Trie-Aware Transformers for Generative Recommendation

Zhenxiang Xu[†‡]
Zhejiang University
Hangzhou, China
zhenxiangxu@zju.edu.cn

Sirui Chen[†‡]
Zhejiang University
Hangzhou, China
chenthree@zju.edu.cn

Yong He
Ant Group
Hangzhou, China
heyong.h@antgroup.com

Jieyu Yang
Ant Group
Hangzhou, China
jieyu.yjy@antgroup.com

Chuan Yuan[*]
Ant Group
Hangzhou, China
yuanzheng.xy@antgroup.com

Ke Ding
Ant Group
Hangzhou, China
dingke.dk@antgroup.com

Jing Cao[†]
Zhejiang University
Hangzhou, China
cjcongjia@zju.edu.cn

Can Wang[†§]
Zhejiang University
Hangzhou, China
wcan@zju.edu.cn

Jiawei Chen[* † ‡§]
Zhejiang University
Hangzhou, China
sleepyhunt@zju.edu.cn

Abstract

Generative recommendation (GR) aligns with advances in generative AI by casting next-item prediction as token-level generation rather than score-based ranking. Most GR methods adopt a two-stage pipeline: (i) *item tokenization*, which maps each item to a sequence of discrete, hierarchically-organized tokens; and (ii) *autoregressive generation*, which predicts the next item's tokens conditioned on the tokens of user's interaction history. Although hierarchical tokenization induces a prefix tree (trie) over items, standard autoregressive modeling with conventional Transformers often flattens item tokens into a linear stream and overlooks the underlying topology. To address this, we propose TrieRec, a trie-aware generative recommendation method that augments Transformers with structural inductive biases via two positional encodings. First, a *trie-aware absolute positional encoding* aggregates a token's (node's) local structural context (*e.g.,* depth, ancestors, and descendants) into the token representation. Second, a *topology-aware relative positional encoding* injects pairwise structural relations into self-attention to capture topology-induced semantic relatedness. TrieRec is also model-agnostic, efficient, and hyperparameter-free. In our experiments, we implement TrieRec within three representative GR backbones, achieving notably improvements of 8.83% on average across four real-world datasets. Our code is available on `https://github.com/cynthia-shengjia/Generative-Recommendation-Benchmark`.

1 Introduction

Motivated by recent advances in generative AI [4, 1], *Generative Recommendation* (GR) has recently emerged as a promising paradigm for recommender systems [46, 6]. Unlike discriminative approaches that score query–item pairs [12, 9, 11], GR directly generates target items in an end-to-end generative manner [23, 34]. A typical GR pipeline comprises two stages [15, 35]: (i) item tokenization, which

[*]Corresponding authors: Jiawei Chen and Chuan Yuan.
[†]State Key Laboratory of Blockchain and Data Security, Zhejiang University.
[‡]College of Computer Science and Technology, Zhejiang University.
[§]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

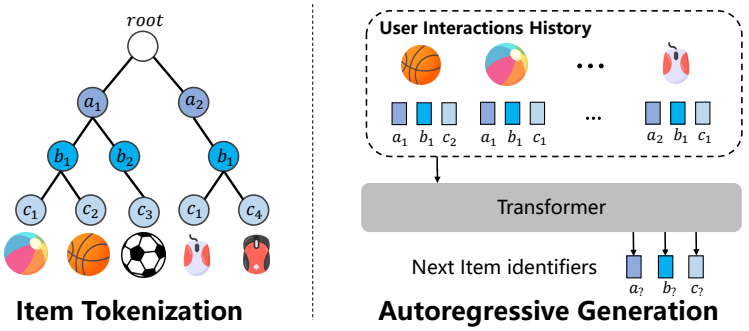


Figure 1: Generative recommendation pipeline. Item tokenization maps items to semantic token sequences via hierarchical quantization, forming a trie structure. Autoregressive generation employs a Transformer to predict the next item's token sequence conditioned on user interaction history.

learns a codebook that maps each item to a sequence of semantic identifiers (discrete tokens); and (ii) autoregressive generation, which predicts the next token conditioned on a token sequence constructed from the user's interaction history. In practice, item tokenization is often realized with hierarchical quantization or clustering (*e.g.,* RQ-VAE[13], RQ-kmeans [18]), producing coarse-to-fine semantic tokens. These tokens naturally form a *hierarchical prefix tree* [32, 52] (trie, *cf.* Figue 1), where each item corresponds to a unique path from root to leaf. Consequently, GR can be viewed as layer-wise child-node (next-token) selection along the trie, rather than direct selection from the entire item corpus, thus restricting the per-step search space and keeping computation affordable [2, 33]. This mechanism aligns the principles of conventional cascade recommendation systems [5], which progressively narrow candidates from coarse to fine via multiple models; while GR advances this concept by unifying the process within a single end-to-end generative framework.

Despite recent progress, existing GR approaches suffer from a critical limitation: *the inherent tree structure of item tokens is often overlooked during autoregressive modeling.* Specifically, existing GR models mainly employ Transformers [31] that flattens hierarchical token sequences into a linear stream, relying solely on self-attention mechanisms to learn token semantics and dependencies. This neglects the rich topological and relational information encoded in the trie structure. In fact, the trie fundamentally provides two sources of valuable knowledge: (i) *Structural position*, which situates a token by its depth and item path, and anchors it in the surrounding hierarchical context. Considering a token (identifier) may be reused across items, cues from ancestors and descendants can enrich its semantics and enhance token modeling. (ii) *Relational topology*, which captures structure-aware relations among tokens (*e.g.,* ancestor–descendant links and shared-prefix siblings), inducing different degrees and types of semantic relatedness. For example, tokens along the same root-to-leaf path belong to a single item and typically exhibit strong semantic dependency; whereas tokens that share a closer lowest common ancestor (LCA) [3] tend to originate from similar items and are more closely related than tokens from distant branches of the trie.

To bridge this gap, we propose TrieRec, a trie-aware generative recommendation method that augments Transformers with structural inductive bias via tailored positional encodings:

- **Trie-aware absolute positional encoding**: a hierarchy-dependent neural encoder that aggregates a node's local structural context (*e.g.,* depth, ancestors and descendants) into an embedding added to the token representation. This equips tokens with awareness of their hierarchical position and item membership, and directly incorporates useful information from structurally related nodes.
- **Topology-aware relative positional encoding**: a pairwise structural bias for attention that encode topological relations between token pairs. Specifically, for each node pair, we compute their lowest common ancestor (LCA) and the distances from each token to the LCA, and map these features to a learnable attention-bias term to capture topological-induced semantic relatedness.

Beyond structural awareness, TrieRec also offers practical advantages: (i) model-agnostic: easily plugged into a variety of GR backbones; (ii) efficiency: limited additional computational overhead; (iii) hyperparameter-free: no new tunable hyperparameters. To empirically validate TrieRec, we integrate it into three representative GR backbones (including TIGER, CoST, LETTER), achieving an average relative improvement of 8.83% on four real-world datasets.

In summary, the contributions of this work are:

- We highlight the critical importance of explicitly encoding the trie structure in the autoregressive modeling of generative recommendation.
- We propose TrieRec, which augments Transformers with two tailored positional encodings, capturing both tokens' structural positions and pairwise topological relations in the trie.
- We conduct extensive experiments to validate the effectiveness of TrieRec, showing notable improvements over state-of-the-art baselines with limited additional computational cost.

2 Preliminary

Problem Definition. Following recent work [23, 34, 51], we focus on sequential recommendation [11, 28], a widely adopted setting in real-world recommender systems. Let $\mathcal{I}$ denote the item set. Given a user's historical interaction sequence $S = \{i_1, i_2, \ldots, i_T\}$, where $i_t \in \mathcal{I}$ denotes the t-th item in the sequence, the goal is to predict the next item i_{T+1} that the user is most likely to interact with.

Generative Recommendation. Recent advances in generative AI have stimulated extensive interest in *generative recommendation* (GR). Different from discriminative recommenders that estimate matching scores for query-item pairs, GR formulates recommendation as an autoregressive generation problem, where the model directly generates the next item in a tokenized space. Most GR methods follow a two-stage pipeline [23, 34, 51]:

Item Tokenization. This stage encodes each item i into a sequence of structured discrete tokens (*i.e.,* semantic identifiers) $\mathbf{c}_i = [c_{i,1}, c_{i,2}, \ldots, c_{i,L}]$. A representative approach is RQ-VAE [13], which recursively quantizes item embeddings from coarse to fine granularity. Recent studies have further explored improved tokenization strategies, such as collaborative-enhanced or learnable tokenization [34], while often adhering to the hierarchical encoding principle.

Such hierarchical tokens naturally induce a *prefix tree* (trie; cf. Figure 1), where each item corresponds to a unique root-to-leaf path and different items may share common prefix tokens. Such hierarchical trie-structured representation offers several advantages: 1) The tree structure is well suited for generative modeling. Intuitively, generation can be interpreted as layer-wise child-node (next-token) selection along the trie, which restricts the search space at each step and keeps modeling computationally manageable. 2) The tree structure facilitates collaborative semantic modeling. items sharing longer prefix paths tend to be semantically closer, and the trie explicitly captures such relatedness in the shared prefix token generative probability.

