CoFiRec: Coarse-to-Fine Tokenization for Generative Recommendation

Tianxin Wei[1,⋆,○], **Xuying Ning**[1,⋆], **Xuxing Chen**[2,†], **Ruizhong Qiu**[1], **Yupeng Hou**[3], **Yan Xie**[2], **Shuang Yang**[2], **Zhigang Hua**[2], **Jingrui He**[1]

[1]University of Illinois Urbana-Champaign, [2]Meta, [3]University of California San Diego
[⋆]Equal Contribution, [†]Core Contributors, [○]Work done at Meta

In web environments, user preferences are often refined progressively as users move from browsing broad categories to exploring specific items. However, existing generative recommenders overlook this natural refinement process. Generative recommendation formulates next-item prediction as autoregressive generation over tokenized user histories, where each item is represented as a sequence of discrete tokens. Prior models typically fuse heterogeneous attributes such as ID, category, title, and description into a single embedding before quantization, which flattens the inherent semantic hierarchy of items and fails to capture the gradual evolution of user intent during web interactions. To address this limitation, we propose **CoFiRec**, a novel generative recommendation framework that explicitly incorporates the **Co**arse-to-**Fi**ne nature of item semantics into the tokenization process. Instead of compressing all attributes into a single latent space, CoFiRec decomposes item information into multiple semantic levels, ranging from high-level categories to detailed descriptions and collaborative filtering signals. Based on this design, we introduce the **CoFiRec Tokenizer**, which tokenizes each level independently while preserving structural order. This design naturally reflects the way users refine their preferences, enabling more structured generation. During autoregressive decoding, the language model is instructed to generate item tokens from coarse to fine, progressively modeling user intent from general interests to specific item-level interests. Experiments across multiple public benchmarks and backbones demonstrate that CoFiRec outperforms existing methods, offering a new perspective on structured token design for generative recommendation. Theoretically, we prove that structured tokenization leads to lower dissimilarity between generated and ground truth items, supporting its effectiveness in generative recommendation. Our code is available at https://github.com/YennNing/CoFiRec.

Date: December 1, 2025
Correspondence: {xuyingn2,twei10,jingrui}@illinois.edu

Meta

1 Introduction

Generative recommendation (GR) (Rajput et al., 2023; Hua et al., 2023; Xu et al., 2023; Tan et al., 2024; Yue et al., 2023; Yang et al., 2023) has recently emerged as a promising paradigm for sequential recommendation (Sun et al., 2019; Kang and McAuley, 2018; Li et al., 2023), where user interacted items are tokenized into discrete sequences and modeled autoregressively. This formulation enables GR models to leverage powerful sequence modeling architectures such as Transformers (Vaswani et al., 2017), while operating over a compact vocabulary of discrete tokens that improves generalization for new items (Tan et al., 2024; Rajput et al., 2023), inference efficiency large-scale recommendation, and recommendation quality (Rajput et al., 2023; Lin et al., 2024a). These models have demonstrated strong empirical performance across various domains and tasks.

While GR methods have made substantial progress, they often overlook a key characteristic of real-world recommender systems: the presence of rich, multi-scale semantic structure in item metadata. Existing GR approaches typically flatten all item attributes into a single embedding and encode them using vector quantization, especially residual quantization-based methods like RQ-VAE (Lee et al., 2022). Although residual quantization introduces an implicit hierarchy in the latent space by encoding residual errors, it fails to capture semantic granularity and interpretability (Rajput et al., 2023; Tan et al., 2024; Wang et al., 2024; Yue et al., 2023). SemID (Hua et al., 2023) provides explicit taxonomy-based semantic indexing, but its design

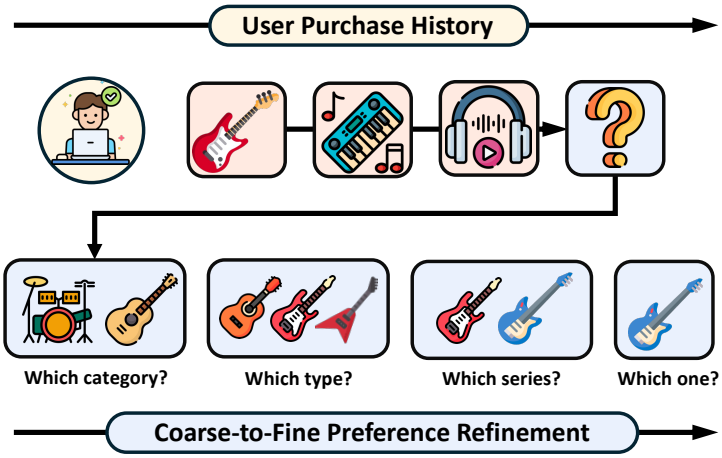


Figure 1 Users on the web refine their preferences progressively from category to type to specific series during item exploration.

is manually defined and non-learnable, limiting adaptability and discriminative power across datasets.

In contrast, we argue that explicitly modeling the coarse-to-fine nature of item semantics is essential to better align with how users form preferences and navigate item spaces. As illustrated in Figure 1, users typically start from broad categories (e.g., musical instruments), then narrow down to mid-level types (e.g., electric guitars), specific series (e.g., Yamaha Pacifica), and the finally individual item. This multi-level organization is not merely a detail of item annotation, it reflects how users naturally perceive, search for, and refine their preferences over time. Furthermore, collaborative signals (Tang and Wang, 2018; Sarwar et al., 2001; He et al., 2020; Zhu et al., 2023; Zhang et al., 2023b; Zheng et al., 2024) such as item co-interaction patterns (Sarwar et al., 2001; He et al., 2020; Sarwar et al., 2001) provide an additional layer of fine-grained, behavior-driven semantics that is vital for generative recommendations.

To address these limitations, we propose **CoFiRec**, a generative recommendation framework that explicitly incorporates the **Co**arse-to-**Fi**ne nature of item semantics into both tokenization and generation. As illustrated in Figure 2, CoFiRec decomposes item metadata and constructs into multiple semantic levels, ranging from high-level categories to fine-grained descriptions, and appends a final item *collaborative filtering (CF) signals* to encode fine-grained behavioral semantics. Unlike prior methods that collapse all item features into a single embedding, CoFiRec retains the hierarchical organization by tokenizing each semantic level independently using coarse-to-fine vector quantization modules in the CoFiRec Tokenizer. This produces a structured token sequence that mirrors the semantic hierarchy of item representation. During generation, the downstream language model is trained to autoregressively decode these tokens in a coarse-to-fine order, reflecting the natural refinement process of user intent. To further enhance generation fidelity and structural alignment, we introduce trainable embeddings that encode both the item identity and the semantic role of each token, enabling the language model to distinguish between levels and maintain consistent semantic conditioning throughout the sequence.

We evaluate our approach on multiple benchmark datasets using LLMs of various scales, including T5 (Ni et al., 2021), LLaMA3.2-1B, and LLaMA3.2-3B (Grattafiori et al., 2024). Experimental results show that CoFiRec outperforms competitive traditional and generative baselines by an average of 7.27% across all datasets. Our main contributions are:

- **Theory-motivated reformulation.** We first theoretically show that hierarchical decoding improves the relevance of recommended items. Motivated by our theory, we propose CoFiRec, a novel coarse-to-fine generative recommendation framework to leverage the semantic hierarchy of items in decoding.
- **Coarse-to-fine quantization.** We design the CoFiRec Tokenizer, which uses coarse-to-fine quantization to tokenize multi-granular semantic attributes and collaborative filtering signals into a unified coarse-to-fine token sequence, supporting structured and interpretable generation.
- **Strong performance.** We empirically demonstrate that coarse-to-fine tokenization improves recommendation

