

AgenticTagger: Generating Structured Item Representation for Recommendation with LLM Agents

Zhouhang Xie^{1*} † Bo Peng^{2*} † Zhankui He^{3*} Ziqi Chen^{2*} Alice Han² Isabella Ye²
Benjamin Coleman³ Naveen Sachdeva³ Fernando Pereira³ Julian McAuley¹
Wang-Cheng Kang³ Derek Zhiyuan Cheng³ Beidou Wang² Randolph Brown²

¹UC San Diego ²Google ³Google DeepMind
zhx022@ucsd.edu bopeng0823@gmail.com {zhankui, zqchen}@google.com

Abstract

High-quality representations are a core requirement for effective recommendation. In this work, we study LLM-based descriptor generation, aiming to unify the strengths of textual and ID-based item representation paradigms by adopting natural language concepts as compact, interpretable identifiers. However, open-ended text generation provides little vocabulary control, resulting in a sparse, high-cardinality feature space that renders downstream modeling challenging. To this end, we propose AgenticTagger, a framework that queries LLMs to represent items with textual descriptors drawn from an automatically mined vocabulary.

AgenticTagger features two core stages: (1) a vocabulary-building stage in which a set of hierarchical, low-cardinality, and informative descriptors is identified, and (2) a vocabulary-assignment stage in which LLMs assign in-vocabulary descriptors to items. To ensure the vocabulary effectively describes the item corpus of interest, we design a multi-agent collaboration mechanism in which an architect LLM iteratively refines the vocabulary, guided by parallelized feedback from annotator LLMs that validate the vocabulary against item data. Experiments on public and private datasets show AgenticTagger brings consistent improvements across diverse recommendation scenarios, including generative retrieval, ranking, and critique-based recommendation.

1 Introduction

High-quality item representations are the foundation of effective recommendation systems (Rendle et al., 2009; Kang et al., 2021; Rajput et al., 2023; Singh et al., 2024; Hou et al., 2023; Coleman et al., 2023; Zhang et al., 2024a, 2025a). In this work, we seek a natural language item representation that unifies sought-after properties currently

Method	Efficient	Interpretable	Model Agnostic
Item IDs	✓	×	✓
Hashed/Sem-IDs	✓	×	✓
Text Embeddings	×	×	×
Free-form Text	×	✓	×
AgenticTagger	✓	✓	✓

Table 1: Overview of item representation paradigms. AgenticTagger represents items with a sequence of LLM-generated descriptors (Figure 1), maintaining the efficiency of standard item IDs while being natively interpretable and compatible with diverse downstream modeling choices.

scattered across distinct representation paradigms: efficiently compressing items into a low-cardinality codebook (Rajput et al., 2023; Singh et al., 2024), remaining natively interpretable like free-form descriptions (Valizadeh et al., 2025; Ren et al., 2024; Shi et al., 2025; Yada and Yamana, 2024; Wang et al., 2024a), and maintaining compatibility with diverse modeling choices, as shown in Table 1.

However, it is nontrivial to realize such a representation. While these individual properties have long benefited recommender systems, current methods enforce an explicit trade-off. On one hand, hashed and semantic item IDs efficiently compress the item corpus into a low-cardinality codebook, flexibly integrating into diverse architectures without requiring language understanding components (Rajput et al., 2023; Hou et al., 2023; Singh et al., 2024; He et al., 2025), yet they are inherently opaque. On the other hand, while free-text item descriptions are interpretable, leveraging them requires specialized recommender systems equipped with language-understanding capabilities. This typically involves either deploying LLMs as the recommendation backbone via prompting (Liu et al., 2025a; Wang et al., 2024b) or fine-tuning (Jiang et al., 2025; Zhang et al., 2026), or featurizing the text into dense embeddings (Valizadeh et al., 2025;

* Equal contribution.

† Work done while ZX and BP were at Google.

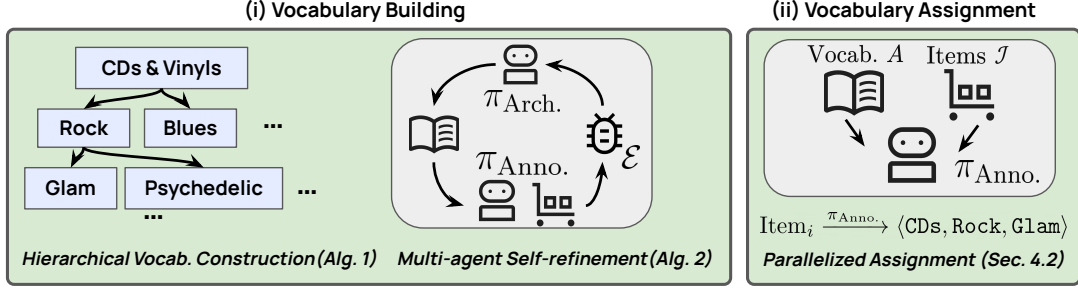


Figure 1: **The AgenticTagger Framework.** AgenticTagger exploits LLMs’ content-understanding ability to automatically construct an interpretable, hierarchical descriptor vocabulary from data. Then, relevant descriptors in the vocabulary are assigned to corresponding items by querying LLMs in parallel over the item corpus (e.g., representing an album as $\langle CDs, Rock, Glam \rangle$). These descriptors could then power various downstream applications, such as generative retrieval, ranking, and critique-based recommendation.