Autoregressive Generation. This stage models the token sequence of the next item in an autoregressive manner. Specifically, a user's historical interactions are represented as a token stream:

$$\mathbf{X} = [c_{1,1}, c_{1,2}, \ldots, c_{1,L}, \ldots, c_{T,1}, c_{T,2}, \ldots, c_{T,L}], \tag{1}$$

where $(c_{t,1}, c_{t,2}, \ldots, c_{t,L})$ denotes the token sequence of the t-th interacted item. A GR model $\mathcal{M}$ is then employed to model the probability of the next-item token sequence conditioned on the history:

$$P_{\mathcal{M}}(\mathbf{y} \mid \mathbf{X}) = \prod_{t=1}^{L} P_{\mathcal{M}}(y_t \mid y_{<t}, \mathbf{X}). \tag{2}$$

where $\mathbf{y} = [y_1, y_2, ..., y_L]$ denotes the token sequence of the predicted next item. The generative probability of the next item factorizes into the product of token-level conditionals.

Given the empirical successes of Transformer architectures in sequence modeling, Transformer-based backbones are commonly adopted in GR. The Transformer relies on self-attention to model token dependencies, which can be summarized as follows:

$$\mathbf{o}_u = \sum_v A_{uv}(\mathbf{e}_v \mathbf{W}^V) \tag{3}$$

$$A_{uv} = \text{Softmax}_v\left(\frac{(\mathbf{e}_u \mathbf{W}^Q)(\mathbf{e}_v \mathbf{W}^K)^\top}{\sqrt{d}}\right) \tag{4}$$

where $\mathbf{e}_u, \mathbf{e}_v \in \mathbb{R}^d$ are the embedding of tokens at positions u and v respectively, and $\mathbf{W}^Q$,$\mathbf{W}^K$,$\mathbf{W}^V$ are learnable projection matrices. Each token u aggregates information from related context tokens v in the sequence according to the attention weights A_{uv}, which reflect semantic dependency measured by the similarity of the transformed token embeddings, resulting in the aggregated output representation $\mathbf{o}_u$. While vanilla Transformers are powerful, they are agnostic to token order. Linear positional encodings are typically employed. For example, the conventional model used by TIGER encodes the token distance within the sequence [22].

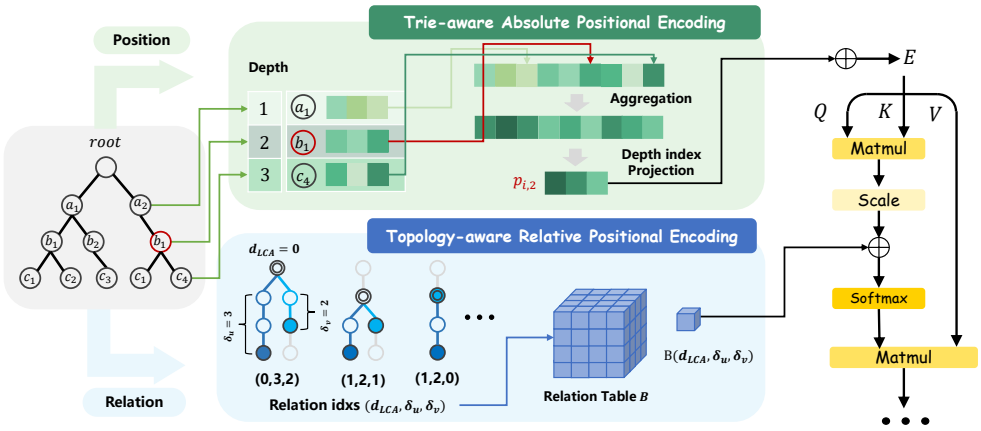


Figure 2: The illustration of our proposed trie-aware absolute positional encoding and topology-aware relative positional encoding in TrieRec.

Limitations of Existing GR Methods. Although the item tokens are often organized as a trie, existing GR models often treat the token stream $\mathbf{X}$ as a flat sequence during autoregressive generation, without explicitly modeling the underlying trie structure. This omission discards two types of crucial structural signals:

(i) Absolute structural position in trie. A token's position in the trie, signifies its depth and root-to-leaf path, as well as anchors it within the hierarchical context. This information is particularly important because token identifiers can be reused across different items. For instance, under RQ-VAE-style tokenization, the same token may just reflect quantized residuals rather than a specific fixed semantic meaning as in natural language vocabularies. Therefore, incorporating cues from ancestor and descendant nodes can help disambiguate token semantics and improve token semantic modeling.

(ii) Relational topology between token pairs. The structural relations in the trie encode varying strengths and types of semantic relatedness. For example, tokens along the same root-to-leaf path belong to the same item and typically exhibit strong semantic dependency. Their hierarchical relationships also exhibit different patterns. Moreover, token pairs with a closer lowest common ancestor (LCA) (*e.g.,* token c_2 of item *basketball* and token c_3 of item *football*) tend to originate from more similar items and are more related than those located in distant branches of the trie (*e.g.,* tokens c_2 of item *basketball* and token c_1 of item *mouse*).

Although the trie structure is central to tokenization, its potential is underexplored in autoregressive generation. To the best of our knowledge, only a single work [26] has attempted to incorporate trie information into a contrastive loss objective. However, their approach is not well aligned with contemporary generative modeling, as it requires computing and storing distinct embeddings for every tree node, incurring substantial time and memory costs. Moreover, their modeling only focuses on local parent–child relationships rather than fully exploiting token structural positions and higher-order relations. Our empirical investigations also reveal their performance is suboptimal (*cf.* Appendix E).

3 Methodology

In this section, we present TrieRec, a trie-aware generative recommendation framework. TrieRec introduces two positional encodings: trie-aware absolute positional encoding (Subsection 3.1) and topology-aware relative positional encoding (Subsection 3.2). We then discuss practical advantages of TrieRec and relate it to relevant prior work (Subsection 3.3).

3.1 Trie-aware absolute positional encoding (TAE)

Token structural positional information provides rich semantic signals, indicating where a token resides in the hierarchical tree (its layer and path) and anchoring it within the structural context. To enrich token embeddings with such structural information, we define a dedicated neural encoder that

generates a position encoding for each token $c_{i,l}$ as follows:

$$\mathbf{x}_i^{con} \overset{①}{=} [\mathbf{e}_{i,1}^\top; \mathbf{e}_{i,2}^\top; \ldots; \mathbf{e}_{i,L}^\top]^\top \in \mathbb{R}^{L \cdot d} \tag{5}$$

$$\hat{\mathbf{x}}_i \overset{②}{=} \text{MLP}(\mathbf{x}_i^{con}) \in \mathbb{R}^{L \cdot d} \tag{6}$$

$$\mathbf{p}_{i,l} \overset{③}{=} \mathbf{W}_l \hat{\mathbf{x}}_i + \mathbf{b}_l \quad l = 1, \ldots, L \tag{7}$$

where $\mathbf{e}_{i,l}$ denotes the original semantic embedding of token $c_{i,l}$ at a given Transformer layer. Our encoder consists of three stages: ① Concatenation: We first collect the embeddings of the token's hierarchical context neighbors, including ancestors and descendants ($c_{i,1}, c_{i,2}, ..., c_{i,L}$), and concatenate them into a flat vector $\mathbf{x}_i^{con}$. ② Aggregation: We pass $\mathbf{x}_i^{con}$ through a MLP to model information aggregation and interaction among its components. ③ Projection: We apply a hierachy-aware projector to generate token positional encodings based on the token layer index.

The rationale of these designs can be explained as follows: 1) In a tree, a node's (token's) position is uniquely determined by the sequence of its ancestors. Accordingly, we explicitly encode the full longitudinal context to capture positional information. Descendant information belonging to the same item is also included. Intuitively, those tokens belonging to same item path can naturally share closely semantic dependency, offering item-level semantics and benefiting token semantic modeling. 2) Unlike permutation-invariant attention, an MLP is position-dependent: each dimension of input token has its own learnable weights to aggregate information across dimensions, leveraging the hierarchical positions of tokens. This also enables simultaneous cross-token and cross-dimension interactions, allowing more fine-grained information aggregation. 3) To further capture different semantic properties across token layers, a hierarchy-aware projector generates the token positional encoding using different parameters for different layers, thereby modeling their distinct characteristics.

The learned position encoding $\mathbf{p}_{i,l}$ is further added to the original token embedding $\mathbf{e}_{i,l}$ in the transformer, enriching its representation with the structural context:

$$\hat{\mathbf{e}}_{i,l} = \mathbf{e}_{i,l} + \mathbf{p}_{i,l} \tag{8}$$

In practice, The MLP can be implemented as a simple two-layer network with an activation function GELU and layer normalization. For efficiency, the TAE is directly added to the initial token embeddings at the input of the Transformer. We find that this lightweight implementation has yielded meaningful improvements with minimal overhead.

3.2 Topology-aware relative positional encoding (TRE)

We further exploit the topological patterns of token pairs. Different topological relations convey different strengths and kinds of semantic dependencies. We adopt two guiding principles: (i) Topology-aware design: explicitly model multiple topological features (*e.g.,* structural distance, the lowest common ancestor, and relative hierarchical relations) for token pairs. (ii) Computational efficiency: keep the design concise and scalable, avoiding heavy per-pair computations.