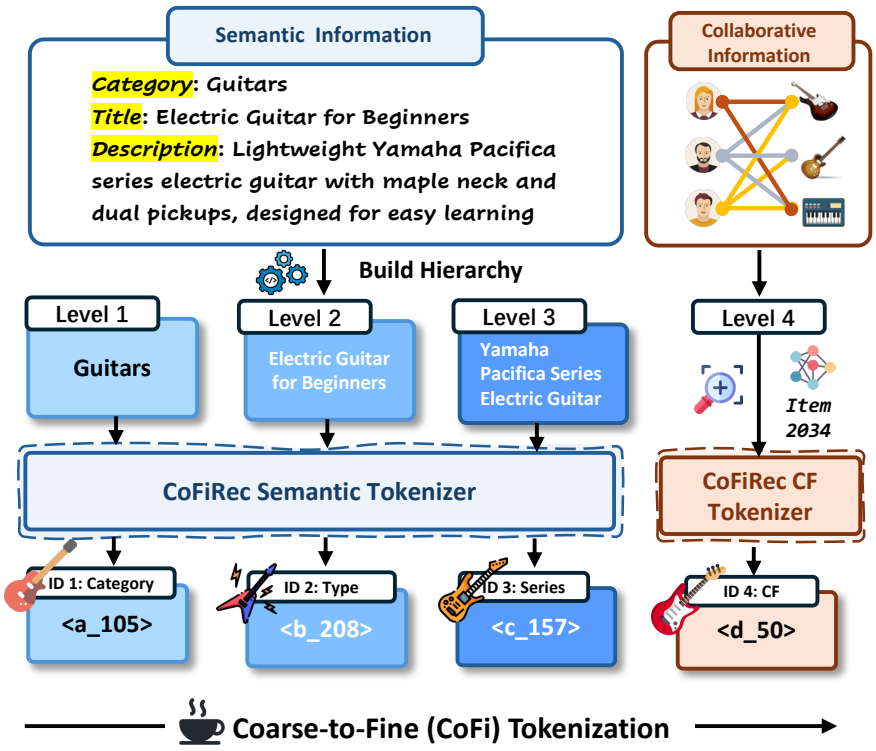


Figure 2 In CoFiRec tokenization, item metadata are hierarchically organized (category, title, description) and each level is tokenized independently, with CF signals serve as the most fine-grained tokens.

accuracy by aligning with how users naturally refine their intents.

2 Preliminaries: Generative Recommendation via Discrete Tokens

Problem Setup. Let $\mathcal{U}$ and $\mathcal{I}$ denote the user and item sets respectively. For each user $u \in \mathcal{U}$, we observe a historical interaction sequence $\mathcal{H}_u = [i_1, i_2, ..., i_T]$, where each $i_t \in \mathcal{I}$. The goal is to predict the next item i_{T+1} that the user is likely to interact with. Generative recommendation models treat this as a sequence modeling task and generate i_{T+1} autoregressively.

Tokenization. To apply language models, each item i must be represented as one token or a sequence of discrete tokens. Prior work uses approaches such as residual quantization (Rajput et al., 2023) or textual encoding (Tan et al., 2024) to map item embeddings into discrete token sequences.

Let $\boldsymbol{e}_i \in \mathbb{R}^d$ denote the dense embedding of item i, typically obtained by encoding its flattened multi-level attributes using a pretrained language model. A vector quantization model $\text{VQ}(\boldsymbol{e}_i)$ maps $\boldsymbol{e}_i$ into a sequence of K discrete tokens $[z_i^{(1)}, z_i^{(2)}, ..., z_i^{(K)}]$, where $z_i^{(k)} \in \mathcal{V}^k$ and $\mathcal{V}^k$ is the codebook at level k. The generation process is modeled as:

$$\mathbb{P}(z^{(K)}, ..., z^{(1)}, \mathcal{H}_u) = \prod_{k=1}^{K} \mathbb{P}(z^{(k)} \mid z^{(<k)}, \mathcal{H}_u), \tag{1}$$

where each token $z^{(k)}$ is generated conditioned on the previous token and the user's interaction history $\mathcal{H}_u$.

Limitation. Existing methods flatten heterogeneous semantic features into the embedding before quantization, discarding the natural hierarchical organization of item attributes. As a result, the generated token sequence lacks interpretability and fails to reflect the coarse-to-fine nature of user preference refinement. Moreover, the standard token-level autoregressive modeling treats each token independently, overlooking the structure across semantic levels.

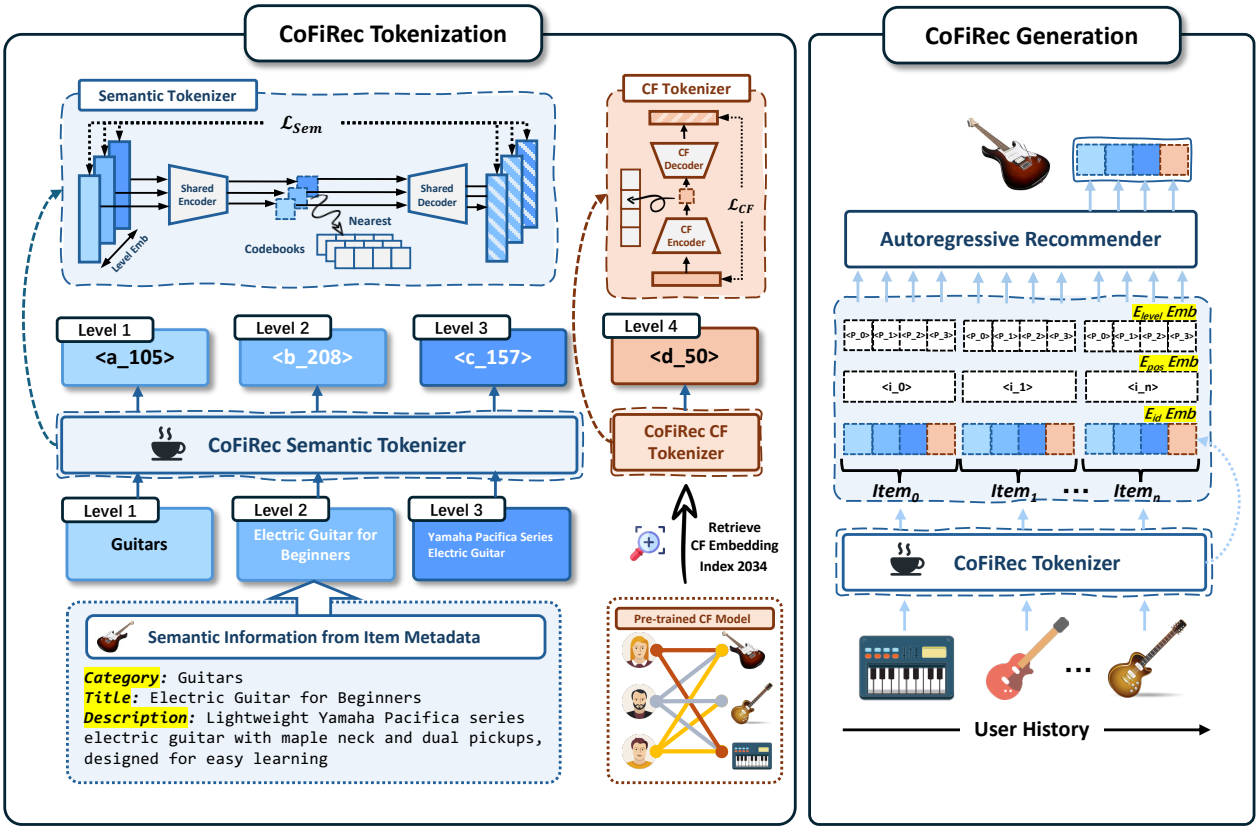


Figure 3 The overall framework of CoFiRec, which models multi-level tokenization for coarse-to-fine generative recommendation. **Left:** The CoFiRec Tokenizer decomposes item metadata into hierarchical semantic levels (e.g., category, title, description) and encodes each level using coarse-to-fine VQ modules. Collaborative filtering (CF) signals are tokenized independently from a pre-trained CF embedding via a dedicated CF Tokenizer. **Right:** During generation, an autoregressive language model generates tokens from coarse to fine, conditioned on user history. We inject semantic-level and positional embeddings to help the model identify the role of each token in the structured generation.

3 Method

In this section, we present the complete methodology of CoFiRec, our proposed coarse-to-fine generative recommendation framework. We begin by reviewing the generative recommendation paradigm and review the flaws of existing framework, which frames recommendation as an autoregressive generation task over tokenized item sequences. Next, we introduce multi-level tokenization strategy, detailing the construction of semantic hierarchies and the design of the CoFiRec Tokenizer. Finally, we reformulate the decoding process to align with the coarse-to-fine progression of user intent and describe the training objective for downstream autoregressive modeling. The framework is shown in Figure 3.

