GRAM: Generative Recommendation via Semantic-aware Multi-granular Late Fusion

Sunkyung Lee[1], **Minjin Choi**[2], **Eunseong Choi**[1], **Hye-young Kim**[1], **Jongwuk Lee**[1]*
[1]Sungkyunkwan University, Republic of Korea, [2]Samsung Research, Republic of Korea
[1]{sk1027, eunseong, khyaa3966, jongwuklee}@skku.edu, [2]min_jin.choi@samsung.com

Abstract

Generative recommendation is an emerging paradigm that leverages the extensive knowledge of large language models by formulating recommendations into a text-to-text generation task. However, existing studies face two key limitations in (i) incorporating implicit item relationships and (ii) utilizing rich yet lengthy item information. To address these challenges, we propose a ***G**enerative **R**ecommender via semantic-**A**ware **M**ulti-granular late fusion (**GRAM**)*, introducing two synergistic innovations. First, we design *semantic-to-lexical translation* to encode implicit hierarchical and collaborative item relationships into the vocabulary space of LLMs. Second, we present *multi-granular late fusion* to integrate rich semantics efficiently with minimal information loss. It employs separate encoders for multi-granular prompts, delaying the fusion until the decoding stage. Experiments on four benchmark datasets show that GRAM outperforms eight state-of-the-art generative recommendation models, achieving significant improvements of 11.5–16.0% in Recall@5 and 5.3–13.6% in NDCG@5. The source code is available at https://github.com/skleee/GRAM.

1 Introduction

Generative recommendation has marked a pivotal shift in recommendation systems, driven by recent advances in large language models (LLMs) (Devlin et al., 2019; Brown et al., 2020). While the traditional recommendation approach focuses on matching user and item embeddings within a ranking framework (Kang and McAuley, 2018; Zhou et al., 2020), generative recommendation formulates it as a text-to-text generation task (Rajput et al., 2023; Geng et al., 2022). It aims to directly generate an *item identifier (ID)* based on the user's historical item sequence.

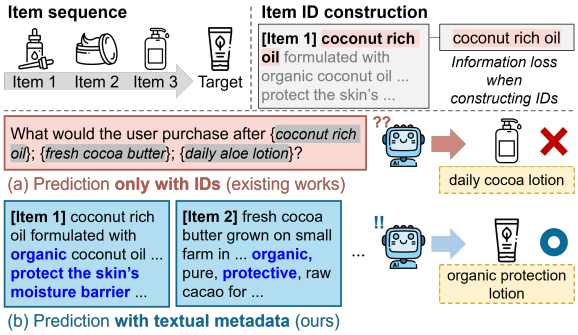

Figure 1: Illustration of our motivation. While (a) existing works rely solely on item IDs for prediction, (b) GRAM directly leverages rich textual metadata during prediction, enabling more accurate recommendations.

A key factor in generative recommendation lies in how well LLMs understand and effectively utilize each item. Existing works (Geng et al., 2022; Tan et al., 2024; Zheng et al., 2024) primarily use rich item metadata to construct abbreviated item IDs, leading to a potential loss of valuable details (Figure 1a). This limitation motivates us to incorporate item information throughout the entire recommendation process (Figure 1b).

However, two significant challenges hinder LLMs from effectively understanding and utilizing item information.

(i) *How do we enable LLMs to capture implicit item relationships?* While LLMs excel at understanding general language semantics, they struggle with recommendation-specific semantics, *e.g.*, implicit relationships between items (Zheng et al., 2024). This challenge manifests in two key aspects. *Hierarchical semantics*: Representing the conceptual hierarchy among items (*e.g.*, product taxonomy) is important for accurate and consistent recommendations. Without explicitly encoding the hierarchy in item IDs (Figure 2a), autoregressive generation can lead to semantically inconsistent recommendations, associating unrelated items ('*lipstick*' with '*soap*') based on superficial token-level

* Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 33294–33312
July 27 - August 1, 2025

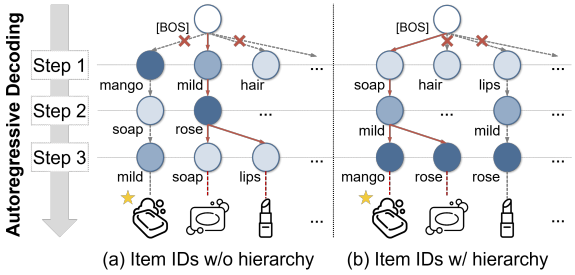

Figure 2: Illustration of the hierarchy when autoregressively decoding IDs. Darker shades represent more fine-grained information, and targets are marked with stars.

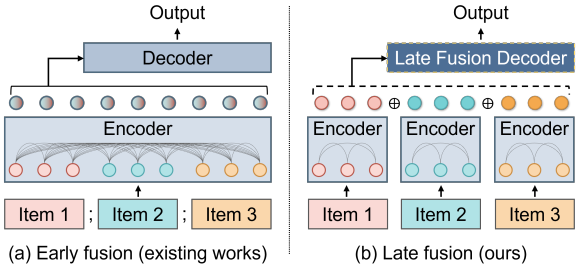

Figure 3: Schematic diagrams of fusion types. 'Item' indicates the textual information. ; and $\oplus$ denote the concatenation of text and hidden representations.

similarities. In contrast, hierarchical item IDs (Figure 2b) enable semantically coherent recommendations by leveraging shared concepts or attributes. Existing methods using predefined categories (Hua et al., 2023) or quantization (Rajput et al., 2023) for hierarchical IDs often fail to distinguish similar items or introduce out-of-vocabulary tokens, limiting the LLM's use of pre-trained knowledge.

Collaborative semantics: Capturing complex user-item interaction patterns is also critical (Zheng et al., 2024). These collaborative filtering patterns cannot be inferred from a single user sequence alone. While recent work (Zheng et al., 2024) attempts to address this through additional training tasks, it requires extensive fine-tuning to align newly defined IDs with language semantics.

(ii) *How do we handle lengthy item information?* Since items contain rich yet lengthy textual information (*e.g.*, product titles, categories, and descriptions), representing a user history as a sequence of detailed item information leads to excessively long sequences (Li et al., 2024c).[1] This poses significant computational challenges due to the quadratic complexity inherent in Transformer-based models. To avoid this, existing studies use partial attributes (Lin et al., 2024) or extract keywords (Tan et al., 2024), inevitably losing information.

To address these challenges, we propose a ***G**enerative **R**ecommender via semantic-**A**ware **M**ulti-granular late fusion (**GRAM**)*, which unlocks the capabilities of LLMs with two key components designed to work synergistically.

(i) **Semantic-to-lexical translation**. To enable LLM to capture implicit item relationships, we encode item relationships into textual representations prior to training. First, *hierarchical semantics indexing* iteratively clusters the item embeddings based on semantic similarity and maps them into LLM's vocabulary space to generate hierarchical textual IDs. Next, *collaborative semantics verbalization* incorporates collaborative semantics by leveraging a collaborative filtering model. For each item, we identify top-k similar items and express them in a textual format using the item IDs.

(ii) **Multi-granular late fusion**. To effectively handle rich yet lengthy item metadata, we process user history as multiple prompts with different granularities: coarse-grained user prompts for holistic user preferences and fine-grained item prompts for detailed item attributes. Subsequently, multi-granular prompts are efficiently integrated via late fusion. Unlike early fusion (Figure 3a), which suffers from quadratic complexity due to concatenated texts at the input level, late fusion (Figure 3b) delays integration until the decoding stage. It aligns with successful techniques in other domains (Izacard and Grave, 2021; Ye et al., 2023). Most importantly, late fusion maximizes the effectiveness of the semantic-to-lexical translation by preserving rich semantics and enabling the processing of significantly longer inputs with minimal information loss.

Our contributions are summarized as follows:

(i) Item relationship modeling: We design *semantic-to-lexical translation* to represent item relationships in the vocabulary space of LLMs.

(ii) Model architecture: Our *multi-granular late fusion* effectively leverages rich textual item information without expensive computational overhead.

(iii) Extensive experiments: GRAM achieves up to 16.0% and 13.6% gains in Recall@5 and NDCG@5 over state-of-the-art generative recommenders across four benchmark datasets.

[1]For the Amazon Beauty dataset (McAuley et al., 2015), user sequences consist of 1,440 tokens on average when represented as a simple concatenation of item texts using the T5 tokenizer. Only 4.9% of sequences have less than 512 tokens.

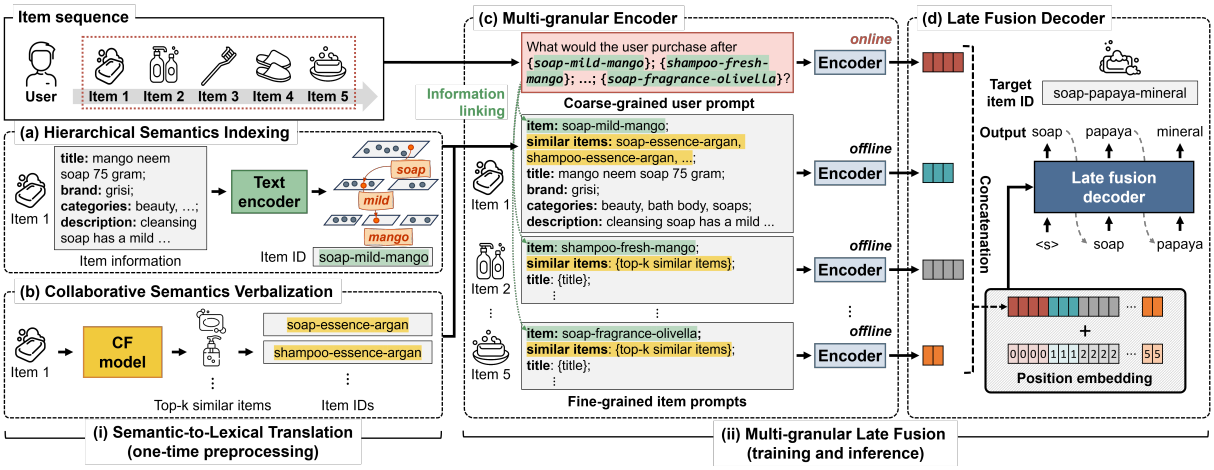

Figure 4: Overall architecture of GRAM. (i) We represent hierarchical and collaborative semantics in textual forms via *semantic-to-lexical translation* before training. (ii) The user/item prompts are constructed and encoded separately. The target item ID is inferred via *multi-granular late fusion*, directly leveraging rich textual information.