To achieve this, for every pair of tokens (u, v) within a sequence, we compute three topological features: (i) $d_{u,v}^{LCA}$: the tree depth of their LCA ; (ii) δ_u: the distances from u to the LCA; (iii) δ_v : the distance from v to the LCA. These three features jointly capture rich structural relations between the tokens. For example, the depth of the LCA reflects the length of their shared prefix along the item paths, indicating semantic relatedness; δ_u and δ_v quantify how far each token is from the LCA, revealing their relative distance in the tree. By comparing δ_u and δ_v, we can infer which token lies higher in the hierarchy. While more sophisticated strategies, *e.g.,* directly encoding the semantic embeddings of the path nodes along the tree, could carry richer information, they would incur substantial computational cost.

With the three-dimensional characterization of token structural relations, we encode this information into a learnable bias that captures the semantic dependencies among tokens. Specifically, we revise the original self-attention scores by adding a topology-aware bias:

$$\tilde{A}_{u,v} = \text{Softmax}_v \left(\frac{(\mathbf{e}_u \mathbf{W}^Q)(\mathbf{e}_v \mathbf{W}^K)^\top}{\sqrt{d}} + B(d_{u,v}^{LCA}, \delta_u, \delta_v) \right) \tag{9}$$

where $B(d_{u,v}^{LCA}, \delta_u, \delta_v)$ denotes a learnable scalar term indexed by the triple $(d_{u,v}^{LCA}, \delta_u, \delta_v)$. The revised attention score $\tilde{A}_{u,v}$ thus combines the original self-attention score with the topology-aware bias, effectively injecting topological information into the transformer. For efficiency, the same parameter set B is shared across layers, so B can be computed once and reused rather than recomputed at each layer.

3.3 Discussions

TrieRec has the following properties:

Model-agxnostic. Notably, our positional encodings are agnostic to the concrete tokenization model or generative model. Thus, TrieRec can be easily plugged into a variety of recent methods by adding two simple terms to the token embeddings and attention computations (as in Eq.(8) and Eq.(9)), yielding improvements by integrating the trie information. In our experiments, we implemented TrieRec with three representative generative models, e.g., TIGER, CoST, and LETTER, and observed consistent and significant improvements.

Efficiency. TrieRec only incurs limited additional computational cost. Considering a user sequence with T items and $T \cdot L$ tokens, and a transformer model with d dimemsional embeddings, h heads and k layers. The TAE involves the aggregation of each tokens with their hierarchal context with MLP; the cost for generating each token encoding is $O(L^2d^2)$. The overall cost of TAE for the sequence is $O(TL^2d^2)$, considering we do not require to recompute B for tokens sharing same token path. TRE requires computing the LCA for each token pair in the sequence, with complexity $O(\log L)$ per pair, which can be further accelerated via memorizing reduplication. The computation of $B(d_{u,v}^{LCA}, \delta_u, \delta_v)$ only involves index with $O(1)$. The overall TRE complexity is $O(hT^2L^2 \log L)$. These costs are typically substantially smaller than the baseline Transformer operations $O(khT^2L^2d)$, considering $k \cdot h \cdot T$ is typically much larger than d, and $k \cdot d$ is also much larger than $\log L$ in practice. Our experiments show TrieRec incurs only a small computational overhead (less than 10%) as illustrated in Figure 4.

Hyperparameter-free. Notably, TrieRec does not introduce additional hyperparameters requiring extensive tuning. This makes TrieRec friendly for direct application.

Differences from other positional encodings: 1) Compared with conventional linear positional encodings [31, 22] that measure token distance along the sequence, TrieRec explicitly encodes the trie structure. Token relations are measured along the tree, enabling explicit injection of information such as LCA, tree distances, and item-path, which often yields better performance. 2) There exists another concurrent work (available only on arXiv) CoFiRec [37] that integrates a positional encoding signaling the layer of the token. Compared with this depth-based encoding, TrieRec offers richer structural information including the surrounding context and hierarchical relations, leading to typically superior performance. 3) Compared with tree-aware position encodings from other fields (e.g., HiT[44], HiRoPE[43] for instance-specific syntax trees in code generation tasks), TrieRec is tailored for the global shared trie in GR. These methods primarily focus on relative structural distances, as dynamic syntax trees lack fixed global semantics at specific depths. When applied to GR, they treat the semantic trie as a generic structure, failing to capture the absolute hierarchical granularity and prefix-based collaborative relatedness unique to our global trie, yielding suboptimal performance. Comprehensive discussions and comparisons with related literature are deferred to Section 5

Differences from DiscRec. There is another concurrent work DiscRec [16] (available only on arXiv) that explicitly encodes information from other tokens of the same item to enrich the target token. TrieRec differs from DiscRec in several aspects: (i) attention in DiscRec is a linear embedding combination of tokens and does not perform finer-grained dimension-level interactions. (ii) Beyond embedding-based augmentation, our TAE also considers related structural relations to guide attention aggregation beyond tokens on a single item.

4 Experiments

In this section, we conduct extensive experiments to answer the following research questions:

- **RQ1:** Does TrieRec outperform state-of-the-art GR methods?
- **RQ2:** How do two positional encodings (TAE and TRE) contribute to the overall performance?
- **RQ3:** Compared to the vanilla Transformer, does the proposed TRE effectively equip the model with the capability to capture prefix tree structures?
- **RQ4:** What is the computational overhead incurred by TrieRec?

4.1 Experimental setup

Datasets. We evaluate TrieRec on four widely-used real-world datasets: *Amazon-Beauty*, *Amazon-Toys*, and *Amazon-Food* are three datasets that contain user ratings on products from the corresponding categories on the Amazon e-commerce platform [19]; *MovieLens-1M* is a movie rating dataset

Table 1: Overall performance comparison. The best results are **bolded**. *Improv.* indicates the relative gain of TrieRec over the vanilla backbone. Note that N and R denote NDCG and Recall, respectively. The marker "*" indicates that the improvement is statistically significant (p-value < 0.05).

Model	Amazon Beauty		Amazon Toys		Amazon Food		MovieLens-1M	
	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10
Caser	0.0137	0.0227	0.0135	0.0234	0.0171	0.0343	0.0217	0.0482
GRU4Rec	0.0079	0.0163	0.0066	0.0131	0.0154	0.0312	0.0199	0.0417
SASRec	0.0263	0.0612	0.0320	0.0739	0.0309	0.0716	0.0641	0.1391
BERT4Rec	0.0269	0.0545	0.0265	0.0507	0.0334	0.0678	0.0597	0.1232
HSTU	0.0264	0.0622	0.0328	0.0754	0.0310	0.0662	0.0719	0.1553
CofiRec	0.0356	0.0642	0.0372	0.0662	0.0387	0.0721	0.0953	0.1768
TIGER	0.0395	0.0719	0.0418	0.0765	0.0424	0.0792	0.0912	0.1713
+ HiT	0.0396	0.0705	0.0398	0.0732	0.0405	0.0746	0.0905	0.1697
+ HiROPE	0.0385	0.0692	0.0383	0.0704	0.0431	0.0828	0.0896	0.1707
+ DiscRec	0.0395	0.0700	0.0430	0.0765	0.0429	0.0800	0.0969	0.1825
+ LayerPE	0.0383	0.0696	0.0406	0.0738	0.0422	0.0794	0.0861	0.1614
+ TrieRec	**0.0410**	**0.0723**	**0.0444**	**0.0766**	**0.0460**	**0.0843**	**0.1053**	**0.1929**
Improv.	*+3.80%**	*+0.56%*	*+6.22%**	*+0.13%*	*+8.49%**	*+6.44%**	*+15.46%**	*+12.61%**
CoST	0.0395	0.0703	0.0413	0.0756	0.0429	0.0811	0.0920	0.1730
+ HiT	0.0390	0.0703	0.0393	0.0727	0.0405	0.0746	0.0916	0.1715
+ HiROPE	0.0395	0.0715	0.0392	0.0722	0.0431	0.0828	0.0917	0.1717
+ DiscRec	0.0392	0.0698	0.0415	0.0752	0.0440	0.0828	0.0998	0.1848
+ LayerPE	0.0401	**0.0722**	0.0402	0.0730	0.0400	0.0760	0.0866	0.1618
+ TrieRec	**0.0409**	0.0720	**0.0461**	**0.0771**	**0.0461**	**0.0851**	**0.1049**	**0.1909**
Improv.	*+3.54%**	*+2.42%**	*+11.62%**	*+1.98%**	*+7.46%**	*+4.93%**	*+14.02%**	*+10.35%**
LETTER	0.0427	0.0765	0.0430	0.0781	0.0444	0.0827	0.0991	0.1858
+ HiT	0.0429	0.0774	0.0431	0.0777	0.0462	0.0849	0.0984	0.1843
+ HiROPE	0.0428	0.0773	0.0421	0.0774	0.0450	0.0832	0.0967	0.1801
+ DiscRec	0.0433	0.0772	0.0437	0.0799	0.0463	0.0856	0.0990	0.1841
+ LayerPE	0.0413	0.0743	0.0417	0.0754	0.0426	0.0802	0.0905	0.1748
+ TrieRec	**0.0464**	**0.0794**	**0.0477**	**0.0801**	**0.0469**	**0.0861**	**0.1129**	**0.2031**
Improv.	*+8.67%**	*+3.79%**	*+10.93%**	*+2.56%**	*+5.63%**	*+4.11%**	*+13.93%**	*+9.31%**

collected by the MovieLens platform [8]. For fair comparisons, we closely follow the setting of previous work [23, 34], and adopt a standard 5-core and retain only the most recent 20 interactions for each user. We adopt the leave-one-out strategy for data splitting, where the most recent item of each user is used for testing, the second most recent item for validation, and the remaining items for training. Detailed statistics of the datasets are summarized in Appendix A.