3.1 Reformulation & Theoretical Motivation

Inspired by the multi-level nature of item semantics and the way users refine their preferences, we reformulate the item recommendation task as a *coarse-to-fine token sequence prediction* problem. Specifically, the model generates a structured sequence of discrete tokens that represent an item, progressing from high-level semantic categories to fine-grained details and behavioral signals. We denote a semantic hierarchy tokenized item as $S_i = [s_i^{(1)}, ..., s_i^{(K-1)}, s_i^{(K)}]$, where the first $K-1$ tokens correspond to progressive semantic levels (e.g., category, title, description), and the final token $s_i^{(K)}$ encodes collaborative filtering (CF) information. We place the CF token last because it captures user-specific interaction signals that are most informative when conditioned on previously decoded semantic content, allowing the model to disambiguate user preferences

given coarse-level intent. The overall generation probability is modeled as:

$$\mathbb{P}(s^{(K)}, ..., s^{(1)}, \mathcal{H}_u) = \prod_{k=0}^{K} \mathbb{P}(s^{(k)} \mid s^{(<k)}, \mathcal{H}_u), \tag{2}$$

where each token $s^{(k)}$ is generated in a coarse-to-fine manner, conditioned on all previously generated coarser-level tokens $s^{(<k)}$ and the user's interaction history $\mathcal{H}_u$, thereby progressively refining the semantic specificity of the predicted item.

Furthermore, we provide a theoretical justification for why hierarchical coarse-to-fine tokenization leads to lower expected dissimilarity compared to independent flat token prediction.

Theoretical setup. We consider two types of generation strategies:

- **Hierarchical decoding setting:** This corresponds to the decoding strategy used in CoFiRec, where tokens are generated sequentially from coarse to fine granularity. Let p denote the conditional probability of the correct prediction at each level. We assume that the model is better than random guess (i.e., $p > 1/V$, where V is the vocabulary size at each level).

  We can view hierarchical decoding as a traversal over a complete V-ary tree of depth K, where each internal node represents a prefix (partial code) and each edge corresponds to selecting a token from a vocabulary of size V. Each item is uniquely represented by a path from the root to a leaf, consisting of K sequential token choices. At each level, the probability of correctly predicting the next token (conditioned on all previous ones) is p. We define the dissimilarity $d(i, \hat{i})$ between the predicted item $\hat{i}$ and the ground-truth item i as half of their distance in the tree.

- **Independent decoding setting:** This models the generation process of methods that do not distinguish token granularity. Each token is predicted independently from a flat vocabulary of size V, with the same per-token success probability p for fair comparison.

Proposition 1 *Let $\mathbb{E}_{\text{hier}}[d(i, \hat{i})]$ and $\mathbb{E}_{\text{indep}}[d(i, \hat{i})]$ denote the expected dissimilarity between the predicted item $\hat{i}$ and the ground-truth item i under hierarchical and independent decoding, respectively. Then,*

$$\mathbb{E}_{\text{hier}}[d(i, \hat{i})] < \mathbb{E}_{\text{indep}}[d(i, \hat{i})]. \tag{3}$$

Proof sketch. The complete proof is deferred to Appendix A.

Hierarchical decoding. The decoder predicts each token level by level. If i and $\hat{i}$ differ at level k, the dissimilarity is $K - k$. The probability of success up to level k is p^k, and complete success has probability p^K. Hence, by the linearity of expectation, the expected dissimilarity is:

$$\mathbb{E}_{\text{hier}}[d(i, \hat{i})] = K - \sum_{k=1}^{K} p^k = K - p - p^2 - \cdots - p^K. \tag{4}$$

Independent decoding. Each of the K tokens is predicted independently without leveraging the hierarchical structure. This corresponds to randomly sampling a code from the V^K possible leaves, where each leaf (i.e., each item) has equal probability. The resulting expected dissimilarity is:

$$\mathbb{E}_{\text{indep}}[d(i, \hat{i})] = \left(K - \sum_{k=1}^{K} \frac{1}{V^k} \right) \cdot \frac{1 - p^K}{1 - \frac{1}{V^K}}. \tag{5}$$

Using the fact that the model is better than random guess (i.e., $p > 1/V$), we can show that $\mathbb{E}_{\text{hier}}[d(i, \hat{i})] < \mathbb{E}_{\text{indep}}[d(i, \hat{i})]$. This establishes that hierarchical coarse-to-fine decoding yields a lower expected dissimilarity between predicted and ground-truth items, confirming the theoretical advantage of structured tokenization.

3.2 Coarse-to-Fine Tokenizer

Hierarchy Construction. We construct a multi-level semantic hierarchy for each item based on commonly available metadata fields in product recommendation systems, including *category*, *title*, and *description*. These attributes are widely present across most e-commerce and content recommendation platforms and naturally reflect different levels of semantic granularity. In our experiments, we use these three levels as the hierarchical structure of item semantics. For a small number of items whose category field is missing, we apply a simple information extraction (IE) procedure to infer category-like terms from their titles or descriptions. The constructed K-level hierarchy for each item i is thus represented as $[x_i^{(1)}, x_i^{(2)}, ..., x_i^{(K)}]$, where the first $K-1$ levels correspond to textual semantics (*category*, *title*, *description*), and the final level $x_i^{(K)}$ encodes *collaborative filtering (CF)* information that captures user–item interaction patterns beyond textual content.

CoFiRec Semantic Tokenizer. Given the constructed semantic hierarchy information, where the first $K-1$ levels correspond to progressive semantic levels. For each semantic level $k \in \{1, 2, \ldots, K-1\}$, we first obtain an embedding representation $\boldsymbol{e}_i^{(k)}$ extracted from a pretrained LLM encoder (Grattafiori et al., 2024) , which captures contextualized semantic information from the corresponding text field $x_i^{(k)}$. We encode each level independently using a shared semantic encoder f^{Sem}. The encoder is typically implemented as a multi-layer perceptron (MLP) that projects the input feature into the code space. Each level is quantized using a coarse-to-fine vector quantization module Q^k to capture semantics at different levels:

$$\boldsymbol{h}_i^{(k)} = f^{\text{Sem}}(\boldsymbol{e}_i^{(k)}), \quad s_i^{(k)} = \text{Q}^k(\boldsymbol{h}_i^{(k)}), \quad \text{for } k = 1, ..., K-1. \tag{6}$$

All semantic levels share the same encoder and decoder to encourage consistency across levels.

CoFiRec CF Tokenizer. We tokenize CF information as the final K-th token. For each item i, we retrieve its collaborative embedding $\boldsymbol{e}_i^{\text{cf}}$ from a pretrained CF model (Kang and McAuley, 2018), and encode it with a separate CF-specific encoder and quantize using a dedicated codebook:

$$\boldsymbol{h}_i^{(K)} = f^{\text{CF}}(\boldsymbol{e}_i^{\text{cf}}), \quad s_i^{(K)} = \text{Q}^{\text{CF}}(\boldsymbol{h}_i^{(K)}) = \text{Q}^K(\boldsymbol{h}_i^{(K)}). \tag{7}$$

Vector Quantization Module. For each semantic level k, we maintain a learnable codebook with code embeddings $\{\boldsymbol{c}_j^{(k)}\}_{j=1}^{|\mathcal{V}^k|}$. The vector quantization process $\text{Q}^k(\cdot)$ assigns the continuous feature $\boldsymbol{h}_i^{(k)}$ to its nearest codeword in $\mathcal{V}^k$ via:

$$s_i^{(k)} \leftarrow \text{Q}^k(\boldsymbol{h}_i^{(k)}) = \arg\min_j \left\| \boldsymbol{h}_i^{(k)} - \boldsymbol{c}_j^{(k)} \right\|^2, \tag{8}$$

where $s_i^{(k)}$ denotes the selected code index for the k-th token of item i, and $\boldsymbol{c}_i^{(k)}$ represents the corresponding selected embedding for item i from the k-th codebook. This quantization operation is optimized using the straight-through estimator and a commitment loss, as described above.