2 Related Work

2.1 Sequential Recommendation

It aims to predict the next item based on the user's historical item interactions (Wang et al., 2019). To capture user preferences, item sequences have been modeled with various neural encoders, *e.g.*, RNNs (Hidasi et al., 2016), GNNs (Wu et al., 2019), and Transformers (Kang and McAuley, 2018; Sun et al., 2019; Ma et al., 2019). Recently, several studies (Zhang et al., 2019; Zhou et al., 2020) have used metadata to model item dependency and user-item interactions. However, they express attributes as discrete IDs, neglecting to exploit *textual metadata* with LLMs.

2.2 Generative Recommendation

Given an item sequence, it generates a *target item identifier*.[2] Existing studies can be categorized depending on the type of item identifiers.

Numeric IDs[3]. P5 (Geng et al., 2022; Hua et al., 2023) adopts numeric IDs and multi-task learning. TIGER (Rajput et al., 2023) and LC-Rec (Zheng et al., 2024) construct codebooks with vector quantization, *i.e.*, RQ-VAE (Zeghidour et al., 2022). LC-Rec further adopted alignment tasks for language and collaborative semantics. LETTER (Wang et al., 2024a) improved vector quantization by integrating collaborative signals. Recently, ELMRec (Wang et al., 2024b) incorporates high-order relationships from the graph. However, since numeric IDs are separated from the vocabulary of LLMs, they suffer from a semantic gap that hinders the full potential of LLMs for recommendations.

Model	Textual metadata usage: ID construction	Textual metadata usage: Prediction	Item ID: Text	Item ID: Hierarchy
P5-SemID	✓	✗	✗	✓
TIGER	✓	✗	✗	✓
IDGenRec	✓	✗	✓	✗
LETTER	✓	✗	✗	✓
ELMRec	✗	✗	✗	✗
LC-Rec	✓	✗	✗	✓
GRAM	✓	✓	✓	✓

Table 1: Comparison of different generative recommendation models on (i) how textual metadata is utilized and (ii) how item IDs are constructed.

Textual IDs. Another line of research has explored textual IDs as a meaningful alternative to numeric IDs. IDGenRec (Tan et al., 2024) generates semantic item IDs from text metadata using an ID generator. TransRec (Lin et al., 2024) combines both numeric and textual IDs in a hybrid approach. However, none of them explore how to (i) directly utilize textual metadata during prediction and (ii) incorporate the item hierarchy into the textual identifiers. Further comparison is presented in Table 1.

3 Proposed Method

We present a ***G**enerative **R**ecommender via semantic-**A**ware **M**ulti-granular late fusion (**GRAM**)*, as depicted in Figure 4. GRAM seamlessly incorporates item relationships (Section 3.2) and fully utilizes rich item information (Section 3.3). We lastly explain the training and inference processes of GRAM (Section 3.4).

[2]For reranking (Gao et al., 2023) or discriminative methods (Hou et al., 2022), refer to Appendix A.

[3]While Wang et al. (2024a); Li et al. (2024c) distinguish codebook and numeric IDs, we group them as numeric IDs.

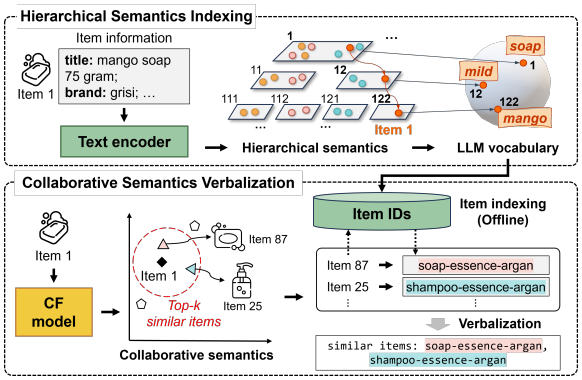

Figure 5: Illustration of semantic-to-lexical translation. Each item is assigned textual IDs based on *hierarchical semantics*, and *collaborative semantics* are verbalized based on the hierarchical IDs.

3.1 Task Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ denote a set of users and items. For each user $u \in \mathcal{U}$, we represent the interaction history as a chronological item sequence $s_u = (i_1, \ldots, i_{|s|})$, where i_t is the item at the t-th position, and $|s|$ indicates the number of items in the sequence s. The goal is to predict the next item $i_{|s|+1}$ that the user is most likely to interact with based on the user's interaction history.

Each item $i \in \mathcal{I}$ is assigned to the unique ID $\tilde{i}$. Each user's item sequence s_u is converted to the sequence of item IDs, *i.e.*, $\tilde{s}_u = (\tilde{i}_1, \tilde{i}_2, \ldots, \tilde{i}_{|s|})$. Given the sequence $\tilde{s}_u$, the generative recommendation is formulated by generating the next item ID $\tilde{i}_{|s|+1}$, which the user is most likely to prefer.

Each item consists of multiple attributes, denoted by $(a_1, \ldots, a_m)$, where m is the number of attributes. Each attribute indexed by j is presented in a key-value format $a_j = (k_j, v_j)$, where k_j is the attribute name a_j, *e.g.*, "*title*," "*brand*," or "*description*," and v_j represents the corresponding attribute value. We represent item text by combining the attributes, *e.g.*, "*title: coastal scents cocoa butter; brand: coastal scents; description: ...*".

3.2 Semantic-to-Lexical Translation

We introduce semantic-to-lexical translation, representing implicit item relationships in a textual form. Figure 5 depicts *hierarchical semantics indexing* that transforms the item hierarchy into textual IDs, and *collaborative semantics verbalization* that represents collaborative signals as extra attributes using textual IDs. Note that these are preprocessing steps performed only once before training.

3.2.1 Hierarchical Semantics Indexing

We present a novel method for constructing hierarchical textual IDs, offering three key advantages: (i) utilizing LLMs' knowledge through natural language tokens, (ii) capturing semantic relationships among items through shared identifier prefixes, and (iii) enabling the progressive generation of identifiers from general to specific item characteristics during autoregressive decoding (Tay et al., 2022).

Hierarchical Semantics Extraction. We employ hierarchical k-means clustering over item embeddings to construct hierarchical identifiers where semantically similar items have identical prefixes. Each item embedding $\mathbf{z} \in \mathbb{R}^e$ is obtained using the text encoder. The clustering process begins by partitioning all items into k clusters. For clusters with more than c items, we recursively apply the k-means clustering to further divide it into sub-clusters. The recursive clustering process terminates when the cluster size is smaller than c or the maximum depth l is reached. Finally, each item is assigned a sequence of cluster indices, where the sequence length is bounded by l.

Hierarchical Semantics Translation. To translate hierarchical semantics into LLM vocabulary, we assign representative tokens to each cluster. GRAM deliberately utilizes existing tokens, unlike previous studies (Rajput et al., 2023; Zheng et al., 2024) that employ out-of-vocabulary tokens. That is, we preserve the hierarchical structure and minimize potential conflicts with language semantics. We convert each text of item i into a $|V|$-dimensional vocabulary space vector $\mathbf{V}_i$. For simplicity, we use the TF-IDF (Jones, 2004) scoring schema. Note that more sophisticated scoring functions (Formal et al., 2021) can be used. We then create a cluster-level vocabulary vector by averaging vectors for all items within a cluster. Lastly, we select the most representative token, *i.e.*, the token with the highest score, from the cluster-level vector. For items whose cluster indices length is shorter than l, we append additional tokens using the vocabulary vector $\mathbf{V}_i$ to ensure a length of l. For duplicate IDs, we append an additional digit for uniqueness. (See the detailed algorithm in Appendix D.)

3.2.2 Collaborative Semantics Verbalization

We integrate collaborative semantics into LLM-based recommenders to complement LLM's capabilities. While LLMs excel at processing textual semantics, they still struggle to incorporate collaborative patterns across items (Zheng et al., 2024;

Kim et al., 2024). To bridge collaborative semantics to LLMs, we extract collaborative signals and convert them as an additional item attribute.

Collaborative Semantics Extraction. We employ the off-the-shelf collaborative filtering (CF) model[4] to capture item relationships through learned embeddings. We select the top-k most similar items for each item i:

$$i_1^{CF}, \ldots, i_k^{CF} = \texttt{argTop-}k_{j \in \mathcal{I}} \text{sim}(\mathbf{e}_i, \mathbf{e}_j), \quad (1)$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ represent item embedding vectors obtained from the off-the-shelf CF model. The similarity is computed by a function $\text{sim}(\cdot, \cdot)$, *e.g.*, dot product. The $\texttt{argTop-}k(\cdot)$ function returns the indices of k items with the highest similarity scores.

Collaborative Semantics Translation. We then transform the collaborative knowledge into a text format using hierarchical item IDs:

$$a_{CF} = (k_{CF}, v_{CF}), \text{ where } v_{CF} = [\tilde{i}_1^{CF}, \ldots, \tilde{i}_k^{CF}]. \quad (2)$$

Here, k_{CF} represents the key '*similar items*', and v_{CF} is the verbalized similar items, *e.g.*, "*soap-essence-argan, shampoo-essence-argan, ...*". It allows us to incorporate collaborative signals when using LLMs. It is also well aligned with existing findings (Yao et al., 2023; Zhang et al., 2024).