Baselines. We comprehensively evaluate TrieRec against three groups of baseline methods: (1) Traditional sequential models, including Caser [30], GRU4Rec [10], SASRec [11], and Bert4Rec [28]; (2) State-of-the-art generative models, including HSTU [41], TIGER [23], LETTER [34], CoST [51], and CofiRec [37]; and (3) Recent positional encoding methods, including HiT [44], HiRoPE [43], DiscRec [16], and LayerPE. To evaluate the backbone-robustness of our approach, we integrate TrieRec and the positional encoding baselines into multiple representative generative backbones (TIGER, CoST, LETTER). Detailed descriptions of all baseline models are provided in Appendix C.

Implementation Details. Full implementation details, including optimal hyperparameter search, tokenization settings, and architecture configurations, are elaborated in Appendix B.

4.2 Overall performance (RQ1)

The performance comparison of our TrieRec against standard baselines is shown in Table 1 (full results in Appendix D), while comparisons with the loss-based trie-aware method [26] are detailed in Appendix E (Table 6). Our proposed TrieRec consistently outperforms all baselines across different metrics and datasets, demonstrating the effectiveness of explicitly encoding the trie structure in the autoregressive modeling of generative recommendation.

Comparing with other recommendation models. Among all baseline methods, generative recommendation models consistently outperform traditional sequential recommendation models across all datasets. This performance gap highlights the advantages of the generative paradigm, which leverages hierarchical semantic IDs to capture both item semantics and collaborative patterns, enabling more accurate next-item prediction through structured sequence generation rather than direct ranking. More importantly, when integrated with different generative backbones (TIGER, CoST, LETTER),

TrieRec consistently achieves the best performance, demonstrating the effectiveness of our trie-aware positional encodings in enhancing generative recommendation.

Comparing with other positional encoding methods. When integrated with different generative backbones, TrieRec delivers significant and stable performance improvements indicating that TrieRec's effectiveness in identifying the most relevant items. Compared to other positional encoding methods, TrieRec shows superior performance for several reasons: (i) HiT and HiRoPE, originally designed for code generation tasks, lack the specific adaptations needed for generative recommendation scenarios; (ii) While LayerPE and DiscRec are tailored for generative recommendation, they rely on coarse-grained structural awareness — LayerPE only adds local labels to input embeddings, and DiscRec achieves item-level isolation through binary masks. In contrast, our method implements fine-grained trie structure reconstruction through TAE and TRE encodings, demonstrating that explicitly reconstructing the trie's topological structure is more effective than the implicit or discrete structural marking approaches adopted in previous work.

Comparing with trie-aware loss. We also compare TrieRec with SEATER [26], a recent method that implicitly incorporates trie information via a contrastive loss. TrieRec achieves significantly superior performance, demonstrating that our explicit and comprehensive structural encoding (via TAE and TRE) is much more effective than relying solely on auxiliary loss constraints. Detailed comparisons and discussions are provided in Appendix E.

4.3 Ablation study(RQ2)

Table 2 presents comprehensive ablation studies using LETTER as backbone, with results on all datasets and full evaluation metrics deferred to Appendix D. We evaluate the contribution of each component in TrieRec by removing or modifying key modules.

Table 2: Ablation study on two datasets (Backbone: LETTER).

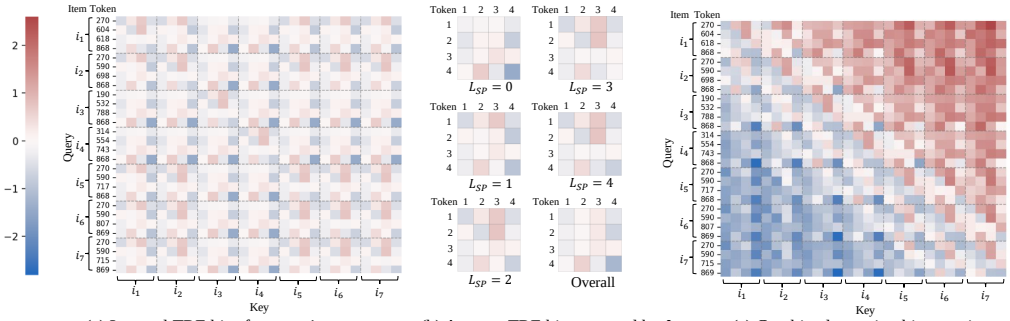
Variant	Amazon Beauty		MovieLens-1M	
	N@10	R@10	N@10	R@10
TrieRec	**0.0464**	**0.0794**	**0.1129**	**0.2031**
(1) w/o TAE	0.0442	0.0781	0.0946	0.1779
(2) w/o TRE	0.0444	0.0779	0.1061	0.1951
(3) TAE-Blocked	0.0450	0.0790	0.0971	0.1816
(4) TAE-Prefix	0.0425	0.0766	0.0956	0.1814
(5) TRE-w/o LCA	0.0453	0.0787	0.1112	0.2007
(6) TRE-w/o δ	0.0449	0.0786	0.1118	0.2017

Effects of TAE. Removing the entire TAE module (*w/o TAE*) leads to substantial performance degradation across all metrics, demonstrating that explicitly encoding the path from root to each item node is crucial for capturing the hierarchical semantics in the trie structure. When we restrict the MLP interaction to the same feature dimension across different depths (*TAE-Blocked*, where dimension j at depth k can only interact with dimension j at other depths), performance also decreases notably, indicating that cross-dimensional interaction is also important. Most strikingly, replacing TAE with a prefix sum-based integration mechanism (*TAE-Prefix*) results in the worst performance, even underperforming the original LETTER baseline, suggesting that simple cumulative integration cannot capture the complex hierarchical relationships in the trie structure, and our TAE is necessary for effective trie encoding.

Effects of TRE. Removing the TRE module (*w/o TRE*) while retaining only the standard 1D relative position bias causes significant performance drops, confirming that modeling trie-specific topological relationships is vital for generative recommendation. Further analysis reveals that both components of TRE are necessary: removing the LCA depth dimension while keeping only vertical distances (*TRE-w/o LCA*) degrades performance, indicating that the absolute hierarchical level of common ancestors provides important structural context beyond simple relative distances. Similarly, using only LCA depth without relative distance components (*TRE-w/o δ*) also hurts performance, demonstrating that knowing the exact path lengths between nodes is equally important. These results confirm that both the common prefix information and the precise topological distances work synergistically to provide comprehensive structural awareness in the trie.

4.4 Case study (RQ3)

To answer RQ3, we analyze the learned Topology-aware Relative Positional Encoding (TRE) on Amazon-Beauty dataset as shown in Figure 3. **(1)** The 4×4 TRE bias matrices exhibit consistent internal patterns. When items share prefixes, earlier tokens attend more to subsequent non-collision tokens (*e.g.,* large values at coordinates (1,2), (1,3), (2,3) when $L_{sp} \geq 1$ in Figure 3(b)), indicating TrieRec enriches ancestor representations using descendant information. Meanwhile, the consistent negative bias in the last columns of 4×4 matrices in Figure 3(a) and (b) shows the model effectively identifies and suppresses semantically meaningless collision identifiers. **(2)** Items sharing longer



(a) Learned TRE bias for a user's sequence. (b) Average TRE bias grouped by L_{SP}. (c) Combined attention bias matrix.

Figure 3: Case Study of the Topology-aware Relative Positional Encoding (TRE) on the Amazon-Beauty dataset. (a) The learned TRE bias $B(d_{u,v}^{LCA}, \delta_u, \delta_v)$ across a user's historical interaction sequence of 7 items (each tokenized into 4 tokens). (b) The average TRE bias for all item pairs in the dataset, grouped by their shared prefix length (L_{SP}). (c) The final attention bias matrix, illustrating the integration of TrieRec's topological structural bias with the backbone's temporal decay bias.

prefixes receive larger attention biases. In Figure 3(a), items sharing the prefix *270* mostly show positive biases, which further increase for i_5, i_6, i_7 sharing a longer prefix *270-590*. Figure 3(b) statistically confirms this universal mechanism: groups with $L_{sp} \geq 1$ consistently exhibit higher biases than the $L_{sp} = 0$ group. **(3)** A deeper Lowest Common Ancestor (LCA) on the trie leads to a larger attention bias. Observing the fourth tokens of item pairs (bottom-right of sub-matrices in Figure 3(b)), as the shared prefix length L_{sp} grows, their LCA depth d^{LCA} deepens, and TrieRec learns progressively larger TRE biases to capture this fine-grained structural proximity. **(4)** As shown in Figure 3(c), the resulting attention bias successfully integrates hierarchical topology with sequential dynamics. TrieRec superimposes the structural bias onto the backbone's foundational time-decaying sequential bias (shown in Appendix F), enabling fully topology-aware token generation.

