LESSRec: Level-Specific History Modeling for Semantic ID-Based Recommendation

Shangxin Yang
Chongqing University
Chongqing, China
yangshangxin@stu.cqu.edu.cn

Zongwei Wang
Chongqing University
Chongqing, China
zongwei@cqu.edu.cn

Jing Long
Chongqing University
Chongqing, China
jing.long@cqu.edu.cn

Junliang Yu
Griffith University
Brisbane, Australia
junl.yu@outlook.com

Min Gao*
Chongqing University
Chongqing, China
gaomin@cqu.edu.cn

arXiv:2610.37286v1 [cs.IR] 27 Oct 2026

Abstract

Semantic ID-based recommendation represents each item as a sequence of semantic tokens obtained through residual quantization. This paradigm generates the next item's tokens autoregressively from coarse to fine, entailing each prediction attending to historical tokens at all semantic levels. However, our inference-time analysis reveals that historical tokens at the target-token's semantic level tend to be more informative, whereas other levels can introduce interference. Motivated by these findings, we propose LESSRec that separately encodes the historical token sequence at each semantic level and restricts attention over historical tokens to the sequence at the target token's semantic level during prediction. To complement these level-specific histories, LESSRec explicitly provides an item-level semantic sequence that represents each historical item's overall semantics in chronological order. The decoder accesses the level-aligned token sequence and the item-level semantic sequence through two separate cross-attention branches and fuses their contexts. We further introduce LESSRec+, which uses separate encoders for different semantic-level histories with the same total parameter count to reduce encoder computation. Extensive experiments across three public datasets and three tokenizers demonstrate the recommendation effectiveness of LESSRec and LESSRec+, with efficiency analyses highlighting the computational benefits of LESSRec+.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Generative Recommendation, Semantic ID, Tokenization

1 Introduction

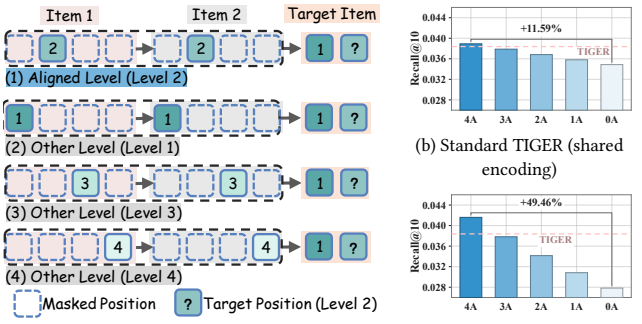
Recommender systems support personalized item discovery on Web platforms such as online retail services [25]. Generative recommendation has recently gained attention as a new paradigm that formulates item prediction as a sequence generation task [2, 3, 35, 41]. Under this paradigm, semantic ID-based approaches have become particularly prominent, which predict items by generating their semantic IDs, each comprising an ordered sequence of discrete semantic tokens obtained through residual quantization [24, 32, 42]. Within each item's semantic IDs, earlier tokens typically represent coarse-grained item semantics, while later tokens encode finer-grained information [16, 33, 39]. This structure enables semantically similar items to share coarse-level prefix tokens, capturing their common semantics while retaining finer-grained distinctions.

Conventional semantic ID-based recommendation models adopt an encoder–decoder architecture [16, 24, 41]. The encoder contextualizes historical semantic tokens through self-attention, allowing tokens from different items and semantic levels to exchange information. The decoder can then attend to encoded tokens from all historical items and semantic levels, regardless of the level being predicted. This design brings together two distinct sequential structures: tokens at the same level across interactions capture behavioral patterns at a consistent semantic granularity, whereas tokens within an item describe its progressive semantic refinement. Given these different roles, it is unclear whether information from all levels is equally useful for next token prediction. This motivates a key question: when predicting a token at a particular semantic level, which levels in the interaction history provide useful evidence?

To understand how the decoder uses historical information from different semantic levels for prediction, we conduct an inference-time masking experiment on TIGER [24], a representative encoder–decoder generative recommender. We keep all parameters fixed and modify only the decoder's cross-attention mask, restricting each target position to tokens at a single semantic level in the interaction history (Figure 1(a)). Across masking configurations, the number of visible tokens remains constant, but performance improves as more target positions attend to their matching semantic levels (Figure 1(b)). With all positions aligned, each prediction accesses only one quarter of the history tokens, yet the recommendation performance slightly outperforms that of original TIGER, which uses unrestricted history access during both training and inference. These results suggest that explicitly matching historical access to the target semantic level can be more effective than providing access to all levels.

However, TIGER's encoder mixes information across semantic levels before decoding, so attending to positions at one level does not isolate information from that level. To examine whether this mixing affects the observed alignment benefit, we train a TIGER variant that restricts encoder self-attention to tokens at the same semantic level, leaving the rest of the architecture unchanged. We then repeat the inference-time masking experiment on this variant.

*Corresponding author.



(a) History-access patterns for a level-2 target

(b) Standard TIGER (shared encoding)

(c) TIGER with level-isolated encoding

Figure 1: Same-level vs. cross-level history access for prediction in TIGER under standard and level-isolated encoding (Recall@10 on Games). Under inference-time cross-attention masking, (m)A denotes (m) of four target positions using same-level access (e.g., (1) in (a)); the remaining positions use level-mismatched access (e.g., (2), (3), and (4) in (a)). Bars average configurations with the same (m). Dashed lines mark the original TIGER baseline; percentages show relative gains of 4A over 0A.

Figure 1(c) shows a larger contrast between aligned and mismatched access. Within each model, fully aligned access improves Recall@10 over fully mismatched access by 11.59% under standard encoding and 49.46% under level-isolated encoding. With cross-level mixing restricted during encoding, the larger gap between aligned and mismatched access provides clearer evidence that same-level history is more useful for prediction than cross-level history.

Motivated by these findings, we propose LESSRec (LEvel-Specific history modeling for Semantic ID-based Recommendation) to better use level-specific historical evidence. Specifically, Level-Isolated Self-Attention separately contextualizes historical tokens at each semantic level, preserving same-level associations across items. During prediction, the decoder uses Level-Aligned Cross-Attention to attend to historical semantic tokens at the same level as the target token. However, a single level offers only a partial semantic view, and relying on it alone may overlook how different levels complement one another in describing an item. To preserve this complementarity, we combine the contextualized states from different levels within each historical item to construct an Item-Level Semantic Sequence. The decoder accesses this sequence through a separate cross-attention branch and fuses its context with the level-aligned context, combining level-specific historical evidence with complementary item semantics for prediction.

LESSRec implements this organization through attention masks while sharing encoder layers across semantic levels. To avoid unnecessary computation over masked cross-level interactions and give each level its own encoder-layer parameters, we further propose LESSRec+. LESSRec+ partitions the existing encoder layers into level-specific groups, each processing only the historical semantic tokens at its assigned level. This reduces encoder computation and intermediate activation storage while retaining the same total parameter count as LESSRec.

Our main contributions are as follows:

- To the best of our knowledge, we are the first to identify the predictive advantage of same-level history in semantic ID-based generative recommendation.
- We propose LESSRec to jointly organize same-level historical associations across items and complementary semantic information within each item for prediction at each semantic level. LESSRec+ further reorganizes the encoder into level-specific groups, reducing computation while retaining the same parameter count as LESSRec.
- Experiments on three datasets with three tokenizers demonstrate the recommendation effectiveness of LESSRec and LESSRec+. Component ablations support the proposed design, while efficiency analyses show the computational benefits of LESSRec+.

2 Preliminaries

Let $\mathcal{I} = (v_1, \ldots, v_T)$ be a user's interaction history in chronological order. Each item v_i is represented by its semantic IDs, obtained through the residual quantization process described in Appendix B:

$$\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,L}),$$

where $s_{i,r}$ is the semantic token at level $r \in \{1, \ldots, L\}$. The item-to-semantic-ID mapping remains fixed after tokenization. We concatenate the semantic IDs of historical items in interaction order to form the input:

$$\mathbf{x} = (s_{1,1}, \ldots, s_{1,L}, \ldots, s_{T,1}, \ldots, s_{T,L}, \text{EOS}). \tag{1}$$

The decoder autoregressively generates the next item's semantic IDs, $Y_1, \ldots, Y_L$, followed by EOS. At target position k, the prediction is conditioned on the history $\mathbf{x}$ and the preceding target tokens $C_k = Y_{<k}$.