Optimization. Each vector quantizer is trained with two objectives: (1) a reconstruction loss to ensure that each decoded representation approximates its original input, and (2) a commitment loss to ensure stable code usage. The total semantic tokenizer loss across all levels is defined as:

$$\mathcal{L}_{\text{Tokenizer}} = \mathcal{L}_{\text{Recon}} + \mathcal{L}_{\text{Commit}}, \tag{9}$$

where the reconstruction loss is computed between the input embedding $\boldsymbol{e}_i^{(k)}$ and its corresponding code embedding $\boldsymbol{c}_i^{(k)}$ from the k-th codebook at each semantic level k:

$$\mathcal{L}_{\text{Recon}} = \sum_{k=1}^{K-1} \underbrace{\left\| \boldsymbol{e}_i^{(k)} - g^{\text{Sem}}(\boldsymbol{c}_i^{(k)}) \right\|^2}_{\mathcal{L}_{\text{Sem}}} + \underbrace{\left\| \boldsymbol{e}_i^{cf} - g^{\text{CF}}(\boldsymbol{c}_i^{(K)}) \right\|^2}_{\mathcal{L}_{\text{CF}}}. \tag{10}$$

where g^{Sem} is a shared semantic decoder that reconstructs continuous semantic embeddings from the corresponding code embeddings, and g^{CF} is a separate decoder used to reconstruct CF embeddings from CF

Table 1 Performance comparison of different methods on the Instruments, Yelp, and Beauty datasets. The best and second-best results are marked in **Bold** and Underlined, respectively. "R@K": Recall@K; "N@K": NDCG@K; "BERT4R.": BERT4Rec; "LGCN": LightGCN; "IDGenR.": IDGenRec. "SemID" and "CID" are adapted from (Hua et al., 2023). The improvements of CoFiRec are statistically significant ($p \ll 0.05$).

Dataset	Metric	ID-based Recommendation				Generative Recommendation						
		MF	BERT4R.	LGCN	SASRec	BigRec	SemID	CID	LETTER	TIGER	IDGenR.	**CoFiRec**
Instruments	R@5	0.0479	0.0671	0.0794	0.0751	0.0513	0.0775	0.0809	0.0807	0.0842	0.0843	**0.0897**
	R@10	0.0735	0.0822	0.0100	0.0947	0.0576	0.0964	0.0987	0.0982	0.1035	0.0999	**0.1118**
	N@5	0.0330	0.0560	0.0662	0.0627	0.0470	0.0669	0.0695	0.0704	0.0720	0.0735	**0.0752**
	N@10	0.0412	0.0608	0.0728	0.0690	0.0491	0.0730	0.0751	0.0760	0.0782	0.0785	**0.0820**
Yelp	R@5	0.0220	0.0186	0.0234	0.0183	0.0154	0.0202	0.0219	0.0164	0.0203	0.0230	**0.0252**
	R@10	0.0381	0.0291	0.0383	0.0296	0.0169	0.0324	0.0338	0.0259	0.0270	0.0323	**0.0395**
	N@5	0.0138	0.0115	0.0146	0.0116	0.0137	0.0131	0.0140	0.0116	0.0130	**0.0165**	0.0162
	N@10	0.0190	0.0159	0.0193	0.0152	0.0142	0.0170	0.0181	0.0147	0.0173	0.0195	**0.0208**
Beauty	R@5	0.0294	0.0203	0.0305	0.0380	0.0243	0.0393	0.0404	0.0263	0.0349	0.0369	**0.0444**
	R@10	0.0474	0.0347	0.0511	0.0588	0.0299	0.0584	0.0597	0.0427	0.0353	0.0550	**0.0679**
	N@5	0.0145	0.0124	0.0194	0.0246	0.0181	0.0273	0.0284	0.0161	0.0224	0.0252	**0.0295**
	N@10	0.0191	0.0170	0.0260	0.0313	0.0198	0.0335	0.0347	0.0213	0.0283	0.0311	**0.0370**

tokens due to their different representation spaces. The commitment loss is:

$$\mathcal{L}_{\text{Commit}} = \sum_{k=1}^{K} \left\| \text{sg}[\boldsymbol{h}_i^{(k)}] - \boldsymbol{c}_i^{(k)} \right\|^2 + \mu \left\| \boldsymbol{h}_i^{(k)} - \text{sg}[\boldsymbol{c}_i^{(k)}] \right\|^2, \tag{11}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator, $\boldsymbol{c}_i^{(k)}$ is the selected code embedding corresponding to $s_i^{(k)}$ from codebook $\mathcal{V}^k$ at level k, and μ is the balancing coefficient.

After optimizing the CoFiRec tokenizer with $\mathcal{L}_{\text{Tokenizer}}$, each item i is represented by a token sequence $[s_i^{(1)}, ..., s_i^{(K-1)}, s_i^{(K)}]$, which reflects a coarse-to-fine progression from high-level semantics to behaviorally grounded signals.

3.3 Coarse-to-Fine Autoregressive Generation

Let $S_u = [S_{i_1}, ..., S_{i_T}]$ denote the tokenized interaction history of user u, where the t-th item i_t is represented by a coarse-to-fine token sequence $S_{i_t} = [s_{i_t}^{(1)}, s_{i_t}^{(2)}, ..., s_{i_t}^{(K)}]$. We train an autoregressive language model to predict the next item's token sequence one token at a time, following the same hierarchical order.

Embedding Layer. To help the model distinguish the role of each token, we use three types of learnable embeddings before inputting to the autoregressive language model's encoder:

$$\boldsymbol{e}_t^{(k)} = \boldsymbol{E}_{\text{id}}(s_{i_t}^{(k)}) + \boldsymbol{E}_{\text{level}}(k) + \boldsymbol{E}_{\text{pos}}(t), \tag{12}$$

where $\boldsymbol{E}_{\text{id}}$ embeds the token ID, $\boldsymbol{E}_{\text{level}}(k)$ encodes the semantic level k, and $\boldsymbol{E}_{\text{pos}}(t)$ encodes the item position t in the user history. These embeddings provide the model with structural cues across levels and positions, facilitating more coherent coarse-to-fine generation.

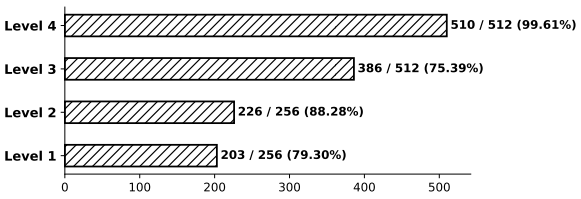
Downstream Training Objective. We adopt a ranking-guided generation loss to enhance token-level discriminability:

$$\mathcal{L}_{\text{Rank}} = -\sum_{k=1}^{K} \log \frac{\exp(\phi(s_{i_{T+1}}^{(k)})/\tau)}{\sum_{v \in \mathcal{V}^k} \exp(\phi(v)/\tau)}, \tag{13}$$

where $s_{i_{T+1}}^{(k)}$ is the ground-truth token at level k of the next item at the $T+1$ step to be predicted, $\phi(s_{i_{T+1}}^{(k)})$ is the model's predicted logit, $\mathcal{V}^k$ is the vocabulary at level k, and τ is a temperature hyperparameter. This objective encourages the model to rank the correct tokens higher than distractors within each semantic level.

Method	R@5	R@10	N@5	N@10
Reverse Indices	0.0859	0.1047	0.0724	0.078
Random Indices	0.0834	0.0991	0.0725	0.077
w/o $\boldsymbol{E}_{\text{level}}$ & $\boldsymbol{E}_{\text{pos}}$	<u>0.0885</u>	0.1072	**0.0753**	<u>0.081</u>
w/o $\boldsymbol{e}_i^{\text{cf}}$	0.0870	<u>0.1082</u>	0.0740	0.080
VAR w/ Text	0.0675	0.0798	0.0599	0.063
CoFiRec	**0.0897**	**0.1118**	<u>0.0752</u>	**0.082**

(a) Ablation study on the Instruments dataset.



(b) Codebook utilization rate of CoFiRec.

Table 2 Results of the ablation study. (a) shows the ablation study. (b) shows the codebook utilization.

4 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of CoFiRec in generative recommendation. This section is organized into three parts. First, we describe the *experimental setup*, including datasets, evaluation protocols, and implementation details. Next, we present the *main comparison* with a variety of state-of-the-art baselines to assess overall performance. Finally, we perform detailed *ablation studies* to analyze the contribution of each component, such as hierarchical tokenization, semantic levels, and codebook configurations.