4.5 Efficiency analysis (RQ4)

To evaluate the practical feasibility of TrieRec, we compare the computational efficiency of TrieRec against the baseline LETTER. Figure 4 illustrates the total training time and single inference latency on two representative datasets. The results indicate that while TrieRec introduces additional components to enhance recommendation quality, it remains within a comparable order of magnitude in terms of both training and inference costs. Although the sophisticated modeling in TrieRec leads to a slight increase in execution time, the growth is well-constrained and does not hinder the scalability of the model. Considering the significant performance improvements demonstrated in previous experiments, the marginal overhead is a justifiable trade-off for the substantial gains in recommendation accuracy.

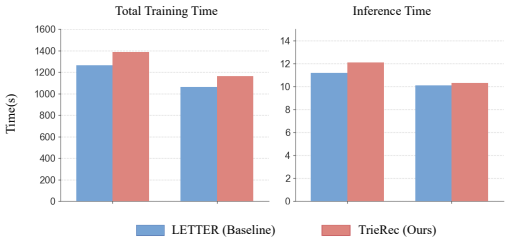


Figure 4: Efficiency comparison between TrieRec (backbone: LETTER) and baseline LETTER on Beauty and Toy datasets. The left panel shows the total training time, and the right panel shows the inference latency.

5 Related work

Sequential Recommendation. Sequential recommendation aims to predict the next item a user is likely to interact with based on their historical behavior sequences. Early neural models explored various deep learning architectures to identify temporal patterns and long-term dependencies within interaction histories, such as GRU4Rec [10], Caser [30], SASRec [11] and Bert4Rec [28]. Recently, there emerge diverse research paths to further enhance recommendation performance, such as leveraging various user actions to capture fine-grained preferences [38, 27] or incorporating additional item side information [48, 14, 47]

Generative Recommendation. Generative recommendation marks a significant departure from the traditional paradigm by representing items as discrete token sequences and formulating recommendation as a sequence-to-sequence decoding task. Generally, the generative recommendation paradigm

can be decomposed into two fundamental stages: item tokenization and the downstream generative recommendation model.

Recently, generative recommendation has emerged as a promising paradigm by reformulating the ranking task as a sequence-to-sequence problem. TIGER [23] serves as a pioneer in this field, utilizing RQ-VAE [13] for item tokenization and a T5-based architecture [22] for autoregressive generation. Following this, research has primarily diverged into two directions. The first line focuses on item indexing strategies: for instance, LETTER [34] enhances RQ-VAE by incorporating collaborative signals, while CoST [51] employs contrastive learning to derive more discriminative identifiers. Other concurrent works have also explored advanced hierarchical indexing methods or VAE-based enhancements [7, 42, 39]. The second direction attempts to optimize the downstream generative models themselves to better handle recommendation tasks [41, 36, 40, 17]. More recently, some studies have shifted toward capturing hierarchical information directly from item codes. For example, CoFiRec [37] introduces item-based position encoding, and DiscRec [16] integrates both collaborative and semantic information at the item level. Despite these advancements, most existing models still treat item codes as flattened 1D sequences during the generation process. This approach inevitably loses the critical topological information in the hierarchical item tree.

Tree-structure Modeling. In the recommendation field, tree structures are widely recognized for their coarse-to-fine properties and have been utilized for hierarchical indexing and retrieval. Classical approaches such as TDM [50] and JTM [49] organize large-scale item catalogs into tree-based indexes to enable efficient logarithmic-time retrieval. Recently, tree structures have further been exploited for layer-wise retrieval [45], transforming the recommendation task into a path-searching process on a taxonomy. In the generative recommendation paradigm, SEATER [26] attempts to model the tree hierarchy by introducing shared-prefix-aware and parent-child relationship-aware losses into the objective function. However, such optimization-level constraints merely provide external guidance during training, failing to fully and explicitly capture the trie structure within the model's computation. As discussed in SubSection 3.3, different from these works, we incorporate the trie structure into the Transformer as specialized position encodings. This design is more natively compatible with current generative tree recommendation frameworks and demonstrates superior effectiveness.

Modeling hierarchical structures is also a long-standing challenge in AI. Beyond the recommendation domain, various Tree-aware neural network have been proposed to capture the inductive bias of tree topologies. Early efforts in NLP, such as Tree-LSTM [29] and Tree-CNN [24], attempted to process hierarchical syntax trees through recursive or convolutional structures. With the dominance of Transformers, researchers have developed specialized encodings to inject structural information into self-attention. These include methods based on path information [25, 44, 20], such as HiT encode the trajectory from the root, and those integrating node positions and relative topological distances [43, 21], such as HiRoPE which utilizes 2D Rotary Positional Embedding (RoPE) to inject tree structure information. These methods were designed for code representation tasks, aiming to capture program-specific structural semantics, such as variable scoping within nested blocks or interaction patterns within function-level units. Specifically, these methods focus on syntax trees, which differ inherently from the prefix tree structure in recommendation. However, their methods are not suitable for the recommendation scenario where the entire item tokens together form a Trie. They struggle to capture key information such as the hierarchical semantics of identifiers and the structural correlations. Consequently, we also observe their performance are compromised as shown in Table 1.

6 Conclusion

In this paper, we identified a critical limitation in current generative recommendation (GR) pipelines: the conventional flattening of hierarchical item tokens discards valuable prefix-tree (trie) structures. To bridge this gap, we proposed TrieRec, a model-agnostic framework that injects structural inductive biases into Transformers via Trie-aware Absolute and Topology-aware Relative Positional Encodings. Extensive experiments demonstrate that integrating TrieRec into existing GR backbones yields significant performance gains (8.83% on average) with negligible computational overhead. Despite these improvements, TrieRec has certain limitations. Primarily, its effectiveness relies heavily on the quality of the offline item tokenization. Because the trie structure is pre-defined and fixed, any semantic noise introduced during hierarchical clustering (e.g., RQ-VAE) is explicitly encoded into the attention mechanism, potentially upper-bounding performance. Consequently, a natural direction for future work is to explore joint optimization strategies that dynamically refine the tree structure during autoregressive training. Additionally, we plan to extend TrieRec to multi-modal scenarios, where complex, multi-faceted item representations can further benefit from our structure-aware topologies.

Acknowledgments and Disclosure of Funding

This work is supported by the National Natural Science Foundation of China (62476244, 62372399), the Open Research Fund of the State Key Laboratory of Blockchain and Data Security, Zhejiang University (A2508), and Ant Group through the CCF-Ant Research Fund.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. Decoding matters: Addressing amplification bias and homogeneity issue in recommendations for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10540–10552, 2024.

[3] Michael A Bender and Martin Farach-Colton. The lca problem revisited. In *Latin American Symposium on Theoretical Informatics*, pages 88–94. Springer, 2000.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[6] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, René Vidal, Maheswaran Sathiamoorthy, Atoosa Kasirzadeh, and Silvia Milano. A review of modern recommender systems using generative models (gen-recsys). In *Proceedings of the 30th ACM SIGKDD conference on Knowledge Discovery and Data Mining*, pages 6448–6458, 2024.

[7] Dengzhao Fang, Jingtong Gao, Chengcheng Zhu, Yu Li, Xiangyu Zhao, and Yi Chang. Hid-vae: Interpretable generative recommendation via hierarchical and disentangled semantic ids. *arXiv preprint arXiv:2508.04618*, 2025.

[8] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

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

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[13] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532, 2022.

[14] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1258–1267, 2023.

[15] Xiaopeng Li, Bo Chen, Junda She, Shiteng Cao, You Wang, Qinlin Jia, Haiying He, Zheli Zhou, Zhao Liu, Ji Liu, et al. A survey of generative recommendation from a tri-decoupled perspective: Tokenization, architecture, and optimization. 2025.

[16] 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*, 2025.

[17] Zihan Liu, Yupeng Hou, and Julian McAuley. Multi-behavior generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1575–1585, 2024.

[18] 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*, pages 5915–5922, 2025.

[19] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908, 2013.

[20] Han Peng, Ge Li, Wenhan Wang, Yunfei Zhao, and Zhi Jin. Integrating tree path in transformer for code representation. *Advances in Neural Information Processing Systems*, 34:9343–9354, 2021.

[21] Han Peng, Ge Li, Yunfei Zhao, and Zhi Jin. Rethinking positional encoding in tree transformer for code representation. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 3204–3214, 2022.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[23] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[24] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-cnn: a hierarchical deep convolutional neural network for incremental learning. *Neural networks*, 121:148–160, 2020.

[25] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems*, 32, 2019.

[26] Zihua Si, Zhongxiang Sun, Jiale Chen, Guozhang Chen, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, Jun Xu, and Kun Gai. Generative retrieval with semantic tree-structured identifiers and contrastive learning. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 154–163, 2024.

[27] Jiajie Su, Chaochao Chen, Zibin Lin, Xi Li, Weiming Liu, and Xiaolin Zheng. Personalized behavior-aware transformer for multi-behavior sequential recommendation. In *Proceedings of the 31st ACM international conference on multimedia*, pages 6321–6331, 2023.

[28] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[29] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[30] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

[33] Bohao Wang, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. Msl: Not all tokens are what you need for tuning llm as a recommender. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1912–1922, 2025.

[34] 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*, pages 2400–2409, 2024.

[35] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023.