For each semantic level r, we define the same-level historical sequence as

$$H_r = (s_{1,r}, s_{2,r}, \ldots, s_{T,r}),$$

which contains the level-r tokens of historical items in interaction order. To specify which tokens are visible under semantic-level masking, let $p(i, r)$ denote the zero-based position of $s_{i,r}$ in the input sequence $\mathbf{x}$:

$$p(i, r) = L(i - 1) + (r - 1).$$

The set $\mathcal{P}_r = \{p(i, r) : 1 \le i \le T\}$ collects the input positions of tokens in H_r. Including the end token EOS at $p_{\text{end}} = LT$, we define $\mathcal{P}_{\text{valid}} = \left(\bigcup_{r=1}^{L} \mathcal{P}_r\right) \cup \{p_{\text{end}}\}$ as the set of all valid input positions, excluding padding.

3 Method

To address cross-level interference, we propose LESSRec (Figure 2). It comprises level-isolated self-attention (Section 3.1), level-aligned cross-attention (Section 3.2), and an item-level semantic sequence (Section 3.3). After describing these components and their integration, we introduce LESSRec+ with level-specific encoders (Section 3.6).

3.1 Level-Isolated Self-Attention (LIS)

To help the model learn same-level contextual information without interference from other levels, we introduce Level-Isolated

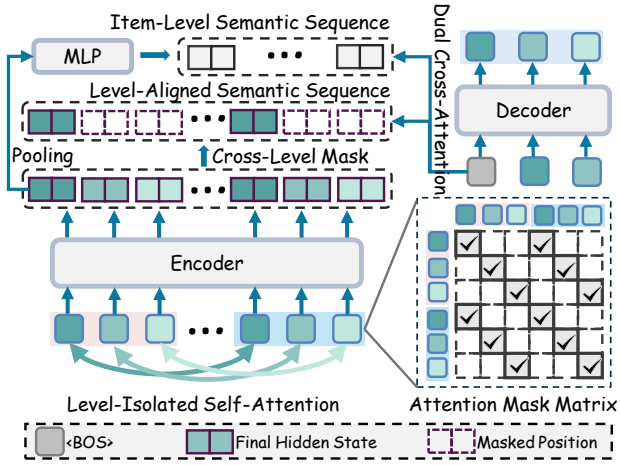


Figure 2: Overview of LESSRec. For visual clarity, each item is represented by three semantic tokens, and <EOS> tokens are omitted. Shades of the same color distinguish semantic levels, from coarse (dark) to fine (light). In the attention mask matrix, ✓ indicates visible positions and blank cells indicate masked positions.

Self-Attention (LIS). LIS restricts each historical semantic token to attending only to tokens at the same level across items:

$$M^{\text{enc}}_{qj} = \begin{cases} 0, & q \in \mathcal{P}_r,\ j \in \mathcal{P}_r, \\ 0, & q = p_{\text{end}},\ j \in \mathcal{P}_{\text{valid}}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{2}$$

Here, q and j are the zero-based positions of the query token and the token being attended to in $\mathbf{x}$, respectively. This mask is applied in every encoder layer so that historical semantic tokens exchange information only within their corresponding level sequences. Because EOS serves as a shared terminator for the complete input sequence, we allow its encoder state to attend to all valid positions, excluding padding. However, historical semantic tokens cannot attend to EOS: $M^{\text{enc}}_{q,p_{\text{end}}} = -\infty$ for all $q \in \bigcup_{r=1}^{L} \mathcal{P}_r$. Thus, information aggregated by EOS cannot flow back into these token states, and the historical sequences remain semantically isolated across levels during encoding.

Let $\mathbf{h}_{i,r} \in \mathbb{R}^d$ denote the final encoded state of the level-r token of item i. Each state incorporates historical context from H_r without mixing information from tokens at other levels.

3.2 Level-Aligned Cross-Attention (LAC)

Although LIS separates historical semantic levels during encoding, unrestricted cross-attention still allows the decoder to combine information from different levels. To focus each prediction on same-level history, we introduce Level-Aligned Cross-Attention (LAC): when predicting a level-r target token, the level-aligned branch attends only to historical level-r tokens.

The cross-attention mask for the level-aligned branch is

$$M^{\text{cross}}_{tj} = \begin{cases} 0, & 0 \le t < L,\ j \in \mathcal{P}_{t+1} \cup \{p_{\text{end}}\}, \\ 0, & t = L,\ j \in \mathcal{P}_{\text{valid}}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{3}$$

Here, $t \in \{0, \ldots, L\}$ denotes the zero-based decoder query position, and j denotes the zero-based position of an encoded history token. Positions $t = 0, \ldots, L-1$ predict the L semantic tokens, and $t = L$ predicts the end token.

LIS and LAC act at complementary stages: LIS prevents cross-level mixing in historical semantic-token states, and LAC selects the matching level's states during prediction. Together, they enable the decoder to use contextual information from same-level history without attending to historical semantic tokens at other levels. Appendix A provides a theoretical analysis of same-level history access under simplified assumptions.

3.3 Item-Level Semantic Sequence (ILS)

LIS and LAC help the decoder use same-level history, but attending to a single semantic level limits access to the combined semantics of each historical item. We therefore construct an Item-Level Semantic Sequence (ILS), in which each representation combines an item's encoded semantic tokens and the representations follow the user's item interaction order. For each historical item v_i, we construct an item-level representation by averaging the LIS-encoded representations of its L semantic tokens:

$$\mathbf{z}_i = \frac{1}{L} \sum_{r=1}^{L} \mathbf{h}_{i,r}. \tag{4}$$

Since each $\mathbf{h}_{i,r}$ already incorporates same-level historical context through LIS, the pooled representation $\mathbf{z}_i$ combines the item's semantics with this context. We therefore construct the item-level sequence from these contextualized representations without an additional self-attention step.

To refine each item representation while retaining the pooled information, we apply a residual projection:

$$\mathbf{m}_i = \mathbf{z}_i + \text{MLP}(\text{LN}(\mathbf{z}_i)), \tag{5}$$

$$\text{MLP}(\text{LN}(\mathbf{z}_i)) = W_2\, \text{GELU}(W_1\, \text{LN}(\mathbf{z}_i) + \mathbf{b}_1) + \mathbf{b}_2. \tag{6}$$

Here, LN denotes layer normalization, and MLP is a two-layer feed-forward network with GELU (Gaussian error linear unit) activation. The residual connection adds the learned nonlinear update to the pooled representation $\mathbf{z}_i$. The resulting sequence $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_T)$ represents historical items in interaction order and provides item-level context alongside the level-aligned history.

3.4 Dual Cross-Attention Fusion

To combine information from the level-aligned semantic sequence and the item-level semantic sequence constructed above, we design dual cross-attention fusion. Let $\mathbf{R}$ denote the encoded historical token sequence produced by LIS and $\mathbf{M}$ the item-level semantic sequence. For the same decoder query $\mathbf{Q}$, two multi-head cross-attention branches [31] separately attend to these sequences:

$$\mathbf{o}_{\text{level}} = \text{MHA}(\mathbf{Q}, \mathbf{R}, \mathbf{R}; B_{\text{level}} + M^{\text{cross}}), \tag{7}$$

$$\mathbf{o}_{\text{item}} = \text{MHA}(\mathbf{Q}, \mathbf{M}, \mathbf{M}; B_{\text{item}} + M^{\text{item}}). \tag{8}$$

Here, M^{cross} is the level-aligned cross-attention mask defined in Equation 3, and M^{item} masks padded positions in the item-level sequence. B_{level} is the original relative position bias that distinguishes semantic-token positions in the level-specific history.

This token-based bias does not match the item-level semantic sequence: consecutive historical items occupy adjacent positions in $\mathbf{M}$, whereas their corresponding same-level tokens are L positions apart in the input semantic-token sequence. We therefore introduce B_{item} to model positions in the item-level semantic sequence:

$$B^{(h)}_{\text{item}}(t, j) = E^{(h)}_{\text{item}}\left[\text{bucket}(j - t)\right]. \tag{9}$$

Here, t is the decoder query position, $j \in \{0, \ldots, T-1\}$ indexes historical items in interaction order, and h indexes attention heads. The function bucket$(\cdot)$ groups relative offsets into distance ranges and returns the corresponding index, which is used to retrieve a bias from the learned table E_{item} [23]. Defining these offsets over item positions provides the decoder with positional information aligned with the chronological order of historical item interactions.

