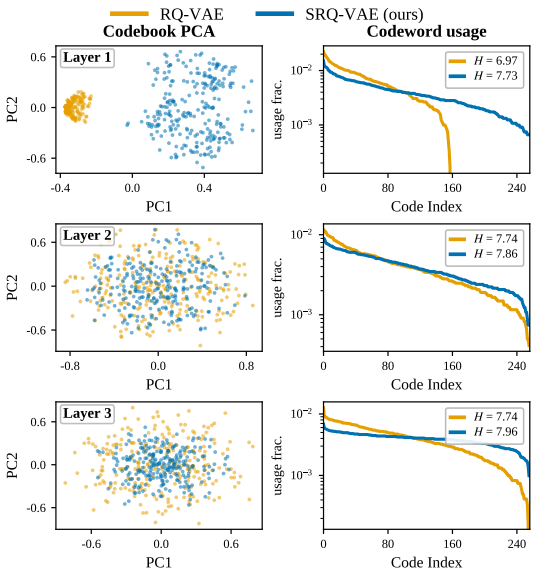


Figure 3: Codebook geometry and usage comparison between RQ-VAE and SRQ-VAE. Left: 2D PCA of ℓ_2-normalized codewords. Right: sorted codeword usage in log scale, with entropy H.

Magnitude distribution and binning. Figure 5 in the appendix shows that λ_l has non-uniform, level-dependent distributions: λ_1 is mostly positive, while λ_2 and λ_3 are more centered around zero. Quantile binning is therefore a natural choice, as it places more bin boundaries in dense regions of the distribution and provides finer resolution where most magnitudes occur.

5.2 Main Results

Two-stage training. Table 1 compares different SID construction methods using Flan-T5-XL item embeddings and an 8-layer Transformer encoder–decoder recommender (Ju et al., 2025). Overall, SRQ variants consistently outperform the baseline tokenizers across Beauty, Toys, and Sports. Compared with the strongest RQ baseline, the best SRQ variant improves Recall@5 by 5.2%, 9.6%, and 2.1% on Beauty, Toys, and Sports, respectively. For NDCG@5, the relative gains are 9.9%, 9.1%, and 5.9%, respectively. Similar gains are observed on Recall@10 and NDCG@10, showing that scale-aware identifiers provide stronger item representations.

Compact Identifiers. A key advantage of SRQ is that it remains effective with fewer codebook levels. As shown in Table 3, RQ-VAE with 2-level codebooks has very high collision rates: 43.44% on Beauty, 49.83% on Toys, 53.47% on Sports. This severe collision increase is reflected in Table 1, where RQ-VAE-2lvl performs worse than 3-level RQ-VAE on most metrics. In contrast, 2-level SRQ-VAE keeps the collision rate much lower, with only 1.68%, 2.31%, and 3.28% with B=32. It also outperforms 3-level SRQ-VAE on all metrics, suggesting that scale-aware magnitude tokens preserve item resolution while the shorter target sequence makes generation easier.

Table 2: The overall performance comparisons between the baselines and **SIGER-U**. The top-1/2/3 results are highlighted in red, yellow, and green, respectively.

	Beauty				Pet			
Method	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10
Traditional Methods								
Caser	0.0252	0.0415	0.0155	0.0208	0.0210	0.0369	0.0132	0.0183
GRU4Rec	0.0397	0.0601	0.0270	0.0336	0.0300	0.0504	0.0188	0.0254
SASRec	0.0284	0.0550	0.0172	0.0244	0.0237	0.0425	0.0142	0.0202
BERT4Rec	0.0322	0.0478	0.0219	0.0270	0.0199	0.0337	0.0124	0.0169
Generative Methods								
LETTER	0.0444	0.0699	0.0288	0.0370	0.0357	0.0590	0.0231	0.0306
EAGER	0.0399	0.0536	0.0289	0.0333	0.0282	0.0404	0.0191	0.0230
OneRec	0.0375	0.0567	0.0251	0.0312	0.0334	0.0539	0.0216	0.0282
ETEG-Rec	0.0525	0.0809	0.0357	0.0448	0.0450	0.0693	0.0292	0.0370
DiscRec-T	0.0513	0.0765	0.0346	0.0427	0.0390	0.0624	0.0257	0.0333
UniGRec	0.0548	0.0825	0.0368	0.0457	0.0470	0.0701	0.0315	0.0389
SIGER-U	0.0555	0.0826	**0.0377**	**0.0464**	**0.0484**	**0.0757**	0.0312	0.0399
SIGER-U-2lvl	0.0544	0.0828	0.0357	0.0463	0.0476	0.0738	**0.0317**	**0.0401**
SIGER-U (code-only)	**0.0560**	**0.0834**	0.0375	0.0463	0.0443	0.0716	0.0294	0.0381