[36] Yifan Wang, Weinan Gan, Longtao Xiao, Jieming Zhu, Heng Chang, Haozhao Wang, Rui Zhang, Zhenhua Dong, Ruiming Tang, and Ruixuan Li. Act-with-think: Chunk auto-regressive modeling for generative recommendation. *arXiv preprint arXiv:2506.23643*, 2025.

[37] Tianxin Wei, Xuying Ning, Xuxing Chen, Ruizhong Qiu, Yupeng Hou, Yan Xie, Shuang Yang, Zhigang Hua, and Jingrui He. Cofirec: Coarse-to-fine tokenization for generative recommendation. *arXiv preprint arXiv:2511.22707*, 2025.

[38] Binquan Wu, Yu Cheng, Haitao Yuan, and Qianli Ma. When multi-behavior meets multi-interest: Multi-behavior sequential recommendation with multi-interest self-supervised learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 845–858. IEEE, 2024.

[39] Longtao Xiao, Haozhao Wang, Cheng Wang, Linfei Ji, Yifan Wang, Jieming Zhu, Zhenhua Dong, Rui Zhang, and Ruixuan Li. Unger: Generative recommendation with a unified code via semantic and collaborative integration. *ACM Transactions on Information Systems*, 44(2):1–31, 2025.

[40] Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, et al. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. *arXiv preprint arXiv:2503.02453*, 2025.

[41] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

[42] Penglong Zhai, Yifang Yuan, Fanyi Di, Jie Li, Yue Liu, Chen Li, Jie Huang, Sicong Wang, Yao Xu, and Xin Li. A simple contrastive framework of item tokenization for generative recommendation. *arXiv preprint arXiv:2506.16683*, 2025.

[43] Kechi Zhang, Ge Li, Huangzhao Zhang, and Zhi Jin. Hirope: Length extrapolation for code models using hierarchical position. *arXiv preprint arXiv:2403.19115*, 2024.

[44] Kechi Zhang, Zhuo Li, Zhi Jin, and Ge Li. Implant global and local hierarchy information to sequence based code representation models. In *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*, pages 157–168. IEEE, 2023.

[45] Wenlin Zhang, Chuhan Wu, Xiangyang Li, Yuhao Wang, Kuicai Dong, Yichao Wang, Xinyi Dai, Xiangyu Zhao, Huifeng Guo, and Ruiming Tang. Llmtreerec: Unleashing the power of large language models for cold-start recommendations. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 886–896, 2025.

[46] Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6889–6907, 2024.

[47] Hailun Zhou, Jiajie Xu, Qiaoming Zhu, and Chengfei Liu. Disentangled graph debiasing for next poi recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1779–1788, 2025.

[48] Xin Zhou, Hongyu Zhou, Yong Liu, Zhiwei Zeng, Chunyan Miao, Pengwei Wang, Yuan You, and Feijun Jiang. Bootstrap latent representations for multi-modal recommendation. In *Proceedings of the ACM web conference 2023*, pages 845–854, 2023.

[49] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. Joint optimization of tree-based index and deep model for recommender systems. *Advances in Neural Information Processing Systems*, 32, 2019.

[50] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1079–1088, 2018.

[51] Jieming Zhu, Mengqun Jin, Qijiong Liu, Zexuan Qiu, Zhenhua Dong, and Xiu Li. Cost: Contrastive quantization based semantic tokenization for generative recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 969–974, 2024.

[52] Jingwei Zhuo, Ziru Xu, Wei Dai, Han Zhu, Han Li, Jian Xu, and Kun Gai. Learning optimal tree models under beam search. In *International Conference on Machine Learning*, pages 11650–11659. PMLR, 2020.

A Dataset statistics

Table 3 summarizes the statistics of the four real-world datasets used in our experiments

Table 3: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions
Amazon Beauty	22,332	12,086	198,215
Amazon Toys	19,124	11,758	165,247
Amazon Food	14,631	8,686	150,809
MovieLens-1M	6,040	3,416	999,611

B Implementation details

For all generative models, we strictly follow the configuration of TIGER [23].

Item Tokenization Stage. We use Sentence-T5-base to extract textual features and employ RQ-VAE to generate semantic IDs consisting of three semantic tokens and one collision token. The RQ-VAE tokenizer utilizes codebooks with 256 embeddings of dimension 32 each, and the weighting coefficient β is set to 0.25. During the RQ-VAE training phase, we employ K-means initialization and train for 20,000 epochs with a learning rate of $1e^{-3}$; to ensure ID uniqueness, we select the epoch with the lowest collision rate before appending the fourth digit (collision token). It is worth noting that appending the collision digit does not affect the inherent prefix tree structure; it merely introduces an additional layer of leaf nodes to the trie. For CofiRec, we implement its hierarchical indexing with 4 codebooks, where the fourth codebook specifically utilizes SASRec embeddings as input without applying a collision-removal strategy.

Autoregressive Generation Stage. We adopt a 4-layer T5 architecture as the backbone. For the generative recommender's architecture, the model dimension is set to 128 with 6 attention heads and a hidden MLP size of 1024, applying a dropout rate of 0.1. The learning rate of the generative recommender is tuned within $\{1e^{-3}, 2e^{-3}, 3e^{-3}, 4e^{-3}, 5e^{-3}\}$ and weight decay within $\{0.1, 0.2, 0.3, 0.4\}$. All models use AdamW as the optimizer.

Application of TAE/TRE. TAE is applied only to encoder-side history, whose complete SIDs are observed, and not to decoder targets. TRE is applied in encoder self-attention and causal decoder self-attention. Under shifted-right decoding, the decoder uses only BOS and the available prefix, and the causal mask excludes future targets. BOS uses a dedicated TRE entry (index 0) as a virtual root.

Decoding and evaluation protocol. All generative methods use width-10 constrained beam search over valid trie children. Hypotheses are scored by summed token log-probabilities. The candidate space is the full catalog trie without sampling, although beam search is not exhaustive item-by-item scoring. Recall/NDCG use the generated top-K items without filtering prior interactions.

Repeated runs and statistical testing. Main results are averaged over five seeds. A paired, two-sided t-test compares seed-matched TrieRec and vanilla-backbone results ($p < 0.05$).

TAE/TRE configuration. One fixed TAE/TRE configuration is used throughout, requiring no additional hyperparameter search in our experiments.

Traditional Sequential Baselines. For all traditional sequential methods, we implement and search for the optimal hyperparameters strictly following the original paper. We have traversed and frequently expanded upon the entire hyperparameter space suggested by the authors to ensure all compared methods achieve optimal performance.

Hardware and Computational Cost. All experiments are conducted on a Linux server equipped with an Intel(R) Xeon(R) Gold 6530 CPU, 512GB of RAM, and an NVIDIA RTX 5090 GPU.

C Baselines

1) Traditional sequential recommendation models: These models treat items as atomic indices and focus on learning ranking functions.

- **Caser**(WSDM'2018) [30]: the CNN-based method that applies convolutional filters to capture multi-level sequential patterns.
- **GRU4Rec**(CIKM'2016) [10]: the RNN-based method that leverages GRU to model the temporal dynamics of user interests.
- **SASRec**(ICDM'2018) [11]: the self-attentive model that employs a causal mask Attention architecture to capture user interests.
- **Bert4Rec:**(CIKM'2019)[28] the bidirectional attention-based model that utilizes BERT to model user preference.

2) Generative recommendation models: This group includes state-of-the-art generative models including:

- **HSTU**(ICML'2024) [41]: the high-performance temporal unit that utilizes an enhanced attention mechanism.
- **TIGER**(NeurIPS'2023) [23]: the classic generative model that uses RQ-VAE to quantize item text embeddings into semantic IDs and performs recommendation by generating these IDs.
- **LETTER**(CIKM'2024) [34]: the state-of-the-art method based on TIGER that incorporates collaborative signals and diversity regularization during the RQ-VAE training stage.
- **CoST**(RecSys'2024) [51]: the state-of-the-art method based on TIGER that replaces the reconstruction loss of RQ-VAE with a contrastive loss.
- **CofiRec**(ArXiv'2025) [37]: the state-of-the-art method that adopts a coarse-to-fine tokenization to generate semantic IDs, and introduces item-based encoding and a hierarchical loss.

To evaluate the backbone-robustness of TrieRec, we evaluate TrieRec with three backbones: TIGER [23], LETTER [34], and CoST [51]. The selection is because they represent three distinct hierarchical quantization paradigms. Besides, they are either classical or represent SOTA methods.

3) Positional encoding methods: We also compare TrieRec with recent positional encoding strategies on multiple backbones:

- **HiT**(ICPC'2023) [44] and **HiRoPE**(ACL'2024) [43]: Two tree-aware positional encodings proposed in other domains inject structural information through path embeddings or extended 2D RoPE.
- **DiscRec**(ArXiv'2025) [16]: a strategy that explicitly encodes information from other intra-item tokens to enrich the target token. While DiscRec involves other not positional encoding modules, we also include them for strict comparisons.
- **LayerPE**: CofiRec [37] also introduces a positional encoding that signals the token layer and the item position within the sequence. We also evaluate the effect of this encoding.

D Complete experimental results

Due to space limitations in the main text, Table 1 and Table 2 only report the results evaluated by NDCG@10 and Recall@10. In this section, we provide the complete experimental results evaluated by additional @5 metrics.