We fuse the contexts from the level-aligned history and the item-level semantic sequence through a weighted sum of the two branch outputs:

$$\mathbf{o} = (1 - \alpha)\mathbf{o}_{\text{level}} + \alpha\mathbf{o}_{\text{item}}. \tag{10}$$

Here, $\alpha \in [0, 1]$ is a hyperparameter controlling the relative contributions of the level-aligned history and the item-level semantic sequence. This fusion retains information from the historical sequence at the target semantic level while incorporating the overall semantics of items in interaction order.

3.5 Training and Inference

We jointly optimize the encoder, decoder, residual item projection, and item-level positional bias using the average autoregressive negative log-likelihood:

$$\mathcal{L} = -\frac{1}{(L+1)|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \sum_{k=1}^{L+1} \log p_\theta(y_k \mid y_{<k}, \mathbf{x}), \tag{11}$$

where $\mathcal{D}$ contains history–target pairs and $\mathbf{y}$ consists of the next item's L semantic tokens followed by the end token. During training, each prediction is conditioned on the preceding tokens in the ground-truth target sequence. No auxiliary objective is required, and the tokenizer and item-to-semantic-ID mapping remain fixed.

At inference, the decoder autoregressively generates the next item's semantic IDs using trie-constrained beam search over valid identifiers [1].

3.6 LESSRec+: Level-Specific Encoders

Level-Isolated Self-Attention provides a simple implementation of semantic-level isolation by masking encoder self-attention. However, all semantic levels still share the same encoder layers, and attention scores are computed for all pairs of historical semantic tokens before masking. LESSRec+ instead divides the original encoder layers into level-specific groups, each encoding only the historical token sequence at its assigned semantic level (Figure 3).

For each semantic level r, we form the level-specific input $\mathbf{x}^{(r)} = (s_{1,r}, \ldots, s_{T,r}, \text{EOS})$. The original encoder has D layers. We divide them into L disjoint groups, assigning D_r consecutive layers to semantic level r with $\sum_{r=1}^{L} D_r = D$. Each group forms an encoder

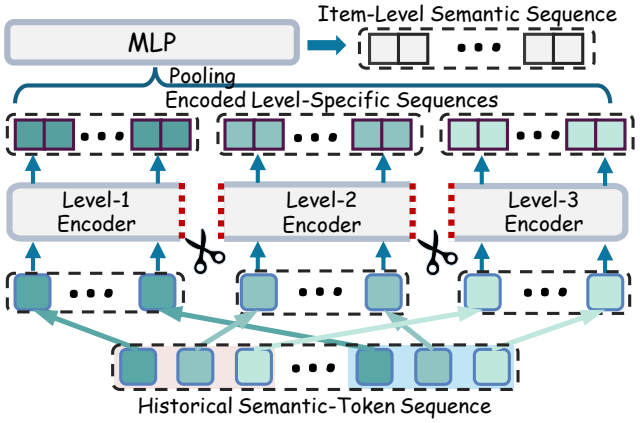


Figure 3: Encoder-side illustration of LESSRec+. Its decoder uses the same dual cross-attention as in Figure 2.

E_r that processes only $\mathbf{x}^{(r)}$. By reusing the existing embeddings and partitioning the encoder layers, each encoder independently models its level's history without increasing the total parameter count.

After encoding, we restore each historical token state to its position $p(i, r)$ in the original layout. The level-specific end-token states are averaged to obtain a single encoder EOS state:

$$\widetilde{\mathbf{h}}_{\text{end}} = \frac{1}{L}\sum_{r=1}^{L} \mathbf{h}^{(r)}_{\text{end}}. \tag{12}$$

The restored states are used to construct Item-Level Semantic Sequence and supply the level-aligned token history. All other modules and the training objective remain as defined above.

Each encoder now processes $T + 1$ positions. Summed over all encoder layers, the attention-score matrix size is $D(T+1)^2$, compared with $D(LT+1)^2$ for masked attention in LESSRec. Moreover, the level-specific encoders have no computational dependencies on one another and can therefore run in parallel, reducing the longest sequential path through encoder layers from D to $\max_r D_r$. Processing shorter input sequences reduces encoder computation and intermediate activation storage without adding parameters; we compare the models in terms of computational complexity, GPU memory usage, and efficiency, and analyze encoder layer allocation in the experimental section.

4 Experiments

We organize the experiments around five research questions:

- **RQ1:** How do LESSRec and LESSRec+ perform against recommendation baselines across datasets and tokenization methods?
- **RQ2:** How do the three main components (LIS, LAC, and ILS) affect recommendation performance?
- **RQ3:** How do different settings (α, the number of semantic tokens per item, and encoder layer allocation) affect recommendation performance?
- **RQ4:** How do the models compare in computational complexity, parameter count, peak GPU memory usage, and runtime?

- **RQ5:** How does interaction-history length affect the recommendation performance and encoding efficiency of LESSRec and LESSRec+?

4.1 Experimental Settings

4.1.1 Datasets. We evaluate LESSRec on Games (Video Games), Software, and Beauty. Games and Software use Amazon Reviews 2023 [7]; its source collection covers May 1996 to September 2023. Beauty comes from an earlier Amazon review collection [19]. Following previous studies [24, 32], we iteratively remove users and items with fewer than five interactions until the remaining data satisfy the 5-core requirement. We restrict the number of items in a user's history to 20. Table 1 reports the statistics after preprocessing.

Table 1: Dataset statistics after final preprocessing. Avg. length is the mean number of interactions per user.

Dataset	Users	Items	Interactions	Sparsity	Avg. length
Games	26,443	10,512	202,250	99.9272%	7.65
Software	22,225	4,654	200,915	99.8058%	9.04
Beauty	22,363	12,101	198,502	99.9266%	8.88

4.1.2 Baselines. We compare LESSRec with conventional sequential recommenders and semantic ID-based generative recommenders: **Caser** [29] models interaction sequences with convolutional filters. **GRU4Rec** [4] uses recurrent neural networks to model sequential interactions. **BERT4Rec** [26] learns bidirectional interaction representations through masked-item prediction. **SASRec** [12] models sequential interactions using self-attention. **RPG** [8] generates the tokens of long, unordered semantic IDs in parallel. **CARE** [14] combines progressive history encoding with query-anchored reasoning to mitigate bias amplification. **DiscRec** [15] is a structure-aware recommender that combines intra-item attention with gated embedding fusion. **TIGER** [24] uses RQ-VAE to quantize item representations into semantic IDs and autoregressively generates the next item's semantic IDs from the interaction history. We instantiate and compare TIGER, LESSRec, and LESSRec+ using three tokenizers. **RQ-VAE** [13] learns hierarchical codebooks by recursively quantizing residuals of item semantic representations. The **RQ-Kmeans** setting uses K-means-guided conditional residual quantization with trainable subcodebooks. Appendix B details both implementations. **LETTER** [32] aligns quantized item representations with collaborative item embeddings and applies diversity regularization to alleviate code-assignment bias.

4.1.3 Evaluation Settings. We adopt a leave-one-out evaluation protocol. Recommendation quality is measured using Recall@5, Recall@10, NDCG@5, and NDCG@10. Within each tokenization setting, comparisons among TIGER, LESSRec, and LESSRec+ use the same data partitions, fixed semantic IDs, history length, candidate index, decoding settings, and checkpoint-selection rule.

4.1.4 Generation of semantic IDs. For each dataset, semantic IDs are generated offline and remain fixed during recommendation training and evaluation. For RQ-VAE, RQ-Kmeans, and LETTER, we use four semantic tokens per item ($L = 4$) and 256 codewords per codebook. Item semantic embeddings are extracted from text using Sentence-T5 [18]. For LETTER, we use SASRec [12] to obtain collaborative item embeddings. For the RPG baseline, we use 32-token semantic IDs.