4.1 Experimental Setup

Datasets. We evaluate our model on three real-world recommendation datasets from different domains: 1) **Instruments** (Musical Instruments) is sourced from the Amazon review datasets (McAuley et al., 2015) , covering user interactions with music-related products. 2) **Beauty** also comes from Amazon reviews, focusing on interactions with beauty products. 3) **Yelp** (Yelp, 2023) is a popular benchmark dataset containing user reviews of local businesses. We follow the data preprocessing procedure in (Kang and McAuley, 2018; Li et al., 2020) by removing users and items with fewer than 5 interactions. We also discard items with entirely missing metadata for building semantic hierarchy. To model user behavior sequentially, we adopt the sequential recommendation setting (Quadrana et al., 2017) and use the *leave-one-out* strategy to split the data into training, validation, and test sets (Kang and McAuley, 2018; Sun et al., 2019). For training, we limit each user's interaction history to the most recent 20 items, as done in (Chen et al., 2021; Sun et al., 2019).

Compared Methods. We compare CoFiRec with a diverse set of strong baselines that fall into two major categories: For the **ID-based models**, we include representative approaches such as MF (Rendle, 2010), which factorizes the user–item interaction matrix into latent embeddings; BERT4Rec (Sun et al., 2019), which formulates sequential recommendation as a masked item prediction task using bidirectional transformers; LightGCN (He et al., 2020), which simplifies graph-based collaborative filtering by retaining only the essential message-passing layers; and SASRec (Kang and McAuley, 2018), which applies self-attention to capture both short- and long-range dependencies in user behavior sequences. For the **LLM-based generative models**, we consider BigRec (Bao et al., 2025), which directly generates recommendations from item titles; P5-SemID and P5-CID (Hua et al., 2023), which introduce structured identifiers based on metadata hierarchies and collaborative trees; LETTER (Wang et al., 2024), which combines residual quantization with a diversity constraint to encourage more informative representations; TIGER (Rajput et al., 2023), which represents items as quantized code sequences via RQ-VAE to enable token generation; (Wang et al., 2024)and IDGenRec (Tan et al., 2024), which learns concise textual IDs from human vocabulary.

Evaluation Settings. We follow the evaluation protocol of TIGER and report performance using Recall@K and NDCG@K, where $K \in \{5, 10\}$. Both metrics are computed over the ranked list of candidate items for each user. We adopt a leave-one-out evaluation scheme, where the most recent interaction of each user is used for testing, the second most recent for validation, and the rest for training. All models share the same data splits and preprocessing for fair comparison. For each method, we select the checkpoint with the best validation performance for testing. For generative models, beam search with beam size 20 is used during inference following prior work (Rajput et al., 2023).

Implementation Details. We represent each item using a token sequence of length $K = 4$, where the first three tokens are derived from a semantic tokenizer and the last token comes from a CF tokenizer. For semantic encoding, we extract embeddings from item metadata using a pre-trained LLaMA-7B. For collaborative signals, we adopt a pre-trained SASRec model as the CF encoder, with an embedding dimension of 768. In the first-stage tokenization, we train the discrete tokenizer for 10,000 epochs using the AdamW optimizer with a learning rate of 1e-3. Each token position corresponds to a separate codebook, with sizes searched in {256, 512, 1024}. Both the encoder and decoder are implemented as multi-layer perceptrons. In the second-stage finetuning, we follow the setup of TIGER (Rajput et al., 2023) and use T5 (Raffel et al., 2020) as the generative backbone. The temperature τ is set to 1, and μ is set to 0.25. The learning rate is searched from {3e-4, 4e-4, 5e-4, 6e-4}.

4.2 Overall Performance

We compare CoFiRec with both *ID-based* and *generative* recommendation models that adopt different item tokenization strategies, evaluated on three public datasets: Instruments, Yelp, and Beauty. The results are summarized in Table 1. Overall, generative recommendation models substantially outperform ID-based methods such as MF, BERT4Rec, LightGCN, and SASRec, highlighting the advantage of leveraging textual and semantic signals beyond discrete item IDs. Among generative approaches, IDGenRec performs competitively by encoding item semantics into textual identifiers, demonstrating the benefit of integrating language-based representations. Building upon this idea, CoFiRec further improves performance by introducing a hierarchical coarse-to-fine tokenization scheme that better captures the structured semantics of items. This design preserves multi-level attribute information (e.g., category, title, description, and collaborative signals) and aligns with how users progressively refine their preferences in real-world web scenarios. As shown in Table 1, CoFiRec delivers a 6.4% improvement over the best-performing baseline, demonstrating its strong generalization ability and effective semantic organization. These results confirm that explicitly modeling the hierarchical structure of item semantics enables more accurate generative recommendation.

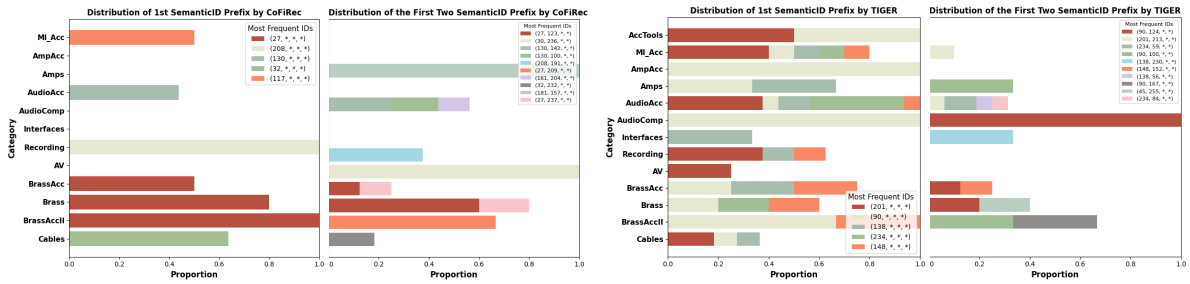
4.3 Ablation Study

We conduct ablation experiments in Table 2a to examine the contribution of each proposed component in CoFiRec.

- *Effect of token structure:* to assess the importance of the coarse-to-fine token structure, we test two variants: reversed token order and random permutation. Both lead to significant performance drops, with the random order performing worst. This indicates that preserving semantic structure in tokenization facilitates more accurate generation in autoregressive models.
- *Effect of collaborative filtering integration.* We replace the final CF token $\boldsymbol{e}_i^{\text{cf}}$ with an additional semantic-level token quantized from title and description. This modification degrades performance, showing that incorporating collaborative signals at the finest level provides essential disambiguation for semantically similar items. The CF token complements textual semantics by capturing user–item co-occurrence patterns.
- *Effect of embedding components in the generation phase:* removing the level embedding $\boldsymbol{E}_{\text{level}}$ and position embedding $\boldsymbol{E}_{\text{pos}}$ causes a clear decline in performance, demonstrating that explicitly encoding token type and position helps the model distinguish semantic roles more effectively.
- *Comparison with alternative hierarchy construction method:* we also adapt VAR (Tian et al., 2024) by applying multi-scale pooling over textual embeddings to construct a hierarchy from the embedding space. This yields inferior results compared to our explicit semantic decomposition, likely due to semantic loss introduced by pooling in the one-dimensional embedding space.

5 Analysis

To gain deeper insights into CoFiRec's behavior, we conduct analyses from several perspectives. We study the *impact of codebook sizes*, examine the *semantic structure of ID distributions*, and present a *case study on*



(a) CoFiRec (b) TIGER

Figure 4 Distributions of semantic IDs on the Instruments dataset. The left panel shows the distribution of CoFiRec, where both the first and first-two ID prefixes align closely with category semantics and form clear hierarchical structures, while the right panel shows the distribution of TIGER, where prefixes are scattered across categories with weaker semantic consistency.

Table 3 Impact of different codebook sizes at each level on the Instruments dataset.

Size	Level Size	R@5	R@10	N@5	N@10
Small	[256,256,256,256]	0.0860	0.1078	0.0725	0.0795
Medium	[256,256,512,512]	**0.0893**	**0.1111**	**0.0751**	**0.0822**
Large	[256,256,512,1024]	0.0867	0.1065	0.0740	0.0804

cold-start generalization. We further explore *automatic hierarchy construction* from minimal metadata and evaluate *backbone adaptability* across different LLM architectures.