Specifically, Table 4 presents the full overall performance comparison across all baselines, and Table 5 details the comprehensive ablation study results. As shown in both tables, our proposed TrieRec consistently outperforms all baseline models and its variants across both @5 and @10 metrics. These complete results perfectly align with the conclusions discussed in Section 4.2 and Section 4.3 of the main text, further validating the robustness and effectiveness of our proposed structural encodings.

E Comparison with loss-based trie-aware methods

SEATER is currently the only trie-aware GR method that attempts to incorporate trie information. However, it does so implicitly through a contrastive loss objective. To evaluate the effectiveness of our explicit structural encoding against this implicit optimization strategy, we compare our generative recommender with SEATER. For a fair comparison, we implement TrieRec using the SEATER backbone (note that SEATER's architecture is relatively complex and difficult to integrate with other standard backbones).

Table 4: Overall performance comparison. The best results are **bolded**. Improv. indicates the relative gain of TrieRec over the vanilla backbone. Note that N and R denote NDCG and Recall, respectively. The marker "*" indicates that the improvement is statistically significant (p-value < 0.05)

Model	Amazon Beauty				Amazon Toys				Amazon Food				MovieLens-1M			
	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10
Caser	0.0116	0.0137	0.0163	0.0227	0.0112	0.0135	0.0161	0.0234	0.0121	0.0171	0.0189	0.0343	0.0134	0.0217	0.0217	0.0482
GRU4Rec	0.0055	0.0079	0.0088	0.0163	0.0045	0.0066	0.0066	0.0131	0.0111	0.0154	0.0178	0.0312	0.0128	0.0199	0.0202	0.0417
SASRec	0.0177	0.0263	0.0346	0.0612	0.0229	0.0320	0.0458	0.0739	0.0210	0.0309	0.0411	0.0716	0.0440	0.0641	0.0765	0.1391
BERT4Rec	0.0196	0.0269	0.0317	0.0545	0.0207	0.0265	0.0326	0.0507	0.0245	0.0334	0.0399	0.0678	0.0420	0.0597	0.0679	0.1232
HSTU	0.0176	0.0264	0.0347	0.0622	0.0238	0.0328	0.0476	0.0754	0.0231	0.0310	0.0416	0.0662	0.0511	0.0719	0.0902	0.1553
CofiRec	0.0287	0.0356	0.0430	0.0642	0.0303	0.0372	0.0449	0.0662	0.0305	0.0387	0.0467	0.0721	0.0760	0.0953	0.1129	0.1768
TIGER	0.0318	0.0395	0.0478	0.0719	0.0332	0.0418	0.0501	0.0765	0.0337	0.0424	0.0519	0.0792	0.0713	0.0912	0.1096	0.1713
+ HiT	0.0320	0.0396	0.0467	0.0705	0.0320	0.0398	0.0488	0.0732	0.0327	0.0405	0.0504	0.0746	0.0704	0.0905	0.1071	0.1697
+ HiROPE	0.0307	0.0385	0.0450	0.0692	0.0302	0.0383	0.0455	0.0704	0.0330	0.0431	0.0514	0.0828	0.0693	0.0896	0.1076	0.1707
+ DiscRec	0.0322	0.0395	0.0475	0.0700	0.0351	0.0430	0.0521	0.0765	0.0336	0.0429	0.0512	0.0800	0.0760	0.0969	0.1174	0.1825
+ LayerPE	0.0307	0.0383	0.0458	0.0696	0.0326	0.0406	0.0487	0.0738	0.0332	0.0422	0.0517	0.0794	0.0763	0.0861	0.1030	0.1614
+ TrieRec	**0.0335**	**0.0410**	**0.0491**	**0.0723**	**0.0368**	**0.0444**	**0.0530**	**0.0766**	**0.0368**	**0.0460**	**0.0558**	**0.0843**	**0.0841**	**0.1053**	**0.1272**	**0.1929**
Improv.	*+5.35%**	*+3.80%**	*+2.72%**	*+0.56%*	*+10.84%**	*+6.22%**	*+5.79%**	*+0.13%*	*+9.20%**	*+8.49%**	*+7.51%**	*+6.44%**	*+17.91%**	*+15.46%**	*+16.06%**	*+12.61%**
CoST	0.0322	0.0395	0.0475	0.0703	0.0331	0.0413	0.0503	0.0756	0.0335	0.0429	0.0518	0.0811	0.0719	0.0920	0.1104	0.1730
+ HiT	0.0312	0.0390	0.0461	0.0703	0.0312	0.0393	0.0475	0.0727	0.0327	0.0405	0.0504	0.0746	0.0723	0.0916	0.1118	0.1715
+ HiROPE	0.0315	0.0395	0.0468	0.0715	0.0316	0.0392	0.0486	0.0722	0.0330	0.0431	0.0514	0.0828	0.0725	0.0917	0.1119	0.1717
+ DiscRec	0.0316	0.0392	0.0466	0.0698	0.0332	0.0415	0.0495	0.0752	0.0344	0.0440	0.0529	0.0828	0.0785	0.0998	0.1186	0.1848
+ LayerPE	0.0325	0.0401	0.0488	**0.0722**	0.0326	0.0402	0.0493	0.0730	0.0308	0.0400	0.0474	0.0760	0.0682	0.0866	0.1048	0.1618
+ TrieRec	**0.0337**	**0.0409**	**0.0494**	0.0720	**0.0391**	**0.0461**	**0.0555**	**0.0771**	**0.0367**	**0.0461**	**0.0562**	**0.0851**	**0.0847**	**0.1049**	**0.1283**	**0.1909**
Improv.	*+4.66%**	*+3.54%**	*+4.00%**	*+2.42%**	*+18.13%**	*+11.62%**	*+10.34%**	*+1.98%**	*+9.55%**	*+7.46%**	*+8.49%**	*+4.93%**	*+17.80%**	*+14.02%**	*+16.21%**	*+10.35%**
LETTER	0.0345	0.0427	0.0512	0.0765	0.0347	0.0430	0.0523	0.0781	0.0341	0.0444	0.0526	0.0827	0.0779	0.0991	0.1200	0.1858
+ HiT	0.0345	0.0429	0.0514	0.0774	0.0349	0.0431	0.0519	0.0777	0.0365	0.0462	0.0556	0.0849	0.0780	0.0984	0.1209	0.1843
+ HiROPE	0.0345	0.0428	0.0516	0.0773	0.0334	0.0421	0.0501	0.0774	0.0353	0.0450	0.0532	0.0832	0.0764	0.0967	0.1169	0.1801
+ DiscRec	0.0352	0.0433	0.0519	0.0772	0.0353	0.0437	0.0538	0.0799	0.0366	0.0463	0.0557	0.0856	0.0784	0.0990	0.1199	0.1841
+ LayerPE	0.0333	0.0413	0.0495	0.0743	0.0335	0.0417	0.0500	0.0754	0.0329	0.0426	0.0502	0.0802	0.0690	0.0905	0.1080	0.1748
+ TrieRec	**0.0385**	**0.0464**	**0.0548**	**0.0794**	**0.0403**	**0.0477**	**0.0569**	**0.0801**	**0.0373**	**0.0469**	**0.0561**	**0.0861**	**0.0914**	**0.1129**	**0.1363**	**0.2031**
Improv.	*+11.59%**	*+8.67%**	*+7.03%**	*+3.79%**	*+16.14%**	*+10.93%**	*+8.80%**	*+2.56%**	*+9.38%**	*+5.63%**	*+6.65%**	*+4.11%**	*+17.33%**	*+13.93%**	*+13.58%**	*+9.31%**

Table 5: Ablation study on four datasets (Backbone: LETTER).

Variant	Amazon Beauty				Amazon Toy				Amazon Food				MovieLens-1M			
	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10	N@5	N@10	R@5	R@10
TrieRec	**0.0385**	**0.0464**	**0.0548**	**0.0794**	**0.0403**	**0.0477**	**0.0569**	**0.0801**	**0.0373**	**0.0469**	**0.0561**	**0.0861**	**0.0914**	**0.1129**	**0.1363**	**0.2031**
(1) w/o TAE	0.0360	0.0442	0.0528	0.0781	0.0383	0.0464	0.0562	0.0800	0.0361	0.0461	0.0539	0.0847	0.0740	0.0946	0.1138	0.1779
(2) w/o TRE	0.0363	0.0444	0.0527	0.0779	0.0381	0.0459	0.0549	0.0791	0.0348	0.0438	0.0524	0.0803	0.0847	0.1061	0.1287	0.1951
(3) TAE-Blocked	0.0362	0.0450	0.0531	0.0790	0.0375	0.0454	0.0559	0.0799	0.0346	0.0441	0.0531	0.0826	0.0762	0.0971	0.1166	0.1816
(4) TAE-Prefix	0.0343	0.0425	0.0513	0.0766	0.0339	0.0418	0.0515	0.0760	0.0351	0.0444	0.0534	0.0824	0.0734	0.0956	0.1127	0.1814
(5) TRE-w/o LCA	0.0374	0.0453	0.0540	0.0787	0.0385	0.0465	0.0554	0.0799	0.0355	0.0455	0.0527	0.0837	0.0901	0.1112	0.1349	0.2007
(6) TRE-w/o δ	0.0371	0.0449	0.0543	0.0786	0.0397	0.0473	0.0566	0.0795	0.0372	0.0468	0.0551	0.0850	0.0900	0.1118	0.1343	0.2017

As shown in Table 6, TrieRec achieves significantly better performance than SEATER across all datasets. This indicates that simply incorporating trie information through auxiliary loss objectives is insufficient for effective trie-aware modeling. Furthermore, SEATER focuses solely on local parent-child relationships while neglecting global positional information and higher-order structural relationships. In contrast, TrieRec achieves a more comprehensive trie encoding strategy through TAE and TRE, explicitly injecting topological structures into the self-attention mechanism, thereby yielding superior performance.