4.1.5 Implementation Details. TIGER, LESSRec, and LESSRec+ are implemented with Hugging Face Transformers using a randomly initialized T5-small backbone [23], with six encoder and six decoder layers and a hidden dimension of 512. LESSRec+ allocates the six encoder layers across semantic levels with depths [2,2,1,1]. The item-memory MLP has an intermediate dimension of 256, and $\alpha = 0.5$. We use AdamW with a learning rate of 5×10^{-4}, weight decay of 0.01, and a cosine schedule with a 1% warmup ratio. The training batch size is 256 with two gradient accumulation steps. Each run uses a single NVIDIA GeForce RTX 4090 GPU (24 GB) for at most 200 epochs, stopping after 20 epochs without validation-loss improvement and selecting the checkpoint with the lowest validation loss. Inference uses a batch size of 32 and trie-constrained beam search with a beam size of 10. For the generative recommendation baselines, we utilize their official implementations. RPG uses a randomly initialized Transformer decoder with two layers, a hidden dimension of 448, and four attention heads. CARE uses the pretrained Qwen2.5-0.5B backbone [22], retaining its 24 Transformer layers, hidden dimension of 896, and 14 attention heads. DiscRec is also implemented with a T5-small backbone [23]. For the conventional sequential recommendation baselines, we implement Caser, GRU4Rec, BERT4Rec, and SASRec using RecBole [37, 38].

4.2 Overall Performance (RQ1)

Table 2 compares LESSRec and LESSRec+ with conventional sequential and generative recommenders on Games, Software, and Beauty. We summarize the main observations as follows.

- **Comparison with existing recommenders.** Generative methods based on semantic IDs show an advantage over conventional sequential recommenders. Their semantic-token representations allow related items to share information while expressing item semantics at multiple granularities, which can benefit next-item generation. Building on these representations, the best LESSRec or LESSRec+ configuration achieves the highest score in every dataset–metric column. This supports explicitly modeling the different predictive roles of historical semantic levels to better exploit semantic IDs.
- **Consistent gains across tokenizers.** Both models outperform the corresponding baseline under all three tokenizers across all datasets and metrics. Relative gains range from 6.2% to 39.2% for LESSRec and from 6.0% to 54.3% for LESSRec+. The improvements are not confined to a particular quantizer. These results suggest that models that encode historical sequences separately by semantic level and use same-level history for prediction are better suited to the hierarchical structure of semantic IDs obtained through residual quantization.
- **Benefits of level-specific encoders.** LESSRec implements level isolation through attention masking, a simple approach in which all semantic levels still share the encoder layers. LESSRec+ instead assigns separate encoder-layer parameters to each semantic-level history while keeping the total parameter count unchanged.

Table 2: Main recommendation results on three public datasets. Score cells report mean performance over seeds {42, 43, 44}. Shaded "Improv." rows report percentage gains over the baseline within the same tokenization group. Bold values indicate statistically significant improvements over the corresponding baseline under a paired t-test ($p < 0.05$).

	Games				Software				Beauty			
Method	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10
Caser	0.0278	0.0456	0.0172	0.0229	0.1296	0.1920	0.0829	0.1029	0.0172	0.0299	0.0105	0.0146
GRU4Rec	0.0278	0.0441	0.0171	0.0224	0.1241	0.1815	0.0765	0.0951	0.0138	0.0234	0.0086	0.0117
BERT4Rec	0.0263	0.0415	0.0166	0.0215	0.1302	0.1898	0.0847	0.1040	0.0123	0.0226	0.0076	0.0110
SASRec	0.0226	0.0421	0.0116	0.0179	0.1093	0.1818	0.0606	0.0839	0.0178	0.0358	0.0095	0.0152
RPG	0.0264	0.0383	0.0181	0.0219	0.0832	0.1080	0.0620	0.0700	0.0261	0.0390	0.0178	0.0219
CARE	0.0228	0.0461	0.0136	0.0211	0.1308	0.2266	0.0828	0.1135	0.0214	0.0465	0.0130	0.0211
DiscRec	0.0283	0.0395	0.0184	0.0219	0.1379	0.1959	0.0925	0.1113	0.0223	0.0365	0.0137	0.0183
TIGER (RQ-VAE)	0.0260	0.0371	0.0172	0.0208	0.1276	0.1868	0.0842	0.1034	0.0201	0.0331	0.0125	0.0167
LESSRec (RQ-VAE)	0.0319	0.0457	0.0213	0.0257	0.1479	0.2092	0.1000	0.1198	0.0276	0.0440	0.0174	0.0227
Improv.	**+22.7%**	**+23.2%**	**+23.8%**	**+23.6%**	**+15.9%**	**+12.0%**	**+18.8%**	**+15.9%**	**+37.3%**	**+32.9%**	**+39.2%**	**+35.9%**
LESSRec+ (RQ-VAE)	0.0335	0.0475	0.0225	0.0270	0.1559	0.2163	0.1067	0.1263	0.0297	0.0464	0.0187	0.0241
Improv.	**+28.8%**	**+28.0%**	**+30.8%**	**+29.8%**	**+22.2%**	**+15.8%**	**+26.7%**	**+22.1%**	**+47.8%**	**+40.2%**	**+49.6%**	**+44.3%**
TIGER (RQ-Kmeans)	0.0316	0.0465	0.0212	0.0260	0.1297	0.1880	0.0838	0.1027	0.0228	0.0369	0.0145	0.0190
LESSRec (RQ-Kmeans)	0.0385	0.0568	0.0255	0.0314	0.1477	0.2076	0.0995	0.1189	0.0315	0.0489	0.0200	0.0256
Improv.	**+21.8%**	**+22.2%**	**+20.3%**	**+20.8%**	**+13.9%**	**+10.4%**	**+18.7%**	**+15.8%**	**+38.2%**	**+32.5%**	**+37.9%**	**+34.7%**
LESSRec+ (RQ-Kmeans)	0.0387	0.0560	0.0259	0.0316	0.1555	0.2148	0.1061	0.1253	0.0343	0.0537	0.0218	0.0281
Improv.	**+22.5%**	**+20.4%**	**+22.2%**	**+21.5%**	**+19.9%**	**+14.3%**	**+26.6%**	**+22.0%**	**+50.4%**	**+45.5%**	**+50.3%**	**+47.9%**
TIGER (LETTER)	0.0334	0.0513	0.0218	0.0276	0.1383	0.2027	0.0930	0.1138	0.0225	0.0375	0.0138	0.0187
LESSRec (LETTER)	0.0379	0.0584	0.0249	0.0315	0.1506	0.2177	0.0993	0.1209	0.0282	0.0471	0.0177	0.0238
Improv.	**+13.5%**	**+13.8%**	**+14.2%**	**+14.1%**	**+8.9%**	**+7.4%**	**+6.8%**	**+6.2%**	**+25.3%**	**+25.6%**	**+28.3%**	**+27.3%**
LESSRec+ (LETTER)	0.0358	0.0577	0.0231	0.0301	0.1616	0.2270	0.1109	0.1320	0.0324	0.0511	0.0213	0.0273
Improv.	**+7.2%**	**+12.5%**	**+6.0%**	**+9.1%**	**+16.8%**	**+12.0%**	**+19.2%**	**+16.0%**	**+44.0%**	**+36.3%**	**+54.3%**	**+46.0%**

It improves over LESSRec in 31 of the 36 dataset–tokenizer–metric comparisons. This overall trend suggests that fully separating encoder-layer parameters helps each encoder capture its assigned level's semantics while limiting cross-level interference. This separation appears more effective than masking attention interactions alone while sharing attention projections and feed-forward network parameters across levels. The computational benefits of LESSRec+ are examined in Section 4.5.

4.3 Ablation Study (RQ2)

Table 3 examines the three main components, LIS, LAC, and ILS, on Games and Beauty. The *+ILS* variant adds only the item-level semantic sequence to TIGER. The remaining variants progressively add LIS, LAC, and ILS to obtain LESSRec.

Adding ILS alone improves TIGER on both datasets, but its results remain below all variants that include LIS. Thus, supplying item-level context alone does not account for the full model's gains. Adding LIS improves performance even with unrestricted decoder access, suggesting that separating semantic levels during encoding helps retain useful same-level contextual information. Adding LAC yields further gains, indicating that explicitly selecting same-level history benefits prediction beyond encoder-side isolation alone.

Table 3: Component ablations on Games and Beauty.