5.1 Impact of Codebook Sizes

CoFiRec uses separate codebooks at each semantic level. Code utilization as shown in Tabel 2b gradually increases from coarse to fine levels, indicating that deeper layers encode more detailed item semantics and therefore require larger code capacity for discrimination. This observation is consistent with the intuition that higher-level categories represent broader semantic clusters, while deeper levels encode finer-grained and more distinctive item semantics. Moreover, CoFiRec naturally reduces code collisions through its hierarchical semantic decomposition and the integration of collaborative signals, as shown in Appendix B. This design contrasts with the RQ-VAE tokenizer in TIGER, which relies on appending additional tokens to resolve code overlaps, often leading to redundancy and semantic drift. We further analyze the impact of different codebook configurations, as shown in Table 3. The medium configuration (256, 256, 512, 512) achieves the best overall performance, suggesting that fine-grained levels benefit from moderately larger codebooks to distinguish similar items. However, excessively large codebooks, e.g., (256, 256, 512, 1024), may over-partition the latent space, fragmenting semantics and slightly degrading performance. These results underscore the importance of balanced capacity allocation across semantic levels for achieving both expressiveness and stability in generative recommendation.

5.2 Qualitative Analysis of ID Distribution of CoFiRec Tokenization

To better understand how well different tokenization methods preserve semantic structure, we visualize the distribution of semantic IDs across item categories for both CoFiRec and TIGER on the Instruments dataset. Figure 4 shows the distribution of the most frequent first-level semantic ID prefixes. We observe that under CoFiRec, items sharing the same prefix are highly aligned with specific high-level categories, for example, items in the same coarse-grained category tend to share the same first token. In contrast, TIGER's IDs are more scattered, with no clear pattern of prefix-category alignment, indicating weaker semantic coherence in its token assignments. Figure 4 further examines the two-level prefix distributions. It reveals that even when items share the same first-level token (coarse semantics), they are further differentiated at the second level, capturing fine-grained category distinctions. This hierarchical organization is especially evident in CoFiRec, but much less structured in TIGER. These results confirm that CoFiRec's tokenizer preserves

Table 4 Performance comparison under warm and cold-start scenarios on the Instruments dataset.

Scenario	Method	R@5	R@10	N@5	N@10
Warm	TIGER	0.0845	0.1038	0.0723	0.0784
	CoFiRec (+4.7%)	**0.0889**	**0.1101**	**0.0747**	**0.0816**
Cold	TIGER	0.0654	0.0801	0.0673	0.0688
	CoFiRec (+10.0%)	**0.0707**	**0.0943**	**0.0690**	**0.0767**

Table 5 Performance comparison of CoFiRec and TIGER tokenization under different backbone models on the sampled Instruments dataset.

Backbone	Method	R@5	R@10	N@5	N@10
T5	TIGER	0.0444	0.0484	**0.0293**	0.0307
	CoFiRec	**0.0484**	**0.0605**	0.0288	**0.0327**
LLaMA-1B	TIGER	0.0202	0.0282	0.0202	0.0226
	CoFiRec	**0.0403**	**0.0444**	**0.0330**	**0.0342**
LLaMA-3B	TIGER	0.0242	0.0282	0.0192	0.0205
	CoFiRec	**0.0403**	**0.0403**	**0.0373**	**0.0373**

semantic hierarchy through a coarse-to-fine structure, grouping similar items under shared prefixes while capturing finer distinctions in deeper tokens, thereby enhancing both semantic coherence and interpretability.

5.3 Case Study on Cold-Start Generalization

To evaluate the generalization ability of our model in cold-start scenarios, we construct a test set containing items that do not appear in the training data. Cold items are defined as those in the bottom 2% of item frequency, measured by their occurrences as the last item in user interaction sequences. The tokenizer is trained solely on the training items and then used to infer semantic IDs for these unseen cold-start items. As shown in Table 4, CoFiRec consistently outperforms TIGER on both warm and cold subsets, achieving a 4.7% improvement in the warm-start case and a larger 10.0% gain under the cold-start setting. This clearly demonstrates that CoFiRec not only maintains strong performance when sufficient user–item interactions are available, but also generalizes more effectively to unseen or infrequent items. We attribute this advantage to its coarse-to-fine tokenization design, which captures structured semantic dependencies among item attributes. By leveraging this hierarchical organization, CoFiRec effectively transfers semantic patterns from known items to unseen ones, achieving robust performance even under challenging cold-start conditions.

5.4 CoFiRec with Automatically Constructed Hierarchies

Although item hierarchical metadata is widely available in most recommendation datasets, real-world scenarios may lack such structured information. To explore whether meaningful hierarchies can still be automatically derived under this condition, we investigate the ability to generate hierarchical semantics from minimal item metadata. Due to budget constraints, we randomly select 1% users and their corresponding 2,258 interacted items from the Instruments dataset. Using GPT-4o, we construct 4-level semantic hierarchies based solely on each item's title and description, without access to any structured metadata such as category or brand (examples are provided in Appendix E). We denote this variant as AHC (Automatic Hierarchy Construction). While TIGER and CoFiRec rely on complete metadata fields, AHC leverages only the GPT-generated hierarchies to form its tokenization structure. As shown in Appendix C, AHC achieves competitive performance, approaching the results of full-metadata CoFiRec. This demonstrates that large language models can effectively infer coarse-to-fine semantic relationships from unstructured text, serving as a viable alternative when explicit metadata is unavailable. These findings highlight the broader potential of integrating LLM-based reasoning with generative recommendation for automatic semantic enrichment.

5.5 Backbone Adaptability

We further evaluate the adaptability of our semantic tokenizer by applying the item IDs generated by CoFiRec to finetune different LLM backbones for sequential recommendation, as shown in Table 5. Across all backbones,

CoFiRec consistently outperforms TIGER, demonstrating that its structured tokenization provides a more informative and transferable representation for downstream models. While TIGER achieves good performance under the T5 backbone, its effectiveness drops noticeably when scaling up to larger decoder-only architectures such as LLaMA-1B and LLaMA-3B. In contrast, CoFiRec maintains stable and competitive results across all backbones, demonstrating that its coarse-to-fine tokenization provides a more consistent and transferable representation.

6 Related Work

In this section, we review related work in two key areas: generative recommendation and tokenization for recommendation.

6.1 Generative Recommendation

Traditional sequential recommender systems often rely on large item embedding tables (He et al., 2020; Kang and McAuley, 2018; Mnih and Salakhutdinov, 2007), which pose scalability and memory challenges. In contrast, generative recommendation (Geng et al., 2022; Rajput et al., 2023; Zhai et al., 2024; Yue et al., 2023; Hou et al., 2025) encodes items as tokens from a shared vocabulary and generates the next item in an autoregressive manner, enabling open-ended, scalable, and compositional recommendation. Recent methods have explored this paradigm from various angles. P5 (Geng et al., 2022) formulates recommendation as a language generation task, prompting pretrained language models using instruction templates and textual item identifiers. TIGER (Rajput et al., 2023) represents each item as a semantic ID composed of discrete codes via residual vector quantization and generates them using a Transformer decoder. IDGenRec (Tan et al., 2024) generates short, interpretable token sequences directly from item metadata. These works demonstrate the potential of generative recommenders in terms of scalability (Rajput et al., 2023; Lin et al., 2024b; Xu et al., 2024), generalization (Rajput et al., 2023; Huang et al., 2024), and LLM alignment (Jin et al., 2023). However, most methods represent each item as a flat ID or sequence of tokens without internal structure, making the generation process less controllable. This also increases the risk of pruning correct items early during beam search, due to the lack of intermediate semantic cues. In contrast, our method introduces a hierarchical tokenization scheme that preserves multi-level item semantics and aligns naturally with users preference refinement.

6.2 Tokenization for Recommendation

A core challenge in LLM-based recommendation is how to tokenize items and interactions in a way that balances semantic fidelity, behavioral alignment, and modeling flexibility. Prior work has explored various tokenization paradigms. ID-based methods (Geng et al., 2022; Hua et al., 2023; Chu et al., 2023) offer simplicity and efficiency, but lack semantic expressiveness. Textual tokenization (Zhang et al., 2023a; Zheng et al., 2024; Lin et al., 2024a; Tan et al., 2024; Bao et al., 2025) leverages pretrained language models but often struggles with structural consistency and noisy representations. Vector quantization-based approaches (Rajput et al., 2023; Zheng et al., 2024; Wang et al., 2024; Lin et al., 2025; Hou et al., 2025; Deng et al., 2025; Han et al., 2025) discretize continuous embeddings into compact token sequences, enabling compatibility with autoregressive generation. Despite their merits, most of these approaches collapse item semantics into a flat representation, discarding hierarchical structure and making it difficult to integrate collaborative signals meaningfully. Some recent methods attempt to inject CF information (Wang et al., 2024; Zhang et al., 2023b), yet they generally overlook positional structure and coarse-to-fine conditioning. P5-SemID (Hua et al., 2023) proposes hierarchical identifiers using taxonomies, but they are non-learnable and lack discriminative power or CF awareness. In contrast, our approach introduces a learnable, structured tokenization framework that constructs a coarse-to-fine semantic hierarchy from item metadata, applies coarse-to-fine quantization, and integrates CF signals as an additional fine-grained token. Our method is conceptually related to VAR (Tian et al., 2024), which performs hierarchical generation in vision, but differs in its application to discrete, multi-level textual metadata. Unlike the continuous latent vision maps in VAR, our tokenization handles discrete semantic structures, requiring explicit construction and level-wise quantization and generation tailored for recommendation tasks.