Efficiency and Ablations. Table 3 further compares token length and training cost. Adding magnitude tokens increases the number of tokens per item from 3 to 5 in the 2-level setting and from 4 to 7 in the 3-level setting, with relative training time increasing to $1.17\times$ and $1.28\times$, respectively. To isolate the effect of scale-aware decomposition without extra magnitude tokens, we also evaluate code-only SRQ variants, which use the same number of tokens as RQ-VAE. In the 2-level setting, code-only SRQ-VAE improves over RQ-VAE-2lvl on all datasets. For example, Recall@5 increases from 0.0365 to 0.0415 on Beauty, from 0.0335 to 0.0365 on Toys, and from 0.0218 to 0.0220 on Sports. The 3-level code-only variant is stronger, achieving the best NDCG@5/10 on Toys and the best NDCG@10 on Sports. These results indicate that scaled residual decomposition improves the codeword hierarchy itself, while magnitude tokens provide higher accuracy.

Unified training. We next evaluate SIGER under the unified tokenizer–recommender training setting. Table 2 reports the results on Beauty and Pet, where SIGER-U denotes SIGER integrated into the unified training pipeline. On Beauty, the best SIGER-U variant improves over UniGRec by 2.2% in Recall@5, 1.1% in Recall@10, 2.4% in NDCG@5, and 1.5% in NDCG@10. On Pet, the best SIGER-U variant improves over UniGRec by 3.0% in Recall@5, 8.0% in Recall@10, 0.6% in NDCG@5, and 3.1% in NDCG@10. The gains are moderate because the SASRec teacher mainly supervises codeword token prediction, without directly guiding the additional magnitude-bin tokens. Nevertheless, the consistent improvements suggest that scale-aware identifiers are compatible with unified training and can provide complementary gains beyond soft identifier optimization.

LLM-based SFT-then-RL training. We further evaluate SIGER under the LLM-based SFT-then-RL setting in Table 5. On Industrial, SIGER achieves the best Recall@5, Recall@10, and NDCG@10, while remaining competitive on the other metrics. On Office, SIGER achieves the best performance across all six metrics, improving over MiniOneRec by 5.9%, 4.2%, 1.3%, 2.0%, 2.5%, and 2.4% on Recall@3, NDCG@3, Recall@5, NDCG@5, Recall@10, and NDCG@10, respectively. The gap between SIGER-SFT and SIGER with RL+SFT also shows that RL post-training is important for aligning generated identifiers with the final recommendation objective.

Generalization to RQ methods. We also apply the scale-aware residual update to RQ-KMeans, yielding SRQ-KMeans. As shown in Table 3 and Table 4 in the appendix, SRQ-KMeans achieves a much lower collision rate and competitive performance, suggesting that scale-aware decomposition can generalize to other residual-quantization tokenizers beyond VAE-based SIDs.

6 Conclusion

We propose SIGER, a scale-aware semantic identifier framework for generative recommendation. By recording both the selected codeword direction and its projection magnitude, SIGER improves residual decomposition and produces more informative item identifiers. Experiments across two-stage, unified, and LLM-based training settings show that SIGER improves identifier quality, reduces SID collisions, and consistently enhances recommendation performance.