Variant	Games		Beauty	
	R@10	N@10	R@10	N@10
TIGER (baseline)	0.0371	0.0208	0.0331	0.0167
+ ILS	0.0408	0.0231	0.0344	0.0169
+ LIS	0.0418	0.0234	0.0374	0.0191
+ LIS + LAC	0.0422	0.0239	0.0395	0.0200
+ LIS + LAC + ILS (**LESSRec**)	**0.0457**	**0.0257**	**0.0440**	**0.0227**

Finally, adding ILS to LIS+LAC achieves the best performance among the compared variants on both datasets. Together with the ILS-only results, this supports its complementary role: the interaction-ordered item-level semantic sequence supplies overall item semantics alongside the same-level historical information learned and accessed through LIS and LAC. Additional design-choice ablations are presented in Appendix C.

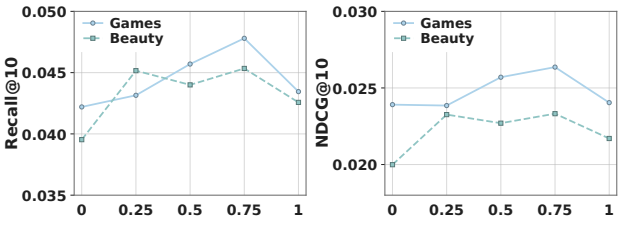


Figure 4: Sensitivity to the cross-attention fusion coefficient α on Games and Beauty.

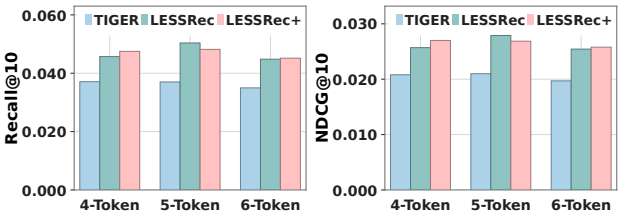


Figure 5: Effect of semantic-token count on Games.

4.4 Effect of Different Settings (RQ3)

4.4.1 Fusion Coefficient α. The decoder combines the level-aligned token history and the item-level semantic sequence using the coefficient α in Eq. 10. We vary α from 0 (only the level-aligned token history) to 1 (only the item-level semantic sequence), keeping the other settings fixed. Figure 4 shows the results on Games and Beauty. Both datasets achieve their best results with intermediate coefficients, supporting the complementary roles of same-level historical information and overall item semantics.

4.4.2 Number of Semantic Tokens per Item. Figure 5 shows the results on Games with four, five, or six semantic tokens per item. For LESSRec+, the encoder depth allocations from coarse to fine semantic levels are $[2, 1, 1, 1, 1]$ for five tokens and $[1, 1, 1, 1, 1, 1]$ for six tokens. Both LESSRec and LESSRec+ consistently outperform TIGER across all three settings. These results suggest that modeling historical semantic levels benefits hierarchical semantic IDs with different token counts, rather than depending on a particular count.

4.4.3 Encoder Depth Allocation. Table 4 shows the results of five encoder depth allocations on Games, each retaining six layers in total. No allocation performs best across all metrics: $[3, 1, 1, 1]$ achieves the highest R@5 and N@5, while $[1, 1, 2, 2]$ achieves the highest R@10; the two share the highest N@10 at the reported precision. The default allocation, $[2, 2, 1, 1]$, remains competitive but is not uniformly optimal. These results do not indicate a consistent advantage from assigning more layers exclusively to earlier or later semantic levels. In contrast, $[1, 2, 2, 1]$ performs worst across all metrics, suggesting that allocating additional depth to the middle levels is less effective in this setting. Nevertheless, all five allocations outperform TIGER, indicating that LESSRec+'s gains persist across the tested depth distributions. Choosing how to distribute encoder capacity across semantic levels remains a useful direction for further optimization.

Table 4: LESSRec+ encoder layer allocations on Games.

Allocation	R@5	R@10	N@5	N@10
[2,2,1,1]	0.0335	0.0475	0.0225	0.0270
[3,1,1,1]	0.0340	0.0469	0.0230	0.0271
[1,2,2,1]	0.0312	0.0459	0.0210	0.0257
[1,1,2,2]	0.0332	0.0481	0.0222	0.0271
[1,1,1,3]	0.0330	0.0478	0.0217	0.0266

Table 5: Complexity of attention-matrix operations.

Model	Encoder self-attention	Decoder cross-attention
TIGER	$O(D_e L^2 T^2 d)$	$O(D_d QLTd)$
LESSRec	$O(D_e L^2 T^2 d)$	$O(D_d QLTd) + O(D_d QTd)$
LESSRec+	$O(D_e T^2 d)$	$O(D_d QLTd) + O(D_d QTd)$

4.5 Complexity and Resource Efficiency (RQ4)

Table 5 compares the computational complexity of attention operations. The expressions exclude linear projections, feed-forward layers, and repeated beam-search steps. Let D_e and D_d denote the total encoder and decoder depths, Q the number of decoder query positions, L the number of semantic levels, T the number of historical items, and d the hidden dimension, with EOS and padding omitted from the sequence-length analysis.

TIGER and LESSRec compute self-attention over LT historical positions in each encoder layer, giving $O(D_e L^2 T^2 d)$. LESSRec+ processes sequences of length T, with the level-specific encoder depths summing to D_e, giving $O(D_e T^2 d)$. This yields a factor of L^2 reduction in these operations.

Decoder cross-attention over the encoded token history costs $O(D_d QLTd)$. Both LESSRec variants retain this branch and add an item-level branch costing $O(D_d QTd)$. The decoder cost is therefore the sum of these two terms, with the additional branch scaling linearly with history length. The main reduction in attention computation comes from the level-specific encoders in LESSRec+.

Table 6 compares model parameters, time to convergence, average training time per epoch, and peak training memory for TIGER, LESSRec, and LESSRec+. Both proposed models add only 0.4% more parameters, primarily from the item-representation MLP and a small number of item-level positional parameters. Time to convergence decreases by 34.0% for LESSRec and 65.2% for LESSRec+, consistent with the models learning useful predictive patterns more readily from separately encoded same-level histories.

LESSRec takes 5.8% longer per epoch because attention masking retains the original attention computation, while the item-level semantic sequence adds representation construction and a second cross-attention branch. In contrast, LESSRec+ reduces per-epoch time by 33.7% and peak training memory by 39.2%. Its level-specific encoders process shorter histories independently, avoiding cross-level attention computations that LESSRec still performs before masking.

Table 6: Model parameters and training costs. Colored arrows indicate percentage increases (red) or decreases (green) relative to TIGER.

Model	Params (M)	Training time to convergence (h)	Training time per epoch (s)	Peak Memory (GiB)
TIGER	60.86	2.27	99.53	8.79
LESSRec	61.12 ↑ 0.4%	1.49 ↓ 34.0%	105.33 ↑ 5.8%	9.46 ↑ 7.5%
LESSRec+	61.12 ↑ 0.4%	0.79 ↓ 65.2%	65.95 ↓ 33.7%	5.35 ↓ 39.2%

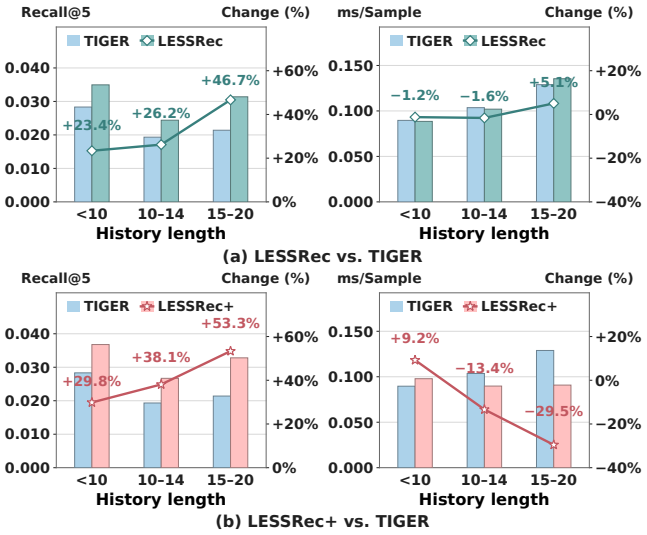


Figure 6: History-length analysis: (a) LESSRec versus TIGER; (b) LESSRec+ versus TIGER. Each row shows Recall@5 (left) and encoder CUDA time per sample (right).