7 Conclusion

In this work, we proposed CoFiRec, a generative recommendation framework that models item semantics in a coarse-to-fine manner through structured tokenization and hierarchical generation. By preserving the multi-level structure of item metadata and integrating collaborative signals, CoFiRec enables more accurate and interpretable recommendations. Both theoretical analysis and empirical results confirm its effectiveness across diverse datasets and model backbones, highlighting the importance of structured token design in generative recommendation. Beyond improving performance, CoFiRec provides a new perspective on how generative models can capture user intent evolution through hierarchical semantics.

References

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *ACM Transactions on Recommender Systems*, 3(4):1–27, 2025.

Tianlong Chen, Zhenyu Liu, Zhen Wang, and Zhangyang Liu. Learning discrete representations via mutual information estimation and contrastive learning. In *International Conference on Machine Learning (ICML)*, 2021.

Zhixuan Chu, Hongyan Hao, Xin Ouyang, Simeng Wang, Yan Wang, Yue Shen, Jinjie Gu, Qing Cui, Longfei Li, Siqiao Xue, et al. Leveraging large language models for pre-trained recommender systems. *arXiv preprint arXiv:2308.10837*, 2023.

Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*, 2025.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, et al. Mtgr: Industrial-scale generative recommendation framework in meituan. *arXiv preprint arXiv:2505.18654*, 2025.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

Yupeng Hou, Jianmo Ni, Zhankui He, Noveen Sachdeva, Wang-Cheng Kang, Ed H Chi, Julian McAuley, and Derek Zhiyuan Cheng. Actionpiece: Contextually tokenizing action sequences for generative recommendation. *arXiv preprint arXiv:2502.13581*, 2025.

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

Feiran Huang, Zhenghang Yang, Junyi Jiang, Yuanchen Bei, Yijie Zhang, and Hao Chen. Large language model interaction simulator for cold-start item recommendation. *arXiv e-prints*, pages arXiv–2402, 2024.

Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, et al. Language models as semantic indexers. *arXiv preprint arXiv:2310.07815*, 2023.

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

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.

Chao Li, Da Pi, Yan Liu, Jun Yan, and Philip S Yu. Time-aware sequential recommendation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 4699–4706, 2020.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443*, 2023.

Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Bridging items and language: A transition paradigm for large language model-based recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1816–1826, 2024a.

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 365–374, 2024b.

Xinyu Lin, Haihan Shi, Wenjie Wang, Fuli Feng, Qifan Wang, See-Kiong Ng, and Tat-Seng Chua. Order-agnostic identifier for large language model-based generative recommendation. *arXiv preprint arXiv:2502.10833*, 2025.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

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.

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.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

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.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. Idgenrec: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364, 2024.

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.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

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.

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.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. Openp5: Benchmarking foundation models for recommendation. *arXiv preprint arXiv:2306.11134*, 2023.

Wujiang Xu, Zujie Liang, Jiaojiao Han, Xuying Ning, Wenfang Lin, Linxun Chen, Feng Wei, and Yongfeng Zhang. Slmrec: empowering small language models for sequential recommendation. *arXiv e-prints*, pages arXiv–2405, 2024.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *Advances in Neural Information Processing Systems*, 36: 24247–24261, 2023.

Yelp. Yelp open dataset. *https://www.yelp.com/dataset*, 2023.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*, 2023.

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.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*, 2023a.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. *arXiv preprint arXiv:2310.19488*, 2023b.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE, 2024.

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. Collaborative large language model for recommender systems. *arXiv preprint arXiv:2311.01343*, 2023.

Appendix

A Proof for Proposition 1

Proposition 1. Let $\mathbb{E}_{\text{hier}}[d(i,\hat{i})]$ and $\mathbb{E}_{\text{indep}}[d(i,\hat{i})]$ denote the expected dissimilarity between the predicted item $\hat{i}$ and the ground truth item i, under hierarchical and independent decoding respectively. Then,

$$\mathbb{E}_{\text{hier}}[d(i,\hat{i})] < \mathbb{E}_{\text{indep}}[d(i,\hat{i})]. \tag{14}$$

Proof. We model hierarchical decoding as a traversal over a complete V-ary tree of depth K, where each internal node represents a prefix (partial code), and each edge corresponds to selecting a token from a vocabulary of size V. Each item is uniquely represented by a path from the root to a leaf, consisting of V sequential token choices. At each level, the probability of correctly predicting the next token (conditioned on all previous ones) is p.

Hierarchical decoding. The decoder predicts each token level by level. If it fails at level k, the dissimilarity is defined as $K-k$. The probability of success at level k is p^k, and complete success has probability p^K. Hence, the expected dissimilarity is:

$$\mathbb{E}_{\text{hier}}[d(i,\hat{i})] = K - \sum_{k=1}^{K} p^k = K - p - p^2 - \cdots - p^K. \tag{15}$$

Independent decoding. Each of the V tokens is predicted independently, without leveraging the tree structure. This corresponds to randomly sampling a code from the V^K possible sequences, where each leaf (i.e., each item) has equal probability. Hence,

$$\mathbb{E}_{\text{indep}}[d(i,\hat{i})] = \left(K - \sum_{k=1}^{K} \frac{1}{V^k}\right) \cdot \frac{1-p^K}{1-\frac{1}{V^K}}. \tag{16}$$

Define an auxiliary function:

$$\psi(p) := \frac{K - \sum_{k=1}^{K} p^k}{1-p^K}, \tag{17}$$

which we will use to simplify the expected dissimilarity. In the following **Lemma 1**, we will show that the function $\psi(p)$ is strictly decreasing with respect to p over the interval $[0,1]$.

Since $p > 1/V$, then the ratio between hierarchical and independent decoding dissimilarity is smaller than 1:

$$\frac{\mathbb{E}_{\text{hier}}[d(i,\hat{i})]}{\mathbb{E}_{\text{indep}}[d(i,\hat{i})]} = \frac{K - \sum_{k=1}^{K} p^k}{\left(K - \sum_{k=1}^{K} \frac{1}{V^k}\right) \cdot \frac{1-p^K}{1-\frac{1}{V^K}}} \tag{18}$$

$$= \frac{\frac{K-\sum_{k=1}^{K} p^k}{1-p^K}}{\frac{K-\sum_{k=1}^{K} \frac{1}{V^k}}{1-\frac{1}{V^K}}} = \frac{\psi(p)}{\psi\left(\frac{1}{V}\right)} < 1. \tag{19}$$

It follows that $\mathbb{E}_{\text{hier}}[d(i,\hat{i})] < \mathbb{E}_{\text{indep}}[d(i,\hat{i})]$.

Conclusion. This result highlights the benefit of hierarchical decoding: correct predictions at higher levels constrain the prediction space at lower levels, resulting in smaller expected dissimilarity between predictions and the ground truth. Independent decoding lacks this structural bias and thus incurs a higher error on average.

Lemma 1 *The function* $\psi(p) := \frac{K-\sum_{k=1}^{K} p^k}{1-p^K}$ *is strictly decreasing over* $[0,1]$ *whenever* $K > 1$.