3.3 Multi-granular Late Fusion

To effectively leverage rich but lengthy item information, we introduce *multi-granular late fusion*. We process inputs into coarse-grained user prompts and fine-grained item prompts with *multi-granular encoder*. The *late fusion decoder* then integrates them while preserving detailed information during the prediction. By delaying fusion until the decoder, it avoids the quadratic complexity and achieves enhanced efficiency, as evidenced by the theoretical and empirical analysis in Appendix C.

Importantly, multi-granular late fusion synergistically leverages the item relationships from semantic-to-lexical translation. Hierarchical relationships in item IDs are incorporated into user prompts, while collaborative relationships in item attributes are integrated into item prompts. This design enables GRAM to fully exploit the benefits of semantic-to-lexical translation.

3.3.1 Multi-granular Encoder

To capture user preference, existing methods (Geng et al., 2022; Tan et al., 2024) combine item IDs with prompt templates. While the item ID sequence provides a holistic view of user behavior and explicitly captures sequential dependencies, it inevitably loses item details. Conversely, using full text preserves details but leads to lengthy inputs. Thus, we adopt two complementary prompts to exploit the distinctive benefits, as exemplified in Appendix B.

Coarse-grained User Prompt. We capture the overall user preferences through a concise representation of the user interaction history by concatenating item IDs as follows:

$$\tilde{s}_{seq} = [\tilde{i}_{|s|}; \tilde{i}_{|s|-1}; \ldots; \tilde{i}_1], \quad (3)$$

where ; is the concatenation of text. The user history is sorted in reverse order to prevent recent items from being truncated (Li et al., 2023a). To transform it into a natural language, we interpolate it into the placeholder of a predefined prompt:

$T_u =$ "`What would the user purchase after {`$\tilde{s}_{seq}$`}?`"

Fine-grained Item Prompt. We represent detailed item characteristics and relationships by leveraging comprehensive item attributes and collaborative semantics a_{CF}, which are obtained from Eq. (2). The item prompt T_i is constructed as follows:

$$T_i = (a_{ID}, a_{CF}, a_1, \ldots, a_m), \quad (4)$$

where $a_{ID} = (\text{'item:'}, \tilde{i}\,)$ represents the item identifier such as "*item: soap-mild-mango*." Notably, we append the IDs to link user prompts with their corresponding item prompts.

Prompt Encoding. The multi-granular user and item prompts are processed independently and represented as a list of prompts:

$$\mathcal{P} = (T_u, T_{i_{|s|}}, \ldots, T_{i_1}). \quad (5)$$

With one user prompt and $|s|$ item prompts, we encode each prompt $\mathcal{P}_j$ using the T5 encoder (Raffel et al., 2020) to obtain token embeddings $\mathbf{H}_j$:

$$\mathbf{H}_j = \text{Encoder}\,(\mathcal{P}_j) \in \mathbb{R}^{M \times d} \text{ for } j \in \{1, \ldots, |s|+1\}, \quad (6)$$

where M is the maximum text sequence length of the encoder, and d is the hidden dimension size.

It is non-trivial to connect items in the user prompt with their corresponding item prompts. We resolve it through *information linking*, where each item ID (a_{ID}) serves as a bridge between coarse- and fine-grained information. This linking effectively integrates information across the prompts.

[4]We utilized SASRec (Kang and McAuley, 2018).

3.3.2 Late Fusion Decoder

The decoder integrates the representations of user preferences and item information to generate recommendations. To explicitly encode the sequential information of user interactions, we incorporate position embedding $\mathbf{P} \in \mathbb{R}^{(L+1)\times d}$, where L is the maximum number of items in user history:

$$\mathbf{X}_j = \mathbf{H}_j + \mathbf{P}_j \text{ for } j \in \{1, \ldots, |s|+1\}. \quad (7)$$

We then combine all position-aware representations $\mathbf{X}_j$ into a unified embedding matrix $\mathbf{X}$:

$$\mathbf{X} = [\mathbf{X}_1; \ldots; \mathbf{X}_{|s|+1}] \in \mathbb{R}^{((|s|+1)\times M)\times d}. \quad (8)$$

Finally, the decoder leverages this comprehensive representation $\mathbf{X}$ as the key-value matrix in cross-attention to aggregate the rich textual information. The target item ID is generated autoregressively, considering both coarse-grained user preferences and fine-grained item attributes. The probability of generating a textual item ID $\tilde{i}$ is defined as:

$$P(\tilde{i}|\mathcal{P}) = \prod_{t=1}^{n} P(\tilde{i}^t|\mathcal{P}, \tilde{i}^{<t}). \quad (9)$$

3.4 Training and Inference

For training, the model learns to generate textual IDs by minimizing the sequence-to-sequence cross-entropy loss with teacher-forcing:

$$\mathcal{L} = -\sum_{t=1}^{n} \log P(\tilde{i}^t|\mathcal{P}, \tilde{i}^{<t}), \quad (10)$$

where $\tilde{i}^t$ is a t-th token of the target item ID $\tilde{i}_{|s|+1}$.

For inference, we adopt a two-stage process. In the offline stage (*i.e.*, pre-processing), we assign IDs to all items, obtain collaborative knowledge from the CF model, and pre-compute the encoder outputs for fine-grained item prompts. Then, during the online stage, we only encode the user prompt and generate the recommendations using constrained beam search. To generate valid IDs, we use a prefix tree Trie (Cormen et al., 2022) following Zheng et al. (2024); Lee et al. (2023). This two-stage approach significantly reduces computational overhead by processing fine-grained item information offline.

4 Experimental Setup

Datasets. We conduct experiments on four real-world datasets: Amazon review (McAuley et al., 2015; He and McAuley, 2016)[5] and Yelp[6]. Among the Amazon datasets, we select three subcategories: "Sports and Outdoors", "Beauty", and "Toys and Games". Following Hua et al. (2023), we remove users and items with fewer than five interactions (5-core setting). The statistics are in Appendix G.1.

Evaluation Protocols and Metrics. We employ the *leave-one-out* strategy to split the train, validation, and test sets following (Kang and McAuley, 2018; Zheng et al., 2024). For each user sequence, we use the last item as test data, the second last item as validation data, and the remaining items as training data. We conduct *full-ranking evaluations* on all items rather than on sampled items for an accurate assessment. For metrics, we adopt top-k Recall (R@k) and Normalized Discounted Cumulative Gain (N@k) with cutoff $k = \{5, 10\}$.

Baselines. We adopt six traditional sequential recommenders: **GRU4Rec** (Hidasi et al., 2016), **HGN** (Ma et al., 2019), **SASRec** (Kang and McAuley, 2018), **BERT4Rec** (Sun et al., 2019), **FDSA** (Zhang et al., 2019), and **S^3Rec** (Zhou et al., 2020). We adopt eight state-of-the-art generative recommenders: **P5-SID**, **P5-CID**, **P5-SemID** (Hua et al., 2023), **TIGER** (Rajput et al., 2023), **IDGenRec** (Tan et al., 2024), **LETTER** (Wang et al., 2024a), **ELMRec** (Wang et al., 2024b), and **LCRec** (Zheng et al., 2024). All results are averaged over three runs with different seeds. Further details are provided in Appendix G.3.

Implementation Details. We implemented GRAM on OpenP5 (Xu et al., 2024). The model is initialized with T5-small (Raffel et al., 2020) to be consistent with existing works (Hua et al., 2023; Wang et al., 2024b; Tan et al., 2024). The model was trained with the Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.001 and a linear scheduler with a warm-up ratio of 0.05. We set the maximum text length to 128 and the batch size to 128. For hierarchical IDs, l and c are tuned among $\{5, 7, 9\}$ and $\{32, 128, 512\}$. We tuned k for collaborative semantics in $\{5, 10, 20\}$. We use a constrained beam search with a beam size of 50. The maximum number of items in sequence was set to 20 following Zheng et al. (2024). More details are in Appendix G.4. For fair comparison, we carefully modified some experimental settings in baselines (Wang et al., 2024b; Tan et al., 2024), with details provided in Appendix E and F.