Table 6: Performance comparison between TrieRec(Backbone: LETTER) and SEATER.

Dataset	Model	N@5	N@10	R@5	R@10
Beauty	SEATER	0.0334	0.0413	0.0508	0.0572
	TrieRec	**0.0385**	**0.0464**	**0.0548**	**0.0794**
Toy	SEATER	0.0351	0.0432	0.0523	0.0778
	TrieRec	**0.0403**	**0.0477**	**0.0569**	**0.0801**
Food	SEATER	0.0342	0.0440	0.0516	0.0822
	TrieRec	**0.0373**	**0.0469**	**0.0561**	**0.0861**
ML-1M	SEATER	0.0726	0.0948	0.1123	0.1813
	TrieRec	**0.0914**	**0.1129**	**0.1363**	**0.2031**

F Visualization of vanilla temporal bias

To better understand the structural improvements introduced by TrieRec, we provide the visualization of the attention bias from the vanilla T5 backbone (without our proposed TRE module) in Figure 5. This visualization uses the exact same user interaction sequence from the Amazon-Beauty dataset as analyzed in Section 4.4.

As illustrated in Figure 5, the vanilla relative positional encoding strictly adheres to a 1D temporal decay pattern. The bias values smoothly decrease as the distance between the query and key tokens increases along the flattened sequence. While this effectively captures the sequential dynamics (i.e., recent historical items receive higher attention), it is completely structurally agnostic. It fails to bind tokens belonging to the same item, cannot differentiate semantic relatedness based on shared prefixes, and treats non-semantic collision tokens identically to root tokens. This baseline visualization further underscores the necessity and effectiveness of the topology-aware patterns demonstrated in the main text.

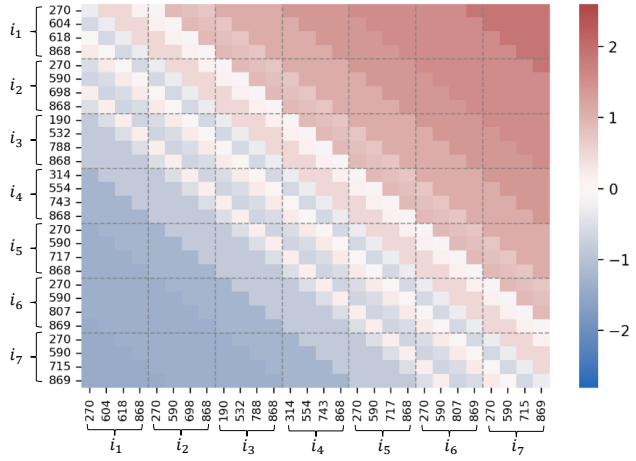


Figure 5: The relative attention bias of the vanilla T5 backbone. It exhibits a simple 1D sequential distance decay, lacking any capability to capture the hierarchical trie structure among semantic IDs.

G Additional controls for trie-aware modeling

We conduct the following controls using LETTER under the same setup.

G.1 Parameter-matched topology controls

For **TAE-RandomPath + TRE**, we apply a fixed shuffled item-to-path mapping only when computing TAE; the actual input/target SIDs, TRE relations, architecture, and parameter count remain unchanged. For **TAE + TRE-RandomLCA**, we replace each token pair's correct relation-table indices with fixed random valid indices. TAE and all other settings remain unchanged. Both variants are trained from scratch under the same configuration.

In Table 7, correctly aligned TrieRec outperforms both parameter-matched controls in all 12 dataset-metric comparisons, indicating that the gains are not explained solely by additional parameters.

G.2 Embedding-proximity controls

PrefixMean + TRE replaces TAE with the mean of prefix SID token embeddings, added directly to the token embedding, while retaining TRE. **TAE + PrefixCos** retains TAE but replaces TRE with the cosine similarity between the two tokens' prefix-mean embeddings, added directly to the attention logit. Both replacements are computed from the generator's SID token embeddings and introduce no additional parameters. We also evaluate the joint **PrefixMean + PrefixCos** replacement.

The simple cosine bias recovers part of the improvement, while both individual replacements and their combination remain below TrieRec. The joint replacement is not consistently better than LETTER. These results suggest that embedding proximity contributes to performance, while learned trie-aware structure provides a complementary benefit.

G.3 Pairwise topology beyond token layers

Both variants in Table 8 remove TAE. **Full TRE** corresponds to *w/o TAE* in Table 5. The **layer-only bias** uses only token-layer indices, removing both LCA depth and the distances to the LCA. Full TRE performs better on both metrics and datasets, indicating that layer information alone is insufficient.

Table 7: Parameter-matched topology controls and embedding-proximity controls with LETTER. N and R denote NDCG and Recall, respectively.

Method	Beauty		Toys		ML-1M	
	N@10	R@10	N@10	R@10	N@10	R@10
LETTER	.0427	.0765	.0430	.0781	.0991	.1858
TAE-RandomPath + TRE	.0450	.0775	.0454	.0760	.1080	.1964
TAE + TRE-RandomLCA	.0442	.0784	.0432	.0771	.1076	.1987
PrefixMean + TRE	.0438	.0773	.0445	.0780	.0910	.1760
TAE + PrefixCos	.0445	.0779	.0452	.0795	.1089	.1969
PrefixMean + PrefixCos	.0426	.0772	.0438	.0777	.0957	.1768
TAE + TRE (TrieRec)	**.0464**	**.0794**	**.0477**	**.0801**	**.1129**	**.2031**

Table 8: Full TRE versus a layer-only bias with LETTER. Both variants remove TAE.

Variant	Beauty		Toys	
	N@10	R@10	N@10	R@10
Full TRE	**.0442**	**.0781**	**.0464**	**.0800**
Layer-only bias	.0419	.0759	.0401	.0739

H Robustness to SID perturbations

We conduct a controlled Beauty experiment that replaces one SID layer (**Random-l**) or all four layers (**RandomAll**) with fixed random valid codes. LETTER and TrieRec are retrained from scratch on the same corrupted SIDs. The assignments are not constrained to remain unique and may introduce collisions. When a generated SID corresponds to multiple items, we select one fixed item for evaluation.

Table 9: Robustness to fixed SID perturbations on Beauty. Both methods use the same SID assignments within each condition.

SID condition	LETTER		TrieRec	
	N@10	R@10	N@10	R@10
Clean	.0427	.0765	**.0464**	**.0794**
RandomAll	.0063	.0127	**.0206**	**.0390**
Random-1	.0376	.0645	**.0386**	**.0672**
Random-2	.0345	.0603	**.0400**	**.0681**
Random-3	.0426	.0749	**.0444**	**.0759**
Random-4	.0425	.0771	**.0452**	**.0783**

TrieRec outperforms LETTER in all five perturbation settings on both metrics. Randomizing all layers leads to a marked performance decrease. These results show that TrieRec continues to provide benefits under SID noise, while its absolute performance remains dependent on hierarchy quality.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction (Section 1) clearly state the claims regarding the effectiveness of explicitly encoding trie structures, which are directly supported by the extensive experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the computational overhead and efficiency trade-offs in Section 4.5, and mentions potential areas for future improvement in the Conclusion (Section 6)

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes novel architectural components (positional encodings) and is primarily empirical. It does not introduce formal mathematical theorems that require proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Detailed experimental setups, including dataset statistics, data splitting strategies, and exhaustive implementation details (e.g., hyperparameter search spaces, tokenization settings) are provided in Section 4.1 and Appendices A and B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide an anonymized URL in the abstract containing our source code and partial data for reproducibility. The complete original datasets (Amazon and MovieLens) are widely used public benchmarks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Data splitting is explained in Section 4.1, and Appendix B provides a comprehensive breakdown of the training details, including model dimensions, dropout rates, learning rates, and the optimizer.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper conducts statistical significance tests, and statistically significant improvements (p-value < 0.05) are explicitly marked with an asterisk (*) in the main results (Table 1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detail the hardware configurations in Appendix B

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research develops foundational recommendation algorithms using standard, anonymized public benchmark datasets, posing no foreseeable ethical violations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper strictly focuses on foundational and methodological improvements to generative recommendation architectures and has no societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The proposed method is applied to standard recommender system benchmarks (e-commerce and movies), which do not pose a high risk for malicious misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original creators of the widely-used public datasets (Amazon Review datasets and MovieLens) are properly cited in Section 4.1 and the References section.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper proposes a new algorithmic method (TrieRec) but does not introduce or release new datasets or similar data assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research is entirely based on offline, publicly available historical interaction datasets and does not involve human subjects or crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Since the study does not involve human subjects or data collection from participants, IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology involves designing structural positional encodings. While it uses a T5 backbone for sequence generation, it does not rely on LLMs as an unconventional or original component of the research method itself.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.