AI USE STATEMENT

In this work, we used generative AI tools only for language editing and revision of author-written text, including improving grammar, clarity, and presentation. We did not use generative AI tools for developing research ideas or designing the methodology. All AI-assisted language revisions were carefully reviewed by the authors to ensure that they preserved the intended technical meaning and did not introduce unsupported claims or inaccuracies. We take full responsibility for the final content of this work, including any text revised with the aid of generative AI.

REFERENCES

Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

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. *CoRR*, abs/2502.18965, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

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 *RecSys*, 2022.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian J. McAuley. Bridging language and items for retrieval and recommendation. *CoRR*, 2024.

Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation: Benchmarking llms as semantic encoders. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3251–3265, 2026.

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.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

Xiaoyu Kong, Leheng Sheng, Junfei Tan, Yuxin Chen, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Minionerec: An open-source framework for scaling generative recommendation. *arXiv preprint arXiv:2510.24431*, 2025.

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.

Chang Liu, Yimeng Bai, Xiaoyan Zhao, Yang Zhang, Fuli Feng, and Wenge Rong. Discrec: Disentangled semantic-collaborative modeling for generative recommendation. *arXiv preprint arXiv:2506.15576*, 2025a.

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, 2025b.

Xinchen Luo, Jiangxia Cao, Tianyu Sun, Jinkai Yu, Rui Huang, Wei Yuan, Hezheng Lin, Yichen Zheng, Shiyao Wang, Qigen Hu, et al. Qarm: Quantitative alignment multi-modal recommendation at kuaishou. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5915–5922, 2025.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the association for computational linguistics: ACL 2022*, pp. 1864–1874, 2022.

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 Mahesh Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 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*, pp. 1441–1450, 2019.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.

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, 2024a.

Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, and Zhenhua Dong. EAGER: two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pp. 3245–3254. ACM, 2024b.

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*, pp. 3245–3254, 2024c.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30, 2022.

Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, Weifeng Ding, Wuchao Li, Xinchen Luo, Xingmei Wang, Zexuan Cheng, Zixing Zhang, Bin Zhang, Boxuan Wang, Chaoyi Ma, Chengru Song, Chenhui Wang, Di Wang, Dongxue Meng, Fan Yang, Fangyu Zhang, Feng Jiang, Fuxing Zhang, Gang Wang, Guowang Zhang, Han Li, Hengrui Hu, Hezheng Lin, Hongtao Cheng, Hongyang Cao, Huanjie Wang, Jiaming Huang, Jiapeng Chen, Jiaqiang Liu, Jinghui Jia, Kun Gai, Lantao Hu, Liang Zeng, Liao Yu, Qiang Wang, Qidong Zhou, Shengzhe Wang, Shihui He, Shuang Yang, Shujie Yang, Sui Huang, Tao Wu, Tiantian He, Tingting Gao, Wei Yuan, Xiao Liang, Xiaoxiao Xu, Xugang Liu, Yan Wang, Yi Wang, Yiwu Liu, Yue Song, Yufei Zhang, Yunfan Wu, Yunfeng Zhao, and Zhanyu Liu. Onerec technical report. *CoRR*, abs/2506.13695, 2025.