[5] https://jmcauley.ucsd.edu/data/amazon/
[6] https://www.yelp.com/dataset

Model	Beauty				Toys				Sports				Yelp			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Traditional recommendation models																
GRU4Rec	0.0429	0.0288	0.0643	0.0357	0.0371	0.0254	0.0549	0.0311	0.0237	0.0154	0.0373	0.0197	0.0240	0.0157	0.0398	0.0207
HGN	0.0350	0.0217	0.0589	0.0294	0.0345	0.0212	0.0553	0.0279	0.0203	0.0127	0.0340	0.0171	0.0366	0.0250	0.0532	0.0304
SASRec	0.0323	0.0200	0.0475	0.0249	0.0339	0.0208	0.0442	0.0241	0.0147	0.0089	0.0220	0.0113	0.0284	0.0214	0.0353	0.0245
BERT4Rec	0.0267	0.0165	0.0450	0.0224	0.0210	0.0131	0.0355	0.0178	0.0136	0.0085	0.0233	0.0116	0.0244	0.0159	0.0401	0.0210
FDSA	0.0570	0.0412	0.0777	0.0478	0.0619	0.0455	0.0805	0.0514	0.0283	0.0201	0.0399	0.0238	0.0331	0.0218	0.0534	0.0284
S^3Rec	0.0377	0.0235	0.0627	0.0315	0.0365	0.0231	0.0592	0.0304	0.0229	0.0145	0.0370	0.0190	0.0190	0.0117	0.0321	0.0159
Generative recommendation models																
P5-SID	0.0465	0.0329	0.0638	0.0384	0.0216	0.0151	0.0325	0.0186	0.0295	0.0212	0.0403	0.0247	0.0299	0.0211	0.0432	0.0253
P5-CID	0.0465	0.0325	0.0668	0.0391	0.0223	0.0143	0.0357	0.0186	0.0295	0.0214	0.0420	0.0254	0.0226	0.0155	0.0363	0.0199
P5-SemID	0.0459	0.0327	0.0667	0.0394	0.0264	0.0178	0.0416	0.0227	0.0336	0.0243	0.0481	0.0290	0.0212	0.0143	0.0329	0.0181
TIGER	0.0352	0.0236	0.0533	0.0294	0.0274	0.0174	0.0438	0.0227	0.0176	0.0143	0.0311	0.0146	0.0164	0.0103	0.0262	0.0135
IDGenRec†	0.0463	0.0328	0.0665	0.0393	0.0462	0.0323	0.0651	0.0383	0.0273	0.0186	0.0403	0.0228	0.0310	0.0219	0.0448	0.0263
LETTER	0.0364	0.0243	0.0560	0.0306	0.0309	0.0296	0.0493	0.0262	0.0209	0.0136	0.0331	0.0176	0.0298	0.0218	0.0403	0.0252
ELMRec†	0.0372	0.0267	0.0506	0.0310	0.0148	0.0119	0.0193	0.0131	0.0241	0.0181	0.0307	0.0203	0.0424	0.0301	0.0501	0.0324
LC-Rec	0.0503	0.0352	0.0715	0.0420	0.0543	0.0385	0.0753	0.0453	0.0259	0.0175	0.0384	0.0216	0.0341	0.0235	0.0501	0.0286
GRAM	**0.0641**	**0.0451**	**0.0890**	**0.0531**	**0.0718**	**0.0516**	**0.0987**	**0.0603**	**0.0375**	**0.0256**	**0.0554**	**0.0314**	**0.0476**	**0.0326**	**0.0698**	**0.0397**
Gain (%)	12.4*	9.5*	14.5*	11.0*	16.0*	13.6*	22.7*	17.1*	11.5*	5.3*	15.2*	8.3*	12.3*	8.1*	30.7*	22.5*

Table 2: Overall performance comparison. The best model is marked in **bold**, and the second-best model is underlined. Gain measures improvement of the proposed method over the best competitive baseline. '$*$' indicates statistical significance ($p < 0.05$) by a paired t-test. '†' indicates models where our reproduced results differ from the original papers due to corrected experimental settings. Please refer to Appendix E and F for details. Efficiency analysis is in Appendix C. Additional results with a cutoff of 20 are in Appendix H.5.

Model	Beauty		Toys	
	R@5	N@5	R@5	N@5
GRAM	**0.0641**	**0.0451**	**0.0718**	**0.0516**
w/o hierarchy	0.0605	0.0438	0.0630	0.0466
w/o CF (a_{CF})	0.0567	0.0396	0.0589	0.0406
w/o user prompt (T_u)	0.0634	0.0443	0.0709	0.0510
w/o item prompt (T_i)	0.0582	0.0404	0.0574	0.0397
w/o linking (a_{ID})	0.0628	0.0441	0.0702	0.0507
w/o position ($\mathbf{P}$)	0.0563	0.0395	0.0665	0.0465

Table 3: Ablation study of GRAM. We examined the effect of (i) semantic-to-lexical translation, (ii) the multi-granular prompts, and (iii) additional techniques.

5 Experimental Results

5.1 Overall Performance

We thoroughly evaluate GRAM's effectiveness on four real-world datasets, as presented in Table 2. Our key findings are as follows.

(i) GRAM consistently outperforms the state-of-the-art models, achieving up to 16.0% and 13.6% improvement in R@5 and N@5. Compared to the best generative models (LC-Rec and IDGenRec), GRAM shows remarkable gains of up to 32.3% and 34.1% in R@5 and N@5. It indicates the effectiveness of capturing user preferences through item relationships and rich item information.

(ii) The generative recommendation models incorporating hierarchical item relationships into IDs (P5-SemID, LC-Rec, and GRAM) demonstrate strong performance, highlighting the importance of hierarchical structures in recommendation. GRAM enhances this advantage by mapping these relationships to LLM vocabulary tokens, enabling better utilization of pre-trained language understanding capabilities.

(iii) GRAM consistently outperforms IDGenRec using textual IDs, yielding average gains of 46.3% in R@5 and 45.9% in N@5. This is due to the inherent limitation of concise item IDs, which compress item information and lead to information loss. In contrast, GRAM's late fusion approach preserves comprehensive information by delaying information aggregation until the decoder.

5.2 Ablation Study

We analyze the contributions of key components in GRAM, as shown in Table 3 (See also Appendix H.3 and H.4 for additional results).

Semantic-to-Lexical Translation. Both hierarchical and collaborative semantics yield 27.2% and 10.8% improvement in N@5, respectively. 'w/o hierarchy' denotes selecting representative tokens from each item text without hierarchy. Hierarchical clustering is particularly effective, demonstrating the benefits of considering item relationships beyond simple vocabulary tokens. Our analysis also

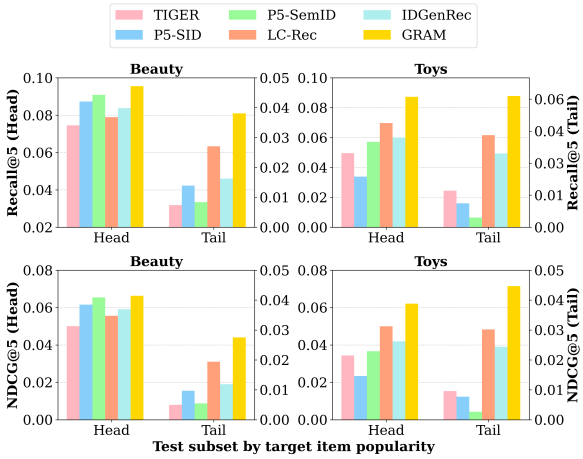

Figure 6: Performance of generative recommendation models depending on the popularity of target items.

ID type	Beauty		Toys	
	R@5	N@5	R@5	N@5
Hierarchical ID	**0.0641**	**0.0451**	**0.0718**	**0.0516**
Title ID	0.0478	0.0342	0.0564	0.0412
Category ID	0.0512	0.0367	0.0465	0.0350
Keyword ID	0.0605	0.0438	0.0630	0.0466
RQ-VAE ID	0.0605	0.0432	0.0662	0.0477

Table 4: Performance of GRAM over various IDs.

reveals that collaborative patterns are successfully integrated into language semantics. We also investigate the generalizability of semantic-to-lexical translation in Appendix H.2.

Multi-granular Prompts. Both user prompts and item prompts contribute to accuracy, improving N@5 by up to 1.2% and 30.1%, respectively. The substantial gain from item prompts highlights the role of expressing detailed information. User prompts are particularly effective in preserving sequential information, yielding higher gains for longer sequences, as evidenced in Appendix H.1.

Additional Techniques. Information linking contributes to seamless late fusion, boosting N@5 by up to 1.8%. The multi-granular information is bridged beyond the length barrier of texts with the simple technique. The position embedding improves N@5 by up to 13.2%, making LLMs distinguish the sequential order of items well during late fusion. It exhibits that the item orders in a sequence play a pivotal role in predicting the next items.

5.3 In-depth Analysis

Effect of Hierarchical ID. Table 4 shows the effectiveness of hierarchical IDs compared to various

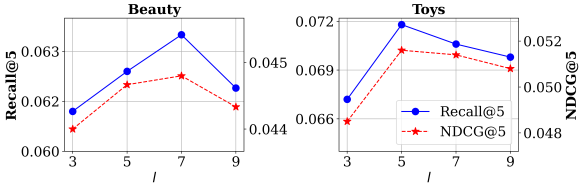

Figure 7: Performance of GRAM over varying length of identifiers l.

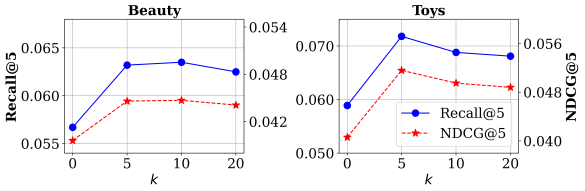

Figure 8: Performance of GRAM over varying number of the top-k similar items in Eq. (1).

IDs: Title ID, Category ID[7], Keyword ID (extracting keywords from item metadata without clustering), and RQ-VAE ID (Rajput et al., 2023; Zheng et al., 2024). (i) Hierarchical ID outperforms Title and Category ID by up to 31.7% and 47.7% in N@5, showing raw metadata lacks sufficient granularity. (ii) Compared to Keyword ID, hierarchical ID shows up to 10.8% gains in N@5, highlighting the benefits of capturing hierarchical relationships. (iii) Hierarchical ID improves N@5 by 8.2% over RQ-VAE ID, demonstrating the benefits of using LLM vocabulary instead of newly defined tokens that may create semantic gaps.

Performance on Head/Tail Items. As shown in Figure 6, we analyze the performance of GRAM and generative models depending on the popularity of target items by splitting the entire user sequence into Head and Tail.[8] GRAM exhibits gains up to 42.6% and 47.8% in R@5 and N@5 for tail items compared to the best competitive method, *i.e.*, LC-Rec. GRAM also improves performance in Head groups, boosting the performance by up to 25.3% and 24.2% in R@5 and N@5.

Hyperparameter Sensitivity. Figures 7 and 8 show the accuracy of GRAM over varying ID length l and the number of the top-k items in Eq. (1). The optimal values for (l, k) are (5, 7) and (10, 5) for the Beauty and Toys, respectively. Collaborative semantics improves N@5 by up to 27.2%. However, providing too many similar items introduces noise

[7]For items with the same categories or titles, we append additional digits to ensure uniqueness.