4.6 Effect of History Length (RQ5)

We group test cases on Games into three history-length buckets: fewer than 10, 10–14, and 15–20 interactions. Figure 6 reports Recall@5 over seed 42 and encoder CUDA time per sample. Bars use the left axes for absolute values, while lines use the right axes for percentage changes relative to TIGER within each bucket.

Both proposed models outperform TIGER in all three buckets, with larger relative gains for longer histories. LESSRec's gains increase from 23.4% in the shortest-history bucket to 46.7% in the longest, while LESSRec+'s gains increase from 29.8% to 53.3%. These results suggest that same-level historical modeling remains effective across different history lengths and offers more pronounced benefits for users with longer observed histories. A possible explanation is that level isolation confines each level-aligned prediction to same-level history, only $1/L$ of all historical tokens, which limits dilution from irrelevant tokens as histories grow. The encoding-time results distinguish the two implementations. LESSRec remains close to TIGER, with changes ranging from a 1.6% reduction to a 5.1% increase. LESSRec+ incurs a 9.2% overhead in the shortest-history bucket but reduces encoding time by 13.4% and 29.5% in the middle and longest buckets, respectively. This pattern is consistent with the computational advantage of processing shorter, level-specific sequences becoming more consequential as history length grows.

5 Related Work

5.1 Semantic Item Tokenization

Semantic item tokenization determines the information encoded in discrete identifiers. VQ-Rec [6] discretizes item text embeddings using product quantization (PQ) [11], while TIGER [24] obtains hierarchical semantic IDs through RQ-VAE [13]. Related approaches explore learnable document identifiers [27], collaborative-aware item tokenization [21], tree-structured codebooks [40], textual identifiers [28], and action-context tokenization [9]. LESSRec keeps semantic IDs fixed and studies their use in historical sequence modeling. We evaluate TIGER, LESSRec, and LESSRec+ with RQ-VAE, RQ-Kmeans, and LETTER tokenization.

5.2 Semantic ID-Based Generative Recommendation

Semantic ID-based generative recommenders predict the next item's identifier from user history. TIGER adopts autoregressive generation of residual-quantized IDs [24]. ETEGRec couples tokenizer and recommender learning through sequence–item and preference–semantic alignment with alternating optimization [16]. EAGER-LLM integrates external behavioral and semantic information into a decoder-only language model through dual-source item indices and multiscale alignment tasks [5]. RPG generates long, unordered semantic IDs in parallel with graph-constrained decoding [8], while CARE combines progressive history encoding with query-anchored reasoning to mitigate bias amplification [14]. These methods improve joint learning, information integration, or generation strategies, complementing our focus on historical semantic-level modeling.

5.3 Structure-Aware Sequence Modeling

Structure-aware methods incorporate the hierarchy of semantic IDs and item boundaries into sequence modeling. TrieRec encodes prefix-tree topology through absolute and relative positional encodings [34]. DiscRec combines intra-item attention with embedding-level fusion of semantic and collaborative representations [15]. Hi-SAM compresses items into anchor tokens and separates inter-item from intra-item positional information [20]. BARGE restores item context during encoding and mitigates semantic drift through hierarchical path reranking and dual-path decoding [36]. LESSRec encodes same-level histories across items and aligns token-history access with the target semantic level, while retaining item-level context through a separate cross-attention branch. We compare LESSRec and LESSRec+ with DiscRec [15], a representative structure-aware baseline, in Section 4.

6 Conclusion

In this paper, we show that historical semantic levels play different roles in target-token prediction, with same-level history generally being more informative. LESSRec separately encodes historical token sequences at each semantic level and aligns decoder access with the target level. An item-level semantic sequence combines contextualized token states across levels within each item, providing complementary item semantics through dual cross-attention. LESSRec+ reduces encoder computation through level-specific encoder groups while retaining the same parameter count as LESSRec. Experiments on three public datasets with three tokenizers demonstrate both models' recommendation effectiveness, while component ablations and efficiency analyses support the proposed design and LESSRec+'s computational benefits.

References

[1] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*. https://arxiv.org/abs/2010.00904

[2] Kairui Fu, Tao Zhang, Shuwen Xiao, Ziyang Wang, Xinming Zhang, Chenchi Zhang, Yuliang Yan, Junjun Zheng, Xiangheng Kong, Shengyu Zhang, Kun Kuang, and Yuning Jiang. 2026. FORGE: Forming Semantic Identifiers for Generative Retrieval in Industrial Datasets. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26)*. 8858–8869.

[3] Xian Guo, Ben Chen, Siyuan Wang, Ying Yang, Mingyue Cheng, Chenyi Lei, Yuqing Ding, and Han Li. 2026. OneSug: The Unified End-to-End Generative Framework for E-Commerce Query Suggestion. *Proceedings of the AAAI Conference on Artificial Intelligence* 40, 17 (2026), 14774–14782.

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2016). https://arxiv.org/abs/1511.06939

[5] Minjie Hong, Yan Xia, Zehan Wang, Jieming Zhu, Ye Wang, Sihang Cai, Xiaoda Yang, Quanyu Dai, Zhenhua Dong, Zhimeng Zhang, and Zhou Zhao. 2025. EAGER-LLM: Enhancing Large Language Models as Recommenders through Exogenous Behavior-Semantic Integration. In *Proceedings of the ACM on Web Conference 2025*. 2754–2762. doi:10.1145/3696410.3714933

[6] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*. 1162–1171.

[7] Yupeng Hou, Jiacheng Li, Xiangjun Fu, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2026. 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)*, Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (Eds.). 3251–3265.

[8] Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. 2025. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 956–966.

[9] Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H. Chi, Julian Mcauley, and Derek Zhiyuan Cheng. 2025. ActionPiece: Contextually Tokenizing Action Sequences for Generative Recommendation. In *Proceedings of the 42nd International Conference on Machine Learning*. 24004–24024.

[10] Yang Hu, Jiayi Guo, Jingui Ma, Ning Li, Jiangling Qin, Yanming Li, Yang Deng, Xiaoshuang Chen, and Kaiqiao Zhan. 2026. Once Generated, Ranked: End-to-End Generative Slate Recommendation with Unified Semantic-Collaborative IDs. *arXiv preprint arXiv:2608.17613* (2026). https://arxiv.org/abs/2608.17613

[11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. doi:10.1109/TPAMI.2010.57

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[13] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive Image Generation Using Residual Quantization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11513–11522.

[14] Xinyu Lin, Pengyuan Liu, Wenjie Wang, Yicheng Hu, Chen Xu, Fuli Feng, Qifan Wang, and Tat-Seng Chua. 2026. Bringing Reasoning to Generative Recommendation Through the Lens of Cascaded Ranking. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. 8939–8949.

[15] Chang Liu, Yimeng Bai, Xiaoyan Zhao, Yang Zhang, Fuli Feng, and Wenge Rong. 2025. DiscRec: Disentangled Semantic-Collaborative Modeling for Generative Recommendation. *arXiv preprint arXiv:2506.15576* (2025).

[16] Enze Liu, Bowen Zheng, Cheng Ling, Lantao Hu, Han Li, and Wayne Xin Zhao. 2025. 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 (SIGIR '25)*. 729–739.

[17] Xinchen Luo, Jiangxia Cao, Tianyu Sun, Jinkai Yu, Rui Huang, Wei Yuan, Hezheng Lin, Yichen Zheng, Shiyao Wang, Qigen Hu, Changqing Qiu, Jiaqi Zhang, Xu Zhang, Zhiheng Yan, Jingming Zhang, Simin Zhang, Mingxing Wen, Zhaojie Liu, and Guorui Zhou. 2025. QARM: Quantitative Alignment Multi-Modal Recommendation at Kuaishou. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 5915–5922.

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

[19] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[20] Pingjun Pan, Tingting Zhou, Peiyao Lu, Tingting Fei, Hongxiang Chen, and Chuanjiang Luo. 2026. Hi-SAM: A Hierarchical Structure-Aware Multi-Modal Framework for Large-Scale Recommendation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26)*. 7830–7841.

[21] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. 2025. Tokenrec: Learning to Tokenize Id for Llm-Based Generative Recommendations. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[22] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2025).