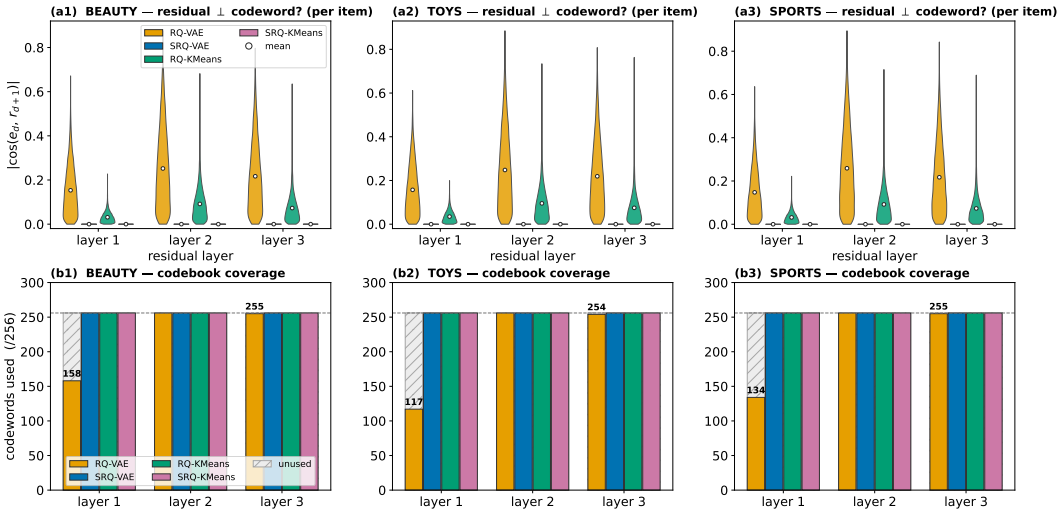


Figure 4: Codebook decomposition quality comparison among RQ-VAE, SRQ-VAE, RQ-KMeans, and SRQ-KMeans. **(a)** SRQ-VAE yields orthogonal residuals, while RQ-VAE and RQ-KMeans leave nonzero residual–codeword correlation. **(b)** SRQ-VAE fully uses the codebook at every level, whereas RQ-VAE collapses, epecially at level 1.

A Experimental Details

A.1 Hyperparameter Choices

We summarize the main hyperparameter choices used in our experiments. For two-stage training, item embeddings are extracted with a frozen Flan-T5-XL encoder with dimension 2048. The SID tokenizer is trained from scratch using an MLP autoencoder with a 64-dimensional latent space, codebook size 256 per level, and either two or three codebook levels. For SRQ-VAE, we select the magnitude-bin size $B \in \{16, 32, 64, 128\}$. The tokenizer is trained for 15,000 steps with learning rate 10^{-3}. The two-stage recommender is a TIGER-style encoder–decoder model trained from scratch with hidden size 128, 4 encoder layers, 4 decoder layers, 6 attention heads, dropout 0.15, learning rate 10^{-3}, and weight decay 10^{-4}. We train it for 50,000 steps and use constrained beam search over valid SID sequences for evaluation.

For unified tokenizer–recommender training, we first pretrain the SRQ-VAE tokenizer and then jointly optimize it with the recommender. The tokenizer uses Sentence-T5-base item embeddings, codebook size 256 per level, latent dimension 32; the main setting uses three codebook levels, while two-level variants are used for compactness analysis. The tokenizer is pretrained with Adam using learning rate 10^{-3} and batch size 1024. The unified recommender uses a T5 encoder–decoder backbone with 6 encoder layers, 6 decoder layers, hidden size 128, feed-forward dimension 512, 4 attention heads, and dropout 0.1. For SIGER, the identifier contains three codeword tokens and three magnitude-bin tokens with $B = 32$ quantile bins. During joint training, we use batch size 512, T5 learning rate 5×10^{-3}, tokenizer learning rate 2×10^{-8}, and early stopping based on validation NDCG@10. Evaluation is performed with constrained beam search using beam size 30.

A.2 Reward Formulation for SFT-then-RL Training

In the SFT-then-RL experiments, we use the same item-level hybrid reward as MiniOneRec (Kong et al., 2025). Given the ground-truth item i_t and a generated candidate item i_k, the reward is defined as

$$R(i_k, i_t) = R_{\text{rule}}(i_k, i_t) + R_{\text{rank}}(i_k, i_t), \tag{11}$$

where the rule-based reward measures whether the generated item matches the ground truth:

$$R_{\text{rule}}(i_k, i_t) = \begin{cases} 1, & i_k = i_t, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Table 3: Raw SID collision rate (%, lower is better) on three Amazon datasets. Tokens/item includes the deduplication token. Training time is relative per-batch time normalized by 3-level RQ-VAE. SRQ-VAE and SRQ-KMeans are with magnitude tokens uses B=32 quantile bins.