[8]Head and Tail denote user groups where the target item is in the top 20% and bottom 80% of popularity, respectively. Refer to Appendix G.2 for detailed statistics.

and degrades performance. An additional analysis of the cluster size c is in Appendix H.6.

6 Conclusion

We present GRAM, a novel generative recommendation model that addresses fundamental challenges in leveraging LLMs for recommendation with two key innovations: (i) semantic-to-lexical translation for bridging complex item relationships with LLMs and (ii) multi-granular late fusion for efficient and effective processing of rich item information. Our extensive experimental results across four real-world benchmark datasets validate the superiority of GRAM over existing sequential recommendation models, showing up to 16.0% and 13.6% gains in R@5 and N@5, respectively.

7 Limitations

While GRAM demonstrates strong performance in generative recommendation, we carefully list limitations as follows.

Vocabulary Selection Method. For hierarchical semantics translation, we rely on TF-IDF scoring to select representative tokens from LLM's vocabulary space. While this provides a simple solution and serves as an efficient proof-of-concept, we conjecture that more sophisticated techniques such as well-designed neural sparse retrieval methods (Formal et al., 2021, 2022; Choi et al., 2022) may potentially yield better representative tokens. Future work could explore integrating such approaches to improve the quality of hierarchical semantics translation while maintaining the benefits of using LLM's vocabulary.

Language Model Capacity. We leverage LLMs' vocabulary and language understanding capabilities. While we demonstrate strong results using T5-small with 60M parameters and T5-base with 220M parameters as our encoder-decoder model, scaling up to a larger model (*e.g.*, FLAN-T5-XL, T5-11B) could potentially yield even better performance, which we leave as future work.

Ethics Statement

This work fully complies with the ACL's ethical guidelines. The scientific artifacts we have utilized are available for research under liberal licenses, and the utilization of these tools is consistent with their intended applications.

Acknowledgments

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2019-II190421, RS-2022-II220680, RS-2025-00564083, IITP-2025-RS-2024-00360227).

References

Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. 2024. Decoding matters: Addressing amplification bias and homogeneity issue for llm-based recommendation. In *EMNLP*, pages 10540–10552.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, pages 1007–1014.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901.

Yongjun Chen, Zhiwei Liu, Jia Li, Julian J. McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *WWW*, pages 2172–2182.

Eunseong Choi, Sunkyung Lee, Minjin Choi, Hyeseon Ko, Young-In Song, and Jongwuk Lee. 2022. Spade: Improving sparse representations using a dual document encoder for first-stage retrieval. In *CIKM*, pages 272–282.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *RecSys*, pages 1126–1132.

Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit Sanghai, Fei Sha, and William W. Cohen. 2023. Fido: Fusion-in-decoder optimized for stronger performance and faster inference. In *Findings of the ACL*, pages 11534–11547.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of

deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural IR models more effective. In *SIGIR*, pages 2353–2359.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: sparse lexical and expansion model for first stage ranking. In *SIGIR*, pages 2288–2292.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *CoRR*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *RecSys*, pages 299–315.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

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

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *KDD*, pages 585–593.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. In *SIGIR-AP*, pages 195–204.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880.

Karen Sparck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Min-Chul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *KDD*, pages 1395–1406.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *CoRR*, abs/2405.17428.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. 2023. GLEN: generative retrieval via lexical index learning. In *EMNLP*, pages 7693–7704.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian J. McAuley. 2023a. Text is all you need: Learning language representations for sequential recommendation. In *KDD*, pages 1258–1267.

Lei Li, Yongfeng Zhang, and Li Chen. 2023b. Prompt distillation for efficient llm-based recommendation. In *CIKM*, pages 1348–1357.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2024a. Large language models for generative recommendation: A survey and visionary discussions. In *LREC-COLING*, pages 10146–10159.

Yaoyiran Li, Xiang Zhai, Moustafa Alzantot, Keyi Yu, Ivan Vulic, Anna Korhonen, and Mohamed Hammad. 2024b. Calrec: Contrastive alignment of generative llms for sequential recommendation. In *RecSys*, pages 422–432.

Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. 2024c. A survey of generative search and recommendation in the era of large language models. *CoRR*.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *SIGIR*, pages 1785–1795.

Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Bridging items and language: A transition paradigm for large language model-based recommendation. In *KDD*, pages 1816–1826.

Zhenghao Liu, Sen Mei, Chenyan Xiong, Xiaohua Li, Shi Yu, Zhiyuan Liu, Yu Gu, and Ge Yu. 2023. Text matching improves sequential recommendation by reducing popularity biases. In *CIKM*, pages 1534–1544.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*, pages 825–833.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from

pre-trained text-to-text models. In *Findings of the ACL*, pages 1864–1874.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. 2023. Recommender systems with generative retrieval. In *NeurIPS*, pages 10299–10315.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Idgenrec: Llm-recsys alignment with textual id learning. In *SIGIR*, page 355–364.

Zuoli Tang, Lin Wang, Lixin Zou, Xiaolu Zhang, Jun Zhou, and Chenliang Li. 2023. Towards multi-interest pre-training with sparse capsule network. In *SIGIR*, pages 311–320.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *NeurIPS*, pages 21831–21843.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. 2019. Sequential recommender systems: Challenges, progress and prospects. In *IJCAI*, pages 6332–6338.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024a. Learnable item tokenization for generative recommendation. In *CIKM*, pages 2400–2409.

Xinfeng Wang, Jin Cui, Fumiyo Fukumoto, and Yoshimi Suzuki. 2024b. Enhancing high-order interaction awareness in llm-based recommender model. In *EMNLP*, pages 11696–11711.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A survey on large language models for recommendation. *CoRR*.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, pages 346–353.

Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a more user-friendly and easy-to-use benchmark library for recommender systems. In *SIGIR*, pages 2837–2847.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024. Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems. In *SIGIR*, pages 386–394.

Jing Yao, Wei Xu, Jianxun Lian, Xiting Wang, Xiaoyuan Yi, and Xing Xie. 2023. Knowledge plugins: Enhancing large language models for domain-specific recommendations. *CoRR*.

Qinyuan Ye, Iz Beltagy, Matthew E. Peters, Xiang Ren, and Hannaneh Hajishirzi. 2023. Fid-icl: A fusion-in-decoder approach for efficient in-context learning. In *ACL*, pages 8158–8185.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2022. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:495–507.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326.

Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Text-like encoding of collaborative information in large language models for recommendation. In *ACL*, pages 9181–9191.

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

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *ICDE*, pages 1435–1448.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, pages 1893–1902.

Prompt example in Beauty dataset

Coarse-grained user prompt T_u

What would the user purchase after ... `soap-salt-sea-dead-genuine-minerals-bars` ; `serum-ovi-nutrients-sea-noi-feed-ains` ?

Fine-grained item prompt T_{i_2}

item: `serum-ovi-nutrients-sea-noi-feed-ains` ; **similar items**: soap-salt-sea-dead-genuine-minerals-bar, gan-limited-33-ar-lix-treatment-eed, eye-dealing-limited-33-suitable-amp-youth, ...; **title**: adovia facial serum anti-aging, skin lifting, facial serum with vitamin c, dead sea salt and green tea get firmer, more radiant looking skin nonoily deep moisturizer delivers minerals amp; nutrients deep into skin satisfaction guaranteed; **brand**: na; **categories**: beauty, skincare, face, oils serums; **description**: also contains seaweed for a rich infusion of nutrients to feed and nourish your skin; **price**: 50.0; **salesrank**: beauty: 43112

Fine-grained item prompt T_{i_1}

item: `soap-salt-sea-dead-genuine-minerals-bar` ; **similar items**: serum-ovi-nutrients-sea-noi-feed-ains, eye-dealing-limited-33-suitable-amp-youth, gan-limited-33-ar-lix-treatment-eed, ...; **title**: dead sea salt deep hair conditioner for dry or damaged hair great for natural curly hair for men and women deeply hydrating and nourishing on scalp amp; hair helps to reduce dandruff and dry scalp made with pure dead sea salt and minerals, chamomile, vitamin e amp; natural aloe vera leaves your hair looking amp; feeling healthy, silky and hydrated; **brand**: na; **categories**: beauty, hair care, conditioners; **description**: made with real dead sea salt 100 pure and genuine for an infusion of 21 minerals essential to proper skin and hair function. leaves hair looking hydrated, healthy and silky. helps to reduce the appearance of dandruff and dry, flaky scalp. aloe vera deeply nourishes the scalp and hair hydrating it from within. dead sea salt contains more than 21 skin and hair rebuilding minerals such as magnesium, calcium, sulfur, bromide, iodine, sodium, zinc and potassium. we sell out fast get it soon before we run out again. we try to produce this item as fast as we can.; **price**: 19.0; **salesrank**: beauty: 7330

...

Figure 9: Example of multi-granular prompts on the Beauty dataset for the user `A3GPKDC4PQXKFR`.

A Additional Related Works

LLM-based Recommendation. Recent studies (Dai et al., 2023; Yao et al., 2023; Li et al., 2023b, 2024b; Bao et al., 2024) focused on leveraging LLMs to capture the complex semantics in detailed textual item information, providing a richer context for recommendations. Existing works are categorized into two approaches based on how they infer items (Li et al., 2024a; Wu et al., 2023): *discriminative* and *generative* approaches. We do not cover other models that perform different recommendation tasks, *e.g.*, yes/no (like/dislike) (Zhang et al., 2024; Bao et al., 2023; Zhang et al., 2023) or candidates re-ranking (Gao et al., 2023; Dai et al., 2023; Liao et al., 2024), since they are primarily beyond the scope of our work.

Discriminative Recommendation. The discriminative approach employs LLMs as a sequence encoder to encode user/item text. The relevance between user and item representations is computed in a manner analogous to traditional sequential recommendation models. Several studies (Hou et al., 2022; Li et al., 2023a; Liu et al., 2023; Tang et al., 2023) have proposed methods to encode user and item representations using the text that comprises various item attributes, *e.g.*, title, brand, and categories. While discriminative methods are one of the prominent pillars of text-based recommendation models, we omit further details as they fall outside the primary scope of our work.