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[24] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, and Jonah Samost. 2023. Recommender Systems with Generative Retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[25] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.Com. *IEEE Internet Computing* 21, 3 (2017), 12–18.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450. https://arxiv.org/abs/1904.06690

[27] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2023. Learning to Tokenize for Generative Retrieval. In *Advances in Neural Information Processing Systems*, Vol. 36. 46345–46361.

[28] Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. IDGenRec: LLM-RecSys Alignment with Textual ID Learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. 355–364.

[29] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[30] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, Vol. 30.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[32] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2400–2409.

[33] Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. 2026. CoFiRec: Coarse-to-Fine Tokenization for Generative Recommendation. In *Proceedings of the 20th ACM Conference on Recommender Systems (RecSys '26)*. 115–124.

[34] Zhenxiang Xu, Jiawei Chen, Sirui Chen, Yong He, Jieyu Yang, Chuan Yuan, Ke Ding, and Can Wang. 2026. Trie-Aware Transformers for Generative Recommendation. *arXiv preprint arXiv:2602.21677* (2026).
[35] Yuhao Yang, ZhI JI, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, shuanglong li, and LIU LIN. 2025. Sparse Meets Dense: Unified Generative Recommendations with Cascaded Sparse-Dense Representations. In *Advances in Neural Information Processing Systems*, Vol. 38, Main Conference. 93746–93770.
[36] Junchao Zeng, Junzhang Zhu, Junyang Chen, Yudong Li, Wei Liu, Chengxiang Zhuo, and Zang Li. 2026. Bridging the Structural Gap: Adapting Autoregressive Generation for Recommendation. *arXiv preprint arXiv:2607.21028* (2026).
[37] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM '22)*. 4722–4726.
[38] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21)*. 4653–4664.
[39] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting Large Language Models by Integrating Collaborative Semantics for Recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 1435–1448.
[40] Bowen Zheng, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2026. Universal Item Tokenization for Transferable Generative Recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26)*. 2509–2520.
[41] 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. 2025. OneRec Technical Report. *arXiv preprint arXiv:2506.13695* (2025).
[42] Jieming Zhu, Mengqun Jin, Qijiong Liu, Zexuan Qiu, Zhenhua Dong, and Xiu Li. 2024. CoST: Contrastive Quantization Based Semantic Tokenization for Generative Recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems (RecSys '24)*. 969–974.

A Why Same-Level History Helps

We seek to explain why same-level history can be more useful for target-token prediction from two perspectives: predictive information and learning signal. We first introduce assumptions on the residual structure and temporal dependence, then examine their implications for prediction and learning.

Assumptions. First, we assume that successive residual components are sufficiently separated for quantization to recover them. Write the tokenizer latent as $\mathbf{u}_i = \sum_{r=1}^{L} \mathbf{e}_{r,Z_{i,r}} + \boldsymbol{\xi}_i$, where $Z_{i,r}$ is a discrete component. With minimum codeword separation $\Delta_r = \min_{a\neq b} \|\mathbf{e}_{r,a} - \mathbf{e}_{r,b}\|_2 > 0$, assume

$$\left\| \sum_{j>r} \mathbf{e}_{j,Z_{i,j}} + \boldsymbol{\xi}_i \right\|_2 < \Delta_r/2 \qquad \text{for all } i, r. \tag{13}$$

This condition ensures $s_{i,r} = Z_{i,r}$. Inductively, the residual is the correct codeword plus a tail $\mathbf{t}$ with $\|\mathbf{t}\|_2 < \Delta_r/2$. Every alternative is farther away, at distance at least $\Delta_r - \|\mathbf{t}\|_2 > \|\mathbf{t}\|_2$. Scalar codebooks $\{-3^{L-r}, 3^{L-r}\}$ with zero final residual provide one example satisfying this condition.

Second, we assume that the component trajectories $Z_{1:T+1,r}$ are independent across levels, with each following a first-order Markov chain. This describes temporal dependence within each residual level; RQ-VAE training alone does not guarantee it. Write $\mathcal{H} = (H_1, \ldots, H_L)$, $Y_k = s_{T+1,k}$, and $C_k = Y_{<k}$. All logarithms are natural.

Predictive information. Under these assumptions, we can compare the predictive value of different histories through their lowest achievable token log loss:

$$\mathcal{R}^*(A) = \inf_q \mathbb{E}[-\log q(Y_k \mid C_k, A)].$$

Here, q ranges over all conditional predictors. The following proposition relates the benefit of same-level history to its temporal predictive information.

Proposition A.1 (Predictive information). *Under the stated assumptions, for $r \neq k$,*

$$\begin{aligned} \mathcal{R}^*(\mathcal{H}) = \mathcal{R}^*(H_k) &= \mathrm{H}(Y_k \mid s_{T,k}), \\ \mathcal{R}^*(H_r) - \mathcal{R}^*(H_k) &= \mathrm{I}(Y_k; s_{T,k}) \geq 0. \end{aligned} \tag{14}$$

The gap is positive when the previous same-level token carries predictive information about the target.

Proof. Expected log loss equals conditional entropy plus a nonnegative expected KL divergence, so its infimum is conditional entropy. Independence of level trajectories and the Markov property give

$$\begin{aligned} P(Y_k \mid C_k, \mathcal{H}) &= P(Y_k \mid s_{T,k}), \\ P(Y_k \mid C_k, H_r) &= P(Y_k), \qquad r \neq k. \end{aligned}$$

Substituting these distributions into the conditional entropies gives the first equality and the mutual-information gap. □

Same-level history therefore retains the predictive information of the full history under these assumptions. Any advantage over full-history attention must arise from how a restricted model uses that information.

Learning signal. We next examine this distinction with a single-head model. Use the binary construction above with $L > 1$ and $T = 1$. Let historical signs X_r be independent and uniform, and set $Y = X_k E$ with independent noise $\Pr(E = -1) = p \in (0, 1/2)$. Other levels have uniform next states independent of the history and E, so C_k is independent of $(X_1, \ldots, X_L, Y)$. A one-layer, level-isolated encoder with fixed sign embeddings and identity value projection returns X_r. A single decoder head uses fixed finite scores depending only on level, with $a_r = \mathrm{softmax}(\ell)_r > 0$. Its two contexts are

$$z_{\text{same}} = X_k, \qquad z_{\text{all}} = mX_k + N, \quad m = a_k, \quad N = \sum_{r\neq k} a_r X_r. \tag{15}$$

Here, N collects the other levels' contributions. The classifier is $\Pr_w(Y = 1 \mid z) = \sigma(wz)$, where $|w| \leq W$, $W > 0$, and σ is the sigmoid. We omit layer normalization, feed-forward sublayers, EOS, and item memory to isolate the effect of mixing levels. The next proposition compares both prediction loss and the gradient used to learn w.

Proposition A.2 (Prediction loss and initial learning signal). *Let $\mathcal{L}(w; z) = \mathbb{E}\log(1 + e^{-Ywz})$. Then*

$$\inf_{|w|\leq W} \mathcal{L}(w; z_{\text{all}}) \geq \inf_{|w|\leq W} \mathcal{L}(w; z_{\text{same}}). \tag{16}$$

At $w = 0$, define the per-example gradient $G = \partial_w \log(1 + e^{-Ywz})$ and $\tau = 1 - 2p$. For the average $\overline{G}_n$ of n independent examples, let $\mathrm{SNR}_n = (\mathbb{E}\overline{G}_n)^2/\mathrm{Var}(\overline{G}_n)$. *Then*

$$\begin{aligned} \mathbb{E}G_{\mathrm{all}} &= -m\tau/2, & \mathbb{E}G_{\mathrm{same}} &= -\tau/2, \\ \mathrm{SNR}_n^{\mathrm{all}} &= \frac{nm^2\tau^2}{m^2(1-\tau^2) + \sum_{r\neq k} a_r^2} \\ &< \frac{n\tau^2}{1-\tau^2} = \mathrm{SNR}_n^{\mathrm{same}}. \end{aligned} \tag{17}$$

Proof. For the loss comparison, N is independent of (X_k, Y) and has mean zero. Conditional Jensen's inequality therefore gives $\mathcal{L}(w; mX_k + N) \geq \mathcal{L}(mw; X_k)$. Taking infima over $[-mW, mW] \subseteq [-W, W]$ proves the bound. For the learning comparison, $G = -Yz/2$ at initialization. Independence yields

$$\mathrm{Var}(G_{\mathrm{all}}) = \tfrac{1}{4}\left[m^2(1-\tau^2) + \sum_{r\neq k} a_r^2\right].$$

The aligned case has $m = 1$ and $N = 0$. Averaging divides the variance by n; positive mismatched weights make the SNR inequality strict. □

The second proposition explains how mixing levels can weaken the learning signal even when all predictive information remains available in the history. With equal scores, the expected gradient magnitude falls by a factor of L, and $\mathrm{SNR}_n^{\mathrm{all}} = n\tau^2/(L - \tau^2)$. The analysis thus supports same-level access under the stated assumptions. Its learning conclusion concerns initialization, not global convergence; learned projections and content-dependent scores may change the comparison.