Tokenizer	Levels	Tokens/item	Collision Rate (%)			Training time
			Beauty	Toys	Sports	
RQ-VAE	2	3	43.44	49.83	53.47	0.86×
SRQ-VAE (codes only)			28.41	32.61	41.64	
SRQ-VAE (B=32)		5	**1.68**	**2.31**	**3.28**	1.17×
RQ-KMeans	3	4	12.40	16.92	19.39	1.00×
SRQ-KMeans ($B = 32$)			**0.12**	**0.39**	**0.45**	
RQ-VAE			5.26	9.62	9.48	
SRQ-VAE (codes only)			2.99	4.51	6.54	
SRQ-VAE (B=32)		7	**0.21**	**0.48**	**0.69**	1.28×

The rank-aware component penalizes high-ranked negative candidates. Let ρ_k denote the generation-probability rank of candidate item i_k, where $\rho_k = 1$ is the highest-ranked candidate. We first compute

$$\tilde{R}_{\text{rank}}(i_k, i_t) = \begin{cases} 0, & i_k = i_t, \\ -\frac{1}{\log(\rho_k+1)}, & \text{otherwise}, \end{cases} \tag{13}$$

and normalize it within the generated candidate group:

$$R_{\text{rank}}(i_k, i_t) = -\frac{\tilde{R}_{\text{rank}}(i_k, i_t)}{\sum_{j=1}^{G} \tilde{R}_{\text{rank}}(i_j, i_t)}. \tag{14}$$

In our current implementation, we have not introduced additional rewards for magnitude-bin tokens.

A.3 Additional results

Residual orthogonality and codebook coverage. Figure 4 compares the decomposition quality and codebook usage of RQ-VAE, SRQ-VAE, RQ-KMeans and its scale-aware variant, SRQ-KMeans. For residual orthogonality, RQ-VAE leaves clear residual–codeword correlation, with average absolute cosine values of $0.153/0.252/0.217$ on Beauty, $0.157/0.248/0.219$ on Toys, and $0.148/0.259/0.217$ on Sports across levels 1–3. In contrast, SRQ-VAE reduces these values to nearly 0 by construction, indicating that the selected codeword direction is effectively removed from the residual. Meanwhile, RQ-KMeans could not result in complete orthogonal residuals either. For codebook coverage, RQ-VAE uses only $158/256$, $117/256$, and $134/256$ first-level codewords on Beauty, Toys, and Sports, respectively, while SRQ-VAE uses all $256/256$ codewords at every level. These results show that scale-aware decomposition improves both residual quality and codebook utilization.

Collision rate and training cost. Table 3 shows that SRQ-VAE substantially reduces SID collisions with moderate overhead. In the two-level setting, RQ-VAE has high collision rates of 43.44%–53.47%, while SRQ-VAE with B=32 reduces them to 1.68%–3.28%. In the three-level setting, SRQ-VAE further lowers collisions to below 0.7% on all datasets. Code-only SRQ-VAE also improves over RQ-VAE with the same token length, indicating that scaled residual decomposition improves the codeword hierarchy itself, while magnitude tokens provide additional identifier resolution at a moderate training-cost increase.

Magnitude distribution and quantile binning. Figure 5 visualizes the learned projection coefficients λ on Beauty and the corresponding quantile bin boundaries. The distributions vary across residual levels: first-level coefficients are mostly positive, while deeper-level coefficients are more centered around zero. This suggests that different levels capture different residual patterns, making fixed-width binning less suitable. Quantile binning adapts to the empirical distribution at each level by allocating more bins to dense regions, which provides finer magnitude resolution and helps reduce SID collisions as the bin budget B increases.