B Examples of Prompts

Figure 9 illustrates the multi-granular prompts on the Beauty dataset used as inputs for GRAM. The prompts consist of a coarse-grained user prompt T_u and fine-grained user prompts T_i as in Eq. (5).

C Efficiency Analysis

C.1 Theoretical Complexity Analysis

Table 5 shows the theoretical complexity of early and late fusion for processing user and item prompts. While early fusion concatenates all texts at the input level with quadratic complexity $\mathcal{O}((|T_u| + |s| \cdot |T_{i_*}|)^2 d)$, our late fusion processes user and item prompts separately, achieving significant gains in efficiency. Specifically, GRAM reduces online computation to $\mathcal{O}(|T_u|^2 d)$ by processing the item prompt $\mathcal{O}(|s| \cdot |T_{i_*}|^2 d)$ in offline.

Method	Encoding complexity	Decoding complexity
Early fusion	Online: $\mathcal{O}((\lvert T_u\rvert + \lvert s\rvert \cdot \lvert T_{i_*}\rvert)^2 d + (\lvert T_u\rvert + \lvert s\rvert \cdot \lvert T_{i_*}\rvert)d^2)$	$\mathcal{O}(\tilde{i}_{\lvert s\rvert+1}(\lvert T_u\rvert + \lvert s\rvert \cdot \lvert T_{i_*}\rvert)d + \tilde{i}^2_{\lvert s\rvert+1}d + \tilde{i}_{\lvert s\rvert+1}d^2)$
GRAM (Late fusion)	Online: $\mathcal{O}(\lvert T_u\rvert^2 d + \lvert T_u\rvert d^2)$ Offline: $\mathcal{O}(\lvert s\rvert \cdot \lvert T_{i_*}\rvert^2 d + \lvert s\rvert \cdot \lvert T_{i_*}\rvert d^2)$	$\mathcal{O}(\tilde{i}_{\lvert s\rvert+1}(\lvert T_u\rvert + \lvert s\rvert \cdot \lvert T_{i_*}\rvert)d + \tilde{i}^2_{\lvert s\rvert+1}d + \tilde{i}_{\lvert s\rvert+1}d^2)$ -

Table 5: Computational complexity comparison between early and late fusion approaches. $|s|$ denotes the number of items in the user sequence. $|T_u|$ and $|T_{i_*}|$ are the number of tokens for the user prompt and the item prompt, respectively. $\tilde{i}_{|s|+1}$ the number of tokens for the target item ID.

Phase	Beauty Time (m)	Toys Time (m)
CF model training	2	2
Top-k CF item retrieval	1	1
Text encoding	17	16
Hierarchical clustering	23	10
Total time	43	29

Table 6: Preprocessing time (minutes) of GRAM. Note that CF model training time can vary depending on the model, and we utilized SASRec.

Model	Beauty R@5	Beauty Time (s)	Toys R@5	Toys Time (s)
IDGenRec	0.0463	0.1504	0.0462	0.1423
LC-Rec	0.0503	1.8312	0.0506	1.8125
GRAM	0.0641	0.2008	0.0718	0.1605

Table 7: Comparison of accuracy and inference time (seconds) between GRAM and key generative baselines. The inference time is measured by processing all user sequences in a single batch on a single A6000 GPU. We report the average inference time per user sequence.

For instance, with $|T_u|$=128, $|T_{i_*}|$=512, and $|s|$=20 items, early fusion requires encoding 10,368 tokens simultaneously, while GRAM only needs to encoding 128 tokens online, with the remaining encoding computations performed offline. It results in an 81× reduction (10,368 vs. 128 tokens) in online encoding complexity. Notably, since the number of tokens for target item ID $\tilde{i}_{|s|+1}$ is typically less than 10, the encoding phase dominates the computational burden (de Jong et al., 2023).

C.2 Empirical Efficiency Analysis

Preprocessing Phase. Table 6 shows the time for preprocessing data before training GRAM. Most importantly, hierarchical semantics indexing and collaborative semantics verbalization are **one-time** preprocessing before training. The preprocessing requires modest computational resources, taking less than an hour for the Beauty and Toys datasets. Notably, operations like clustering are highly parallelizable and can be further optimized. This preprocessing enables GRAM to achieve significant performance without affecting online inference time.

Inference Phase. Table 7 illustrates the inference time of GRAM against generative baselines. The inference time of GRAM represents worst scenarios where all computations are performed online without offline processing. GRAM achieves competitive speeds with superior performance. Even with all online processing, GRAM shows only a marginal increase in inference time (<50ms) compared to IDGenRec while achieving substantial performance gains. This overhead is minimal as the item prompts can be pre-computed offline. Notably, GRAM achieves 9x faster speed than LC-Rec with better performance, demonstrating its scalability for large-scale datasets.

D Hierarchical k-means for textual identifier

Algorithm 1 Hierarchical k-means

Input: Item embeddings $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_n\} \in \mathbb{R}^e$, Number of clusters k, Minimum cluster size c, Maximum depth l, Vocabulary $\mathcal{V}$
Output: Hierarchical lexical identifiers $\mathcal{H} = \{h_1, ..., h_n\}$

```
function HierarchicalClustering(Z, depth = 0)
    identifiers ← ∅
    if |Z| > c and depth < l then
        C_1, ..., C_k ← k-means(Z, k)
        for i ← 1 to k do
            t_i ← GetRepresentativeToken(C_i, V)
            H_i ← HierarchicalClustering(C_i, depth + 1)
            identifiers[i] ← Concatenate(t_i, H_i)
        end for
    end if
    return identifiers
end function

function GetRepresentativeToken(C, V)
    v ← 0 ∈ R^|V|
    for z in C do
        v ← v + TfIdf(ItemText(z))
    end for
    return argmax_{t∈V}(v[t])
end function
```

Model	Beauty				Toys				Sports			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Reported in Wang et al. (2024b)												
ELMRec (original)	0.0609	0.0486	0.0750	0.0529	0.0713	0.0608	0.0764	0.0618	0.0538	0.0453	0.0616	0.0471
Reproduced												
ELMRec (original)	0.0612	0.0486	0.0759	0.0533	0.0729	0.0638	0.0784	0.0649	0.0503	0.0421	0.0580	0.0444
ELMRec (ours)	**0.0372**	**0.0267**	**0.0506**	**0.0310**	**0.0148**	**0.0119**	**0.0193**	**0.0131**	**0.0241**	**0.0181**	**0.0307**	**0.0203**

Table 8: Reproduced results of ELMRec (Wang et al., 2024b) on the Beauty, Toys, and Sports datasets based on the indexing. We reported the result of **ELMRec (ours)** to resolve the leakage issue of sequential item IDs.

Model	R@5	N@5	R@10	N@10
Reported in Tan et al. (2024)				
w/ user ID	0.0618	0.0486	0.0814	0.0541
Reproduced				
w/ user ID	0.0634	0.0487	0.0832	0.0551
w/o user ID	**0.0463**	**0.0328**	**0.0665**	**0.0393**

Table 9: Reproduced results of IDGenRec (Tan et al., 2024) on the Beauty dataset based on the user IDs. We reported the result of **w/o user ID** to resolve the issue.

E Prompt Modification for IDGenRec

We observed a potential data leakage issue in the original implementation of IDGenRec (Tan et al., 2024) related to the construction of user IDs. The original prompt is shown below:

```
Considering user {user_id} has interacted
with items {history}. What is the next
recommendation for the user?
```

'`{user_id}`' takes the user ID, and '`{history}`' takes a sequence of concatenated item IDs generated from the item's metadata. The issue is that the user ID is constructed by concatenating all item IDs from the sequence, including validation and test items. For instance, given an item sequence $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_4$, according to the leave-one-out setting, $i_1 \rightarrow i_2$ is the training sequence, i_3 and i_4 are validation and test items, respectively. Here, the user ID generation process incorporates information about i_4, leading to data leakage during testing. To resolve this issue, we exclude user IDs in prompt sentences by following the suggestion of the authors as follows (Tan et al., 2024).[9]

[9]Please refer to the details in https://github.com/agiresearch/IDGenRec/issues/1.

```
Considering user has interacted with items
{history}. What is the next recommendation
for the user?
```

The rest of our code for IDGenRec is kept identical to the official source code provided in the paper. We also confirmed that the reported results from the original work are successfully reproduced in the original setting ('IDGenRec w/ user ID'), as shown in Table 9.

F Indexing Modification for ELMRec

As discussed in previous studies (Rajput et al., 2023; Lin et al., 2024)[10], the original sequential indexing method in P5 has data leakage issues (Geng et al., 2022). The issue arises when consecutive numeric IDs are assigned to items within user sequences, including validation and test items. For example, a sequence `[4392, 4393, ..., 4399]` where `4399` is the target item creates overlapping subword tokens (*e.g.*, '43') when tokenized by SentencePiece tokenizer (Sennrich et al., 2016). It creates unintended correlations between training and evaluation data.

When reproducing P5 variants, we follow the corrected setup from Hua et al. (2023); Xu et al. (2024) to resolve the issue, rather than using the original P5 indexing. We apply sequential indexing only to training data while excluding validation and test items. However, ELMRec (Wang et al., 2024b) adopts the original P5 sequential indexing in the public codebase[11]. For fair comparison, we modified ELMRec to use the same indexing approach as P5-SID, excluding validation and test items. We also confirmed that ELMRec's performance was successfully reproduced under the provided setting as shown in Table 8.

[10]Please refer to Appendix D of Rajput et al. (2023) and Appendix A.6 of Lin et al. (2024).