Proof. We rewrite the function as:

$$\psi(p) = \frac{K - \sum_{k=1}^{K} p^k}{1 - p^K} = 1 + \sum_{k=1}^{K-1} \frac{1 - p^k}{1 - p^K}. \tag{20}$$

It suffices to show that for every $1 \leq k \leq K-1$,

$$\psi_k(x) := \frac{1 - p^k}{1 - p^K} \tag{21}$$

is strictly decreasing. To analyze its monotonicity, we check the sign of its derivative:

$$\psi_k'(x) = -\frac{p^{k-1}}{(1-p^K)^2}(k + (K-k)p^K - Kx^{K-k}). \tag{22}$$

Since $-\frac{p^{k-1}}{(1-p^K)^2} < 0$ over $(0,1)$, it remains to analyze the sign of

$$\zeta_k(p) := k + (K-k)p^K - Kp^{K-k}. \tag{23}$$

Since for any $0 < x < 1$,

$$\zeta_k'(p) = -(K-k)Kp^{K-k-1}(1-p^k) < 0, \tag{24}$$

then $\zeta_k(p)$ is strictly decreasing over $(0,1)$. Thus, for any $0 < p < 1$,

$$\zeta_k(p) > \zeta_k(1) = k + (K-k) - K = 0. \tag{25}$$

Hence, for any $0 < p < 1$,

$$\zeta_k'(x) = -\frac{p^{k-1}}{(1-p^K)^2}\zeta_k(p) < 0. \tag{26}$$

Therefore, for every $1 \leq k \leq K-1$, $\psi_k(p)$ is strictly decreasing over $p \in [0,1]$. It follows that

$$\psi(p) = 1 + \sum_{k=1}^{K-1} \psi_k(p) \tag{27}$$

is strictly decreasing over $[0,1]$.

B Analysis on Code Collision

We analyze the impact of hierarchical tokenization on code collisions. Table 6 compares CoFiRec with the RQ-VAE tokenizer under the same codebook size (256 per level). CoFiRec achieves significantly lower collision rates across all datasets, reducing collisions by more than one order of magnitude. This demonstrates that decomposing item semantics into multiple hierarchical levels and incorporate collaborate signals as the most fine-grained token granularity allows more efficient utilization of the code space and avoids over-compression within a single latent space.

To further investigate how the size of the codebook affects collisions, we evaluate three configurations that correspond to the settings in Table 3. As shown in Table 7, enlarging the codebook capacity in deeper semantic levels further suppresses collisions. The collision rate remains near zero under the medium and large configurations, confirming that granularity-aware capacity allocation effectively mitigates redundancy in the token space.

C Automatic Hierarchy Construction (AHC)

In our main experiments, CoFiRec is built upon simple and widely available fields—*category*, *title*, and *description*—which exist in most practical product recommendation scenarios. To verify the generality of

Table 6 Code collision rate comparison between CoFiRec and RQ-VAE tokenizers.

Model	Instruments	Yelp	Beauty
RQ-VAE Tokenizer	0.0027	0.0410	0.0009
CoFiRec Tokenizer	**0.0001**	**0.0002**	**0.0002**

Table 7 Code collision rate of CoFiRec under different codebook size configurations on the Instruments dataset.

Codebook Size	Level Size	Collision Rate
Small	[256, 256, 256, 256]	0.0001
Medium	[256, 256, 512, 512]	0.0000
Large	[256, 256, 512, 1024]	0.0000

CoFiRec, we further explore whether meaningful hierarchical semantics can be constructed when structured metadata is unavailable. Specifically, we use GPT-4o to automatically generate 4-level hierarchies for items in the Instruments dataset based solely on their textual information. The generated hierarchies are used only to test the feasibility of our framework under minimal metadata conditions, rather than to replace real-world metadata. The GPT-based variant, denoted as AHC (Automatic Hierarchy Construction), serves as a fallback strategy for extreme cases where structured metadata is missing. Shown in Table 8, AHC still achieves competitive performance.

Table 8 Performance comparison on the Instruments dataset between TIGER, CoFiRec with full metadata, and CoFiRec with automatically constructed hierarchies (AHC).

Method	R@5	R@10	N@5	N@10
TIGER	0.0444	0.0484	0.0293	0.0307
CoFiRec w/ AHC	0.0444	0.0484	**0.0313**	0.0327
CoFiRec	**0.0484**	**0.0605**	0.0288	**0.0327**

D Future Work

Building upon the promising results of CoFiRec, several future directions can further extend its capabilities and impact:

(1) **End-to-end integration of tokenization and generation.** Currently, the semantic tokenizer and the generative recommendation model are trained separately. While this modularity facilitates transferability across tasks, a fully end-to-end framework could enable stronger mutual adaptation between tokenization and generation. Future work may investigate joint optimization strategies to align the latent token space with downstream generation objectives, improving both efficiency and semantic consistency.

(2) **Extension to multimodal recommendation.** Our present framework operates on textual metadata, but many real-world platforms involve rich multimodal signals such as product images, videos, or user interactions. Extending CoFiRec's coarse-to-fine tokenization to incorporate visual and other modalities offers a natural next step. Such an expansion could further enhance semantic grounding and lead to more comprehensive generative recommendation systems.

E Prompt for Automatic Hierachy Construction

Example: Generating Semantic Hierarchy with GPT

Input Prompt:
You are a product taxonomy expert specializing in musical instruments.
Given an item's title and description, generate a 4-level coarse-to-fine hierarchical categorization within the "Instruments" domain. The hierarchy should reflect increasing specificity from general product type (Level 1) to full product-level detail (Level 4). Do not include "Musical Instruments" or "Instruments" as Level 1 – assume the item is already within that category.

Guidelines:

- Each level must be a single short phrase or sentence.
- Each finer level should include or imply the coarser level's meaning.
- Level 4 should include brand, model, main features, and supported use cases, written as a concise product summary.
- Format the output as: Level 1: ..., Level 2: ..., Level 3: ..., Level 4: ...

Here is the item:

- **Title**: "Chrome Oval Style 1/4 Guitar Pickup Output Input Jack Plug Socket for Fender Strat."
- **Description**: "Specification: 100% Brand New, Never Used This is a mono Electric guitar output jack plate. Width at the widest point of the plate: 2.7cm Has 3 Total Prongs Measurement between the centre of the 2 screw holes: 3.8cm Replace your worn out parts Boat Style (Oval Indented) Standard recessed oval plate which is used commonly on some LP SG guitars Color:chrome Package Include: 1x Chrome Jack Plate."

Hierarchy Output:

- **Level 1:** Guitar Accessories
- **Level 2:** Guitar Hardware Components
- **Level 3:** Guitar Output Jack Plates
- **Level 4:** Chrome Oval Style 1/4 Guitar Pickup Output Input Jack Plug Socket for Fender Strat, featuring a mono electric guitar output jack plate with a standard recessed oval design, suitable for replacing worn-out parts on LP and SG guitars.

F Dataset Statistics

We summarize the datasets used in our experiments in Table 9. The three datasets—**Instruments**, **Yelp**, and **Beauty**—are derived from the Amazon review and Yelp dataset (Yelp, 2023) and are widely adopted in sequential recommendation research. Each dataset consists of user–item interaction sequences, where for each user u, the sequence is denoted as $\mathcal{S}_u = [i_1, i_2, \ldots, i_T]$, and the goal is to predict the next item i_{T+1}. All datasets are preprocessed by filtering out users and items with fewer than five interactions and sorting interactions chronologically.

Table 9 Dataset statistics for Instruments, Yelp, and Beauty.

Dataset	# Users	# Items	# Interactions	Avg. Seq. Length
Instruments	24,772	9,922	206,153	8.32
Yelp	30,431	20,033	316,354	10.40
Beauty	22,363	12,101	198,502	8.88

Table 10 Runtime comparison between CoFiRec and TIGER.

Method	Stage	Training Time	Testing Time
TIGER	Tokenization	0.3488s / step	0.3113s / step
CoFiRec	Tokenization	0.3584s / step	0.3256s / step
TIGER	Generation	400.6869s / epoch	0.0998s / step
CoFiRec	Generation	402.3185s / epoch	0.0943s / step

G Runtime and Scalability Analysis

We analyze the computational efficiency and scalability of CoFiRec compared with TIGER. Since access to large-scale industrial datasets is limited, we follow representative generative recommendation studies (e.g., TIGER, P5, IDGenRec) and conduct experiments on standard public datasets such as Amazon and Yelp.

To evaluate scalability, we measure both training and inference runtimes under the same experimental environment using the same backbone. As summarized in Table 10, CoFiRec introduces minimal overhead compared to TIGER across both the tokenization and generation stages. The additional hierarchical modules incur only slight increases in computation time during tokenization, while generation remains nearly identical in efficiency.

In terms of memory usage, CoFiRec remains highly efficient: the tokenization stage requires less than 1.8 GB of GPU memory, and generation under 4 GB when fine-tuning the same T5 backbone—comparable to TIGER. The main computational cost still lies in downstream fine-tuning of large language models, which is a shared characteristic across all generative recommender systems.