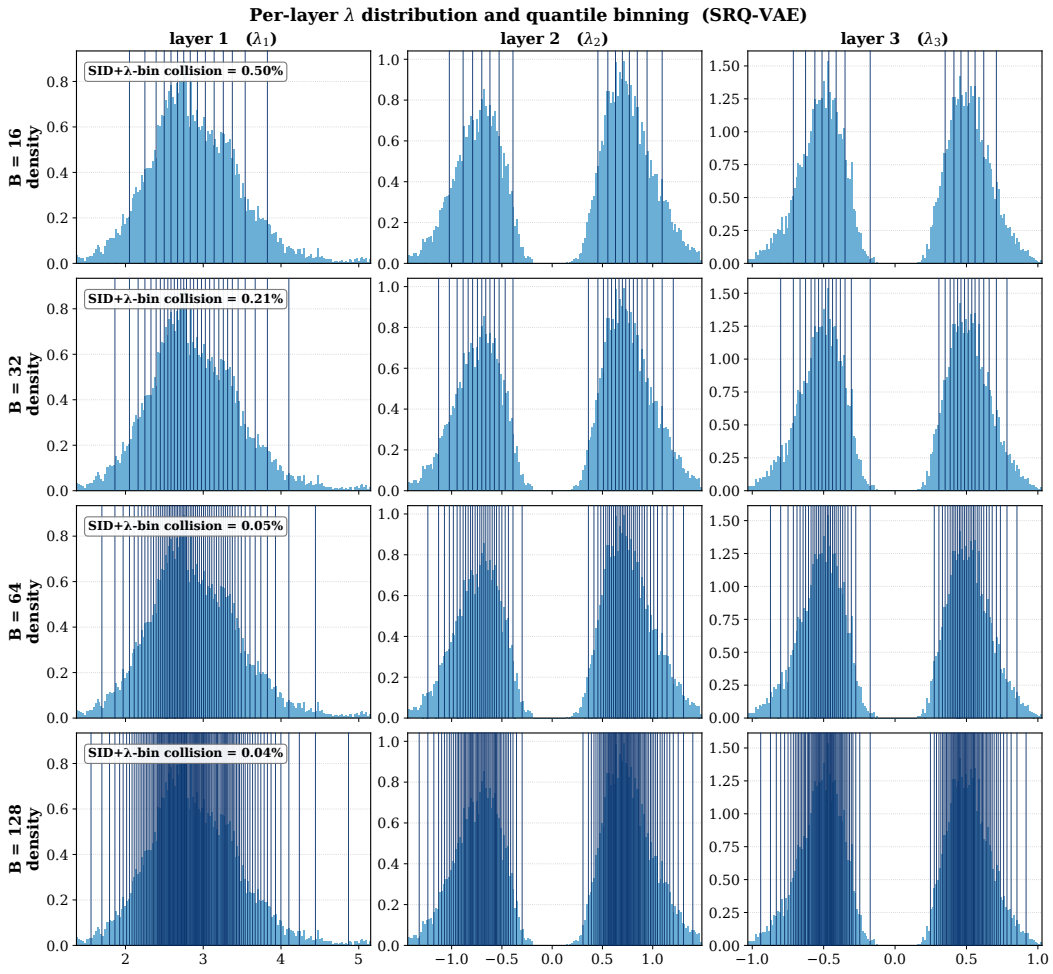


Figure 5: Per-level λ distributions and quantile binning on Beauty. Columns correspond to residual levels $l = 1, 2, 3$, and rows use bin budgets $B \in \{16, 32, 64, 128\}$. Blue vertical lines denote quantile bin boundaries. The first-level magnitudes are mostly positive, while deeper levels are more centered around zero. Quantile binning allocates finer resolution to dense regions of the λ distribution and reduces identifier collisions as B increases.

Table 4: Generalization of scale-aware decomposition to different residual-quantization tokenizers. All methods use 3 codebook levels.

Method	Recommendation Performance (Recall@5/Recall@10/NDCG@5/NDCG@10)											
	Beauty				Toys				Sports			
RQ-VAE	0.0404	0.0593	0.0268	0.0329	0.0342	0.0514	0.0224	0.0280	0.0205	0.0312	0.0132	0.0166
SRQ-VAE	**0.0426**	**0.0642**	**0.0295**	**0.0365**	**0.0383**	**0.0552**	**0.0252**	**0.0308**	**0.0240**	**0.0359**	**0.0155**	**0.0194**
RQ-KMeans	0.0416	0.0635	0.0275	0.0346	0.0382	0.0572	0.0241	0.0302	0.0226	0.0346	0.0147	0.0186
SRQ-KMeans	**0.0430**	**0.0660**	**0.0293**	**0.0364**	**0.0396**	**0.0589**	**0.0265**	**0.0327**	**0.0233**	**0.0364**	**0.0155**	**0.0197**