[11]https://github.com/WangXFng/ELMRec

Dataset	#Users	#Items	#Inters	Density
Beauty	22,363	12,101	198,502	0.0734%
Toys	19,412	11,924	167,597	0.0724%
Sports	35,598	18,357	296,337	0.0453%
Yelp	30,431	20,033	316,354	0.0519%

Table 10: Statistics of four benchmark datasets.

Subset	Beauty		Toys	
	#Users	#Items	#Users	#Items
Head	9,908	2,459	7,764	2,504
Tail	12,455	9,642	11,648	9,420

Table 11: Statistics of the Beauty and Toys datasets based on test subsets by target item popularity.

G Additional Experimental Setup

G.1 Datasets

The Amazon dataset consists of user reviews and item metadata collected from 1996 to 2014. The Yelp dataset contains user reviews and business information from 2019. We create item sequences with historical user reviews from the datasets following Xu et al. (2024). The statistics of preprocessed datasets are summarized in Table 10.

G.2 Data Statistics of Head/Tail Experiments

Table 11 provides statistics used in Table 6. We provide the number of items based on the groups: Head (top 20% by popularity) and Tail (remaining 80%). We also report the number of users based on each target item's group.

G.3 Baselines

We adopt six traditional sequential models and eight generative models as follows.

- **GRU4Rec** (Hidasi et al., 2016) is an RNN-based model that employs GRUs to encode sequences.
- **HGN** (Ma et al., 2019) employs a hierarchical gating network to capture both long-term and short-term user interests.
- **SASRec** (Kang and McAuley, 2018) uses the last item representation as the user representation using the uni-directional Transformer encoder.
- **BERT4Rec** (Sun et al., 2019) utilizes a bi-directional self-attention mechanism to perform the masked item prediction task.
- **FDSA** (Zhang et al., 2019) distinguishes between feature- and item-level self-attention for modeling and integration of item attributes.
- **S^3Rec** (Zhou et al., 2020) adopts four auxiliary self-supervised objectives to learn the correlations among sequence items and attributes.

Hyperparameters	Beauty	Toys	Sports	Yelp
ID length l	7	5	7	9
# of clusters k	128	32	32	32
cluster size c	128	32	32	32
# of similar items k	10	5	10	5

Table 12: Final hyperparameters for GRAM.

- **P5-SID** (Hua et al., 2023) adopts sequential indexing, which assigns numeric IDs based on the order of item appearance.
- **P5-CID** (Hua et al., 2023) employs spectral clustering to generate numeric IDs considering co-occurrence patterns of items.
- **P5-SemID** (Hua et al., 2023) uses item metadata, *e.g.*, categories, for assigning numeric IDs.
- **TIGER** (Rajput et al., 2023) introduces codebook IDs based on RQ-VAE.
- **IDGenRec** (Tan et al., 2024) generates textual IDs with item metadata by training ID-generator.
- **LETTER** (Wang et al., 2024a) integrates hierarchical semantics, collaborative signals, and diversity when assigning RQ-VAE IDs.
- **ELMRec** (Chen et al., 2022) incorporates high-order relationships while utilizing soft prompt and re-ranking strategy based on numeric IDs.
- **LC-Rec** (Zheng et al., 2024) utilizes RQ-VAE IDs and further integrates language and collaborative semantics via multi-task learning.

G.4 Implementation Details

G.4.1 Setup for Proposed Method

For hierarchical semantics extraction, we concatenated the item textual metadata as input and used NV-Embed (Lee et al., 2024) as a text encoder. For hierarchical semantics translation, we transformed item texts into T5 vocabulary, where $|V| = 32,100$. The model with the highest N@10 on the validation set was selected for test set evaluation. We conducted all experiments on a desktop with 2 NVIDIA RTX A6000, 512 GB memory, and 2 AMD EPYC 74F3. The final hyperparameters are in Table 12.

G.4.2 Setup for Baselines

- **Traditional methods**: We implemented traditional recommendation models on the open-source library RecBole (Xu et al., 2023). The baselines are optimized using Adam optimizer with a learning rate of 0.001, batch size of 256, and an embedding dimension of 64. We stopped the training if the validation NDCG@10 showed no improvement for 10 consecutive epochs. For

Dataset	Variants	All		Short (≤ 10)		Long (> 10)	
		R@5	N@5	R@5	N@5	R@5	N@5
Beauty	**GRAM**	0.0655	0.0462	0.0609	0.0425	0.0902	0.0661
	w/o user prompt	0.0634	0.0443	0.0591	0.0410	0.0860	0.0618
	Gain (%)	3.4	4.3	3.0	3.5	4.9	7.0
Toys	**GRAM**	0.0718	0.0516	0.0728	0.0526	0.0658	0.0457
	w/o user prompt	0.0709	0.0510	0.0722	0.0522	0.0630	0.0444
	Gain (%)	1.3	1.2	0.8	0.9	4.5	2.9

Table 13: Effectiveness of user prompts based on test subsets by lengths of user sequences. Gain measures the improvement of the proposed method over 'w/o user prompt.'

evaluation, we report test set accuracy using model checkpoints that achieved the highest validation scores. We followed the original papers' configurations for other hyperparameters and carefully tuned them if not specified.

- **P5-variants** (Hua et al., 2023): We utilized T5-small following the original implementation provided by the authors[12]. For the sequential indexing approach (P5-SID), we excluded validation and test items during the assignment of numeric IDs. (For a detailed discussion, see Appendix F.)
- **TIGER** (Rajput et al., 2023): As the official code was not publicly available, we implemented the model based on the specifications detailed in the paper. We used the Sentence-T5 (Ni et al., 2022) for extracting semantic embeddings with a hidden dimension of 768. The vocabulary size was set to 1024 (256×4).
- **IDGenRec** (Tan et al., 2024): Following the official code[13], we used T5-small but excluded user IDs to prevent data leakage following the authors' request, incurring lower accuracy than the original paper. (Refer to the further discussion in Appendix E.) We follow the 'standard recommendation' setting in the paper.
- **LETTER** (Wang et al., 2024a): We initiated LETTER on TIGER using the official code[14]. We followed the original setting and adopted a 4-level codebook configuration (256×4), with each codebook having a dimension of 32. The original settings were maintained, and the hyperparameters α and β were thoroughly tuned within the original search range.
- **ELMRec** (Wang et al., 2024b): We used T5-small following the official codebase[15]. While the original paper utilized validation loss for checkpoint selection and early stopping, we found this approach leads to convergence issues and significantly lower performance in our experiments. Thus, we adopted validation NDCG@10 for checkpoint selection and early stopping. We only performed direct and sequential recommendation tasks for the Yelp dataset, excluding the explanation generation task. Additionally, since items in user sequences can reappear as targets in the Yelp dataset, we did not employ the proposed reranking strategy. We also thoroughly fine-tuned hyperparameters of ELMRec ($\alpha, \beta, N, \sigma, L$) following the search range of the original paper.
- **LC-Rec** (Zheng et al., 2024): We followed the official code[16], including the data pre-processing described in the paper. We fully fine-tuned LLaMA-7B (Touvron et al., 2023) using the authors' hyperparameters. The only exception was that the number of samples for the 'fusion sequence recommendation task' was adjusted according to each dataset's interaction numbers. Specifically, we use 15k samples for the Toys, 20k for the Beauty, and 25k for the Yelp and Sports datasets. For generating user preferences, we employed `gpt-4o-mini-2024-07-18`[17] on the Amazon datasets. The Yelp dataset was excluded due to its lack of metadata (*i.e.*, reviews) for preference generation.

H Additional Experimental Results

H.1 In-depth Study of User Prompts

Table 13 shows the impact of user prompts across different sequence lengths to understand their role in the recommendation. While GRAM consistently outperforms the variant without user prompts

[12] https://github.com/Wenyueh/LLM-RecSys-ID
[13] https://github.com/agiresearch/IDGenRec
[14] https://github.com/HonghuiBao2000/LETTER
[15] https://github.com/WangXFng/ELMRec
[16] https://github.com/RUCAIBox/LC-Rec
[17] https://chatgpt.com

	Beauty		Toys	
	R@5	N@5	R@5	N@5
P5-SID	0.0465	0.0329	0.0216	0.0151
P5-CID	0.0465	0.0325	0.0223	0.0143
P5-SemID	0.0459	0.0327	0.0264	0.0178
P5 + HID	**0.0582**	**0.0404**	**0.0574**	**0.0397**
IDGenRec	0.0463	0.0328	0.0462	0.0323
IDGenRec + HID	**0.0499**	**0.0352**	**0.0656**	**0.0475**

Table 14: Effectiveness of hierarchical IDs (HID) when applying to existing generative recommenders.

	Beauty		Toys	
	R@5	N@5	R@5	N@5
P5-SID	**0.0465**	**0.0329**	0.0216	0.0151
P5-SID + CF	0.0454	0.0295	**0.0419**	**0.0263**
IDGenRec	0.0463	0.0328	0.0462	0.0323
IDGenRec + CF	**0.0623**	**0.0420**	**0.0513**	**0.0337**

Table 15: Effectiveness of collaborative semantics verbalization (CF) when applying to existing generative recommenders.

across all groups, the improvements are particularly significant for long sequences (length > 10). For the Beauty dataset, incorporating user prompts achieves gains of up to 7.0% in N@5 for long sequences, compared to 3.5% for short sequences. Similar trends are observed in the Toys dataset, with gains of 2.9% and 0.9% in N@5 for long and short sequences, respectively. It demonstrates that user prompts effectively preserve sequential dependencies, especially for longer user histories, without sacrificing efficiency through late fusion.

H.2 Generalizability Analysis of Components

As shown in Table 14, hierarchical IDs demonstrate consistent improvements when applied to existing generative recommendation models. When applied to P5, it yields substantial gains of up to 23.0% in N@5. Similarly, IDGenRec achieves improvements of up to 47.1% in and N@5. It implies that semantic-to-lexical translation effectively bridges the gap between item relationships and language understanding capabilities of LLMs, even with other generative recommendation models.