B Residual Quantization for Item Tokenization

We describe the offline tokenizers used to construct the fixed semantic IDs. Let $\mathbf{a}_i$ denote the content embedding of item v_i, L the number of semantic tokens, and K the number of token values at each level. Tokenizer parameters are trained separately from the recommendation model.

B.1 RQ-VAE

Following residual quantization for image generation [13] and its use in TIGER [24], RQ-VAE encodes the content embedding into a continuous latent vector $\mathbf{u}_i = \mathcal{E}_{\mathrm{q}}(\mathbf{a}_i)$. Each level r has a codebook $C_r = \{\mathbf{e}_{r,k}\}_{k=0}^{K-1}$. Starting with $\boldsymbol{\rho}_{i,0} = \mathbf{u}_i$, quantization selects the nearest codeword and subtracts it from the residual:

$$\begin{aligned} s_{i,r} &= \arg\min_{0\leq k<K} \|\boldsymbol{\rho}_{i,r-1} - \mathbf{e}_{r,k}\|_2^2, \\ \boldsymbol{\rho}_{i,r} &= \boldsymbol{\rho}_{i,r-1} - \mathbf{e}_{r,s_{i,r}}, \qquad r = 1, \ldots, L. \end{aligned} \tag{18}$$

The selected indices form $\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,L})$. Earlier codewords approximate the latent representation, while later codewords describe the remaining residual, producing a coarse-to-fine semantic-level structure. The quantized representation and reconstructed content embedding are

$$\widehat{\mathbf{u}}_i = \sum_{r=1}^{L} \mathbf{e}_{r,s_{i,r}}, \qquad \widehat{\mathbf{a}}_i = \mathcal{D}_{\mathrm{q}}(\widehat{\mathbf{u}}_i). \tag{19}$$

The encoder, decoder, and codebooks are jointly trained using reconstruction, codebook, and commitment losses [30]:

$$\begin{aligned} \ell_{i,r}^{\mathrm{vq}} &= \mathrm{MSE}(\mathrm{sg}[\boldsymbol{\rho}_{i,r-1}], \mathbf{e}_{r,s_{i,r}}) \\ &\quad + \beta\,\mathrm{MSE}(\boldsymbol{\rho}_{i,r-1}, \mathrm{sg}[\mathbf{e}_{r,s_{i,r}}]), \\ \mathcal{L}_{\mathrm{q}} &= \frac{1}{N}\sum_{i=1}^{N}\left[\mathrm{MSE}(\mathbf{a}_i, \widehat{\mathbf{a}}_i) + \frac{\lambda}{L}\sum_{r=1}^{L}\ell_{i,r}^{\mathrm{vq}}\right]. \end{aligned} \tag{20}$$

Here, N is the number of items, MSE averages squared errors over vector coordinates, and sg denotes stop-gradient. The codebook term fits selected vectors to the residuals, while the commitment term encourages the encoded residuals to remain close to their selected vectors. The implementation uses $\beta = 0.25$, $\lambda = 1$, a straight-through estimator for quantization, and K-means initialization of the codebooks from an initial batch of encoded residuals. Codebooks are subsequently updated by gradient descent; K-means initialization alone does not make this tokenizer a residual K-means tokenizer.

B.2 RQ-Kmeans

RQ-Kmeans constructs hierarchical semantic IDs by clustering residual representations at successive levels [10, 17]. For the residuals entering level r, standard residual K-means learns a codebook through

$$\min_{C_r} \sum_{i=1}^{N} \min_{\mathbf{e}\in C_r} \|\boldsymbol{\rho}_{i,r-1} - \mathbf{e}\|_2^2. \tag{21}$$

Each item is assigned to its nearest centroid, which is subtracted before quantizing the next residual. The resulting codeword indices form the semantic IDs, and the sum of the selected centroids approximates the input representation. Earlier levels capture larger-scale variation, while later levels refine the remaining residual.

Our implementation combines this residual formulation with a learned autoencoder and K-means-guided conditional subcodebooks. Let $C_{r,g} = \{\mathbf{e}_{r,g,k}\}_{k=0}^{K_{r,g}-1}$ denote subcodebook g at level r, containing $K_{r,g}$ vectors. Earlier semantic tokens determine the subcodebook through a routing function Γ_r:

$$\begin{aligned} g_{i,r} &= \Gamma_r(s_{i,1}, \ldots, s_{i,r-1}), \\ k_{i,r} &= \arg\min_{0\leq k<K_{r,g_{i,r}}} \|\boldsymbol{\rho}_{i,r-1} - \mathbf{e}_{r,g_{i,r},k}\|_2^2, \\ \boldsymbol{\rho}_{i,r} &= \boldsymbol{\rho}_{i,r-1} - \mathbf{e}_{r,g_{i,r},k_{i,r}}. \end{aligned} \tag{22}$$

The initial residual is $\boldsymbol{\rho}_{i,0} = \mathcal{E}_{\mathrm{q}}(\mathbf{a}_i)$, and the first-level subcodebook is shared by all items. The selected subcodebook and local codeword index jointly identify the semantic token at each level. As in RQ-VAE, the selected vectors are summed and decoded to reconstruct the content embedding.

C Additional Experimental Analysis

Figure 7 further examines LESSRec design variants on Games. *w/o MLP* directly uses the mean-pooled representations as item vectors, while *Cat.+MLP* replaces within-item mean pooling with concatenation before the MLP. Removing the MLP reduces performance in both metrics, supporting its role in refining pooled representations for recommendation. In contrast, concatenation followed by the MLP outperforms the mean-pooling-based LESSRec reference in both metrics. This suggests that retaining the individual

semantic-level representations before learned fusion can benefit item-level modeling; mean pooling is therefore not necessarily the best aggregation choice.

We also evaluate *w/o EOS*, which removes EOS from the encoder input. Performance remains close to LESSRec in both metrics. Although the EOS state can aggregate information across semantic levels and is accessible to the decoder, its removal produces no consistent performance gain. These results suggest that a single shared EOS state, despite its access to all semantic levels, does not introduce cross-level interference strong enough to noticeably affect recommendation performance in this setting. Historical semantic tokens cannot attend to EOS under LIS, so their encoded states remain level-isolated even when EOS is retained as the sequence terminator.

We further compare *QKV LIS*, *QKV LAC*, and *QKV LIS+LAC*, which compute attention in separate semantic-level blocks in encoder self-attention, decoder cross-attention, or both, respectively, while retaining shared QKV projections. Splitting encoder-side QKV computation has little effect on performance, whereas splitting decoder-side computation leads to a clear decline. A possible explanation concerns attention to the shared item-level memory: in LESSRec, target positions at different semantic levels access the same item-level semantic sequence through a shared cross-attention module. Reorganizing decoder attention into separate level-specific blocks may affect how these queries access and combine the shared item-level context with their corresponding same-level histories, potentially weakening the complementarity of the two branches.

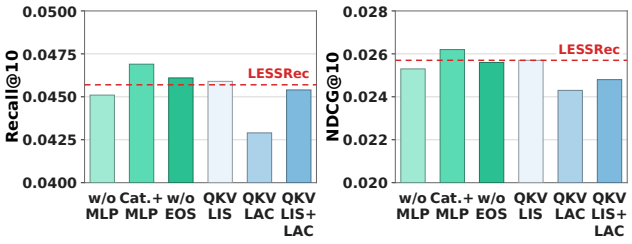


Figure 7: Ablation of LESSRec design choices on Games.