Generalization to other residual-quantization tokenizers. To examine whether scale-aware decomposition is specific to RQ-VAE, we also apply the same projection-based update to RQ-KMeans, resulting in SRQ-KMeans. As shown in Table 4, SRQ-KMeans consistently improves over RQ-KMeans across all datasets and metrics. On Beauty, NDCG@5 increases from 0.0275 to 0.0293 and NDCG@10 from 0.0346 to 0.0364; on Toys, the improvements are more pronounced, with NDCG@5 increasing from 0.0241 to 0.0265 and NDCG@10 from 0.0302 to 0.0327. Similar gains are observed on Sports, where Recall@10 improves from 0.0346 to 0.0364 and NDCG@10 from 0.0186 to 0.0197. Overall, SRQ-KMeans improves all 12 reported metrics over its RQ-KMeans

Table 5: Performance comparison under the LLM-based SFT-then-RL setting. The top-1/2/3 results are highlighted in red, yellow, and green, respectively.

Dataset	Methods	Recall@3	NDCG@3	Recall@5	NDCG@5	Recall@10	NDCG@10
Industrial	**Traditional**						
	GRU4Rec	0.0638	0.0542	0.0774	0.0598	0.0999	0.0669
	Caser	0.0618	0.0514	0.0717	0.0555	0.0942	0.0628
	SASRec	0.0790	0.0700	0.0909	0.0748	0.1088	0.0806
	Generative						
	HSTU	0.0927	0.0885	0.1037	0.0918	0.1163	0.0958
	TIGER	0.0852	0.0742	0.1010	0.0807	0.1321	0.0908
	LCRec	0.0915	0.0805	0.1057	0.0862	0.1332	0.0952
	LLM-based						
	BIGRec	0.0931	0.0841	0.1092	0.0907	0.1370	0.0997
	D^3	0.1024	**0.0991**	0.1213	0.0989	0.1500	0.1082
	S-DPO	0.1032	0.0906	0.1238	0.0991	0.1524	0.1082
	Ours						
	MiniOneRec	**0.1125**	0.0988	0.1255	**0.1042**	0.1508	0.1123
	SIGER (SFT)	0.0933	0.0844	0.1092	0.0909	0.1432	0.1017
	SIGER (RL+SFT)	0.1099	0.0974	**0.1262**	0.1041	**0.1538**	**0.1131**
Office	**Traditional**						
	GRU4Rec	0.0629	0.0528	0.0789	0.0595	0.1019	0.0669
	Caser	0.0748	0.0615	0.0865	0.0664	0.1093	0.0737
	SASRec	0.0861	0.0769	0.0949	0.0805	0.1120	0.0858
	Generative						
	HSTU	0.1134	0.1031	0.1252	0.1079	0.1400	0.1126
	TIGER	0.0986	0.0852	0.1163	0.0960	0.1408	0.1002
	LCRec	0.0921	0.0807	0.1048	0.0859	0.1237	0.0920
	LLM-based						
	BIGRec	0.1069	0.0961	0.1204	0.1017	0.1434	0.1091
	D^3	0.1204	0.1055	0.1406	0.1139	0.1634	0.1213
	S-DPO	0.1169	0.1033	0.1356	0.1110	0.1587	0.1255
	Ours						
	MiniOneRec	0.1217	0.1088	0.1420	0.1172	0.1634	0.1242
	SIGER (SFT)	0.1171	0.1033	0.1348	0.1106	0.1630	0.1196
	SIGER (RL+SFT)	**0.1289**	**0.1134**	**0.1439**	**0.1195**	**0.1675**	**0.1272**

counterpart, indicating that the benefit of scale-aware residual decomposition is not specific to RQ-VAE and can generalize to other residual-quantization tokenizers.