Table 15 illustrates the impact of collaborative semantics verbalization. For compatibility, collaborative signals are appended after the original prompt sentence of each model. Notably, IDGenRec shows gains of up to 34.6% in R@5 with collaborative signals. However, the improvements vary by model, indicating that separating item prompts

Semantic	Fusion	Beauty		Toys	
		R@5	N@5	R@5	N@5
✓	✓	**0.0641**	**0.0451**	**0.0718**	**0.0516**
✓	✗	0.0582	0.0404	0.0574	0.0397
✗	✓	0.0534	0.0382	0.0538	0.0384
✗	✗	0.0516	0.0366	0.0459	0.0336

Table 16: Performance of GRAM based on two main components: semantic-to-lexical translation ('Semantic') and multi-granular late fusion ('Fusion').

	Beauty		Toys	
	R@5	N@5	R@5	N@5
T5-small	0.0641	0.0451	0.0718	0.0516
T5-base	**0.0646**	**0.0457**	**0.0719**	**0.0520**

Table 17: Performance of GRAM over various model sizes. We adopt 'T5-small' for other experiments.

is more beneficial than direct concatenation with existing prompts. This aligns with our multi-granular design principle of maintaining distinct granularities of information.

H.3 Synergistic Effects between Components

To analyze the effectiveness of key components of GRAM, we extend our investigation beyond the ablation study in Table 3. Table 16 demonstrates both the individual and combined effects of the components. Each component independently improves performance over GRAM without any components (fourth row), with semantic-to-lexical translation showing a more substantial impact compared to multi-granular late fusion. Most importantly, the results reveal significant synergistic effects between the two key components. While semantic-to-lexical translation alone yields up to 25.1% improvement in R@5 (third and fourth row), its combination with multi-granular late fusion leads to a 33.4% improvement (first and second row). This enhanced performance demonstrates the potential synergy of the components within our framework.

H.4 Effect of Model Size

To evaluate the generalizability of GRAM, we examine the accuracy across different model sizes, as presented in Table 17. GRAM achieves superior performance compared to competitive baselines with both T5-small and T5-base architectures. Although increasing the model size yields modest performance improvements, we hypothesize that more extensive hyperparameter tuning could potentially achieve better performance with larger models.

Model	Beauty		Toys		Sports		Yelp	
	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20
Traditional recommendation models								
GRU4Rec	0.0934	0.0430	0.0794	0.0372	0.0573	0.0247	0.0659	0.0273
HGN	0.0903	0.0372	0.0840	0.0352	0.0539	0.0221	0.0777	0.0366
SASRec	0.0676	0.0300	0.0562	0.0271	0.0315	0.0137	0.0457	0.0275
BERT4Rec	0.0729	0.0294	0.0567	0.0231	0.0384	0.0154	0.0655	0.0274
FDSA	0.1065	0.0551	0.1057	0.0578	0.0568	0.0281	0.0859	0.0365
S^3Rec	0.0984	0.0405	0.0927	0.0388	0.0577	0.0243	0.0533	0.0212
Generative recommendation models								
P5-SID	0.0823	0.0431	0.0407	0.0208	0.0518	0.0276	0.0625	0.0302
P5-CID	0.0928	0.0456	0.0555	0.0236	0.0597	0.0299	0.0567	0.0250
P5-SemID	0.0958	0.0468	0.0619	0.0278	0.0675	0.0342	0.0504	0.0225
TIGER	0.0775	0.0355	0.0657	0.0282	0.0418	0.0179	0.0406	0.0171
IDGenRec†	0.0930	0.0460	0.0899	0.0446	0.0579	0.0273	0.0636	0.0311
ELMRec†	0.0444	0.0231	0.0114	0.0067	0.0263	0.0157	0.0367	0.0225
LC-Rec	0.0973	0.0485	0.0964	0.0485	0.0550	0.0257	0.0720	0.0341
GRAM	**0.1216**	**0.0613**	**0.1303**	**0.0682**	**0.0796**	**0.0375**	**0.1012**	**0.0476**
Gain (%)	14.2*	11.3*	23.3*	18.0*	18.0*	9.8*	17.8*	30.1*

Table 18: Performance comparison with cutoff 20. The best model is marked in **bold**, and the second-best model is underlined. Gain measures the improvement of the proposed method over the best competitive baseline. '$*$' indicates statistical significance ($p < 0.05$) by a paired t-test.

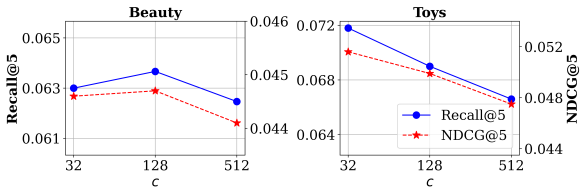

Figure 10: Performance of GRAM over varying numbers of cluster size c for hierarchical clustering.

H.5 Additional Results with Cutoff 20

Table 18 shows the effectiveness of GRAM with a larger recommendation list (cutoff=20). GRAM consistently outperforms all baselines across datasets, achieving gains of up to 23.3% and 30.1% in R@20 and N@20, respectively. The performance gains are more pronounced than the results with the cutoff of 10, particularly for Yelp and Toys datasets. It suggests that GRAM effectively maintains accuracy in longer recommendation lists, likely due to its capacity to leverage both item relationships and comprehensive item information.

H.6 Effect of Cluster Size

Figure 10 demonstrates the impact of the cluster size c. The optimal cluster size varies across datasets. It underscores the importance of careful hyperparameter tuning when applying the model to datasets with distinct characteristics.

H.7 Case Study

Figure 11 shows the cross-attention scores for prompts, consisting of coarse-grained user prompts and fine-grained item prompts, as defined in Eq. (8). These scores directly affect recommendation results by identifying salient tokens during the decoding process. As highlighted in red, GRAM leverages descriptions from fine-grained item prompts for recommendations. While previous methods struggle with input length limits and efficiency constraints, GRAM effectively detours the challenges through separate prompt encoding and late fusion.

In the first example, GRAM assigns higher scores to terms such as "*restores*" and "*normalizes*" in the description, which captures fine-grained information of the first item. These terms semantically align with the hierarchical identifier of the target item, "*hair-pho-ge-reconstruct-minute-kermoderate*." The absence of this information in other attributes underscores the importance of maintaining diverse and detailed attributes.

The third example illustrates how GRAM assigns higher scores to collaborative knowledge, specifically for similar items. Interestingly, when the CF model predicts the target item as a similar item, and GRAM effectively leverages this information to rank the target item as the top-1 recommendation. It demonstrates the critical role of collaborative semantics during prediction, effectively enhancing the ability to utilize collaborative knowledge.

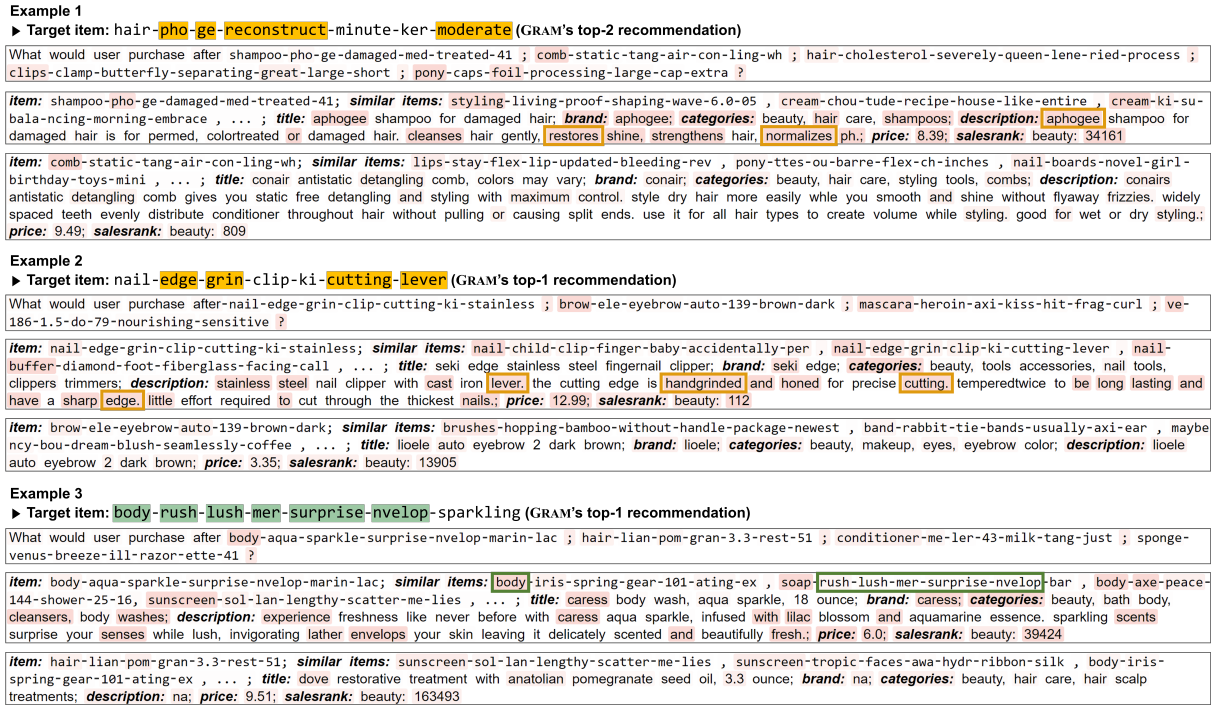

Figure 11: Visualization of cross-attention scores in GRAM. Our model utilizes not only item IDs but also similar items and detailed textual information for recommendations. Darker shades of red indicate higher attention scores. The orange box highlights textual descriptions, while the green box highlights the collaborative signals involved in the recommendation.