Ren et al., 2024; Yada and Yamana, 2024). The former imposes strict architectural constraints on the recommendation pipeline, while the latter compromises the efficiency of discrete codebooks and sacrifices interpretability.

In contrast, if we can exchange opaque ID codebooks for a similarly compact vocabulary of natural language concepts, one could represent each item as a structured sequence of concept descriptors. For example, as illustrated in Figure 1, an album could be concisely described as $\langle CDs \& Vinyls, Rock, Glam \rangle$. For the sake of discussion, we refer to these concepts as *descriptors*, and the collection of descriptors as the *vocabulary*. On the surface, this could be achieved by prompting an LLM to generate descriptors for each item. However, while LLMs are exceptional at generating item descriptions (Valizadeh et al., 2025; Ren et al., 2024; Shi et al., 2025; Yada and Yamana, 2024; Wang et al., 2024a), open-ended generation lacks the global control required for producing a low-cardinality descriptor vocabulary. Consequently, such a method inevitably produces descriptors with subtly varying surface forms (e.g., “Rock Music” versus “Rock Hits”), leading to a sparse, high-cardinality vocabulary. Meanwhile, without a view of the whole corpus, it is also unclear what kind of descriptors meaningfully summarize informative properties of an item. We provide empirical validation of these challenges in Appendix A.1 and Appendix C.3.

To address these issues, we propose AgenticTagger, a multi-LLM framework that queries LLMs for structured natural language descriptors, which are drawn from an automatically learned low-cardinality vocabulary. Specifically, AgenticTagger features two key stages: (1) **vocabulary building** (Section 4.1), where AgenticTagger automatically

learns a hierarchical coarse-to-fine descriptor taxonomy to capture item essence at multiple granularities, drawing insights from LLMs for taxonomy mining (Wan et al., 2024; Xie et al., 2025; Pham et al., 2024) and codebook learning for item representation in recommendation (Hou et al., 2023, 2025b,a; Singh et al., 2024; He et al., 2025; Rajput et al., 2023); and (2) **vocabulary assignment** (Section 4.2), which prompts LLMs to assign relevant descriptors from the vocabulary to items. To compress the information of an entire item corpus into a fixed vocabulary, we design a multi-agent feedback loop: a strong “architect” LLM iteratively refines the vocabulary based on verbal feedback (Shinn et al., 2023; Majumder et al., 2024; Xie et al., 2024) from parallelized “annotator” LLMs that scan the item corpus, looking for outlier items that cannot be described by the vocabulary. This process ensures the vocabulary is both representative and comprehensive. By mining a high-quality, low-cardinality descriptor vocabulary and performing in-vocabulary generation, AgenticTagger effectively constrains the LLMs’ generation space while equipping them with a global, hierarchical view of the item corpus, enforcing structural constraints on LLM-generated descriptors.

We conduct experiments on AgenticTagger across public benchmarks and a private dataset, showing the broad applicability of AgenticTagger via its improvements to diverse recommendation settings, including generative recommendation and ranking. We also show that these natural language descriptors natively bring interpretability and controllability to recommender systems, e.g., by making generative recommender models natively compatible with natural language critiques. Our contributions are as follows:

- We propose a natural language item representation that bridges discrete codebooks and free-form LLM generation, functioning as compact categorical features while preserving human-interpretable semantics.
- We identify a core challenge in generating such representations, the lack of vocabulary control, and propose a multi-agent, two-stage descriptor generation framework to address this issue.
- To the best of our knowledge, we are the first to apply agentic frameworks to feature mining to benefit recommender systems, opening up a new direction for future work.

2 Related Work

LLM-generated Features in RecSys. Prior work in introducing LLM-produced features in recommender systems broadly falls into two streams: LLM-as-encoders (Zhang et al., 2024a, 2025a) and LLM-as-text-generators (Ren et al., 2024; Yada and Yamana, 2024; Valizadeh et al., 2025; Liu et al., 2024, 2025b), and this work aims at enhancing the latter. In contrast to the popular goal of generating user profiles that improve an LLM itself at the recommendation task (Liu et al., 2025b,a), we consider specifically the case where the recommendation model is a general feature-aware ML model, which aligns well with most existing deployed recommendation models. Compared to the most similar works that encode LLM-produced descriptions (Ren et al., 2024; Yada and Yamana, 2024; Valizadeh et al., 2025), we are the first to study descriptor (fixed-slot, nominal feature) generation, and the first to demonstrate the value of LLM agents in producing features for RecSys.

Discrete Representation in RecSys. Our work is also related efforts in building better discrete representations for RecSys, which typically involve discretizing item embeddings via quantization (Jegou et al., 2011; Ge et al., 2014; Hou et al., 2023, 2025a), RQ-VAE (van den Oord et al., 2017; Rajput et al., 2023), and K-means (Ju et al., 2025; Zhang et al., 2024c). However, these methods yields uninterpretable results, with recent attempts on LLM-generated text-based IDs such as Term IDs as an interpretable exception (Zhang et al., 2026). Yet, these free-form generated IDs targets the case of recommendation using LLM backbone models, and in contrast we seek representations

that are model agnostic.

LLM Agents for Recommendation. There have been numerous recent efforts in leveraging LLM agents to improve RecSys. These frameworks generally focus on leveraging LLMs for item recommendation (Shu et al., 2023; Zhao et al., 2024; Thakkar and Yadav, 2024; Wang et al., 2024e; Xia et al., 2026; Huang et al., 2025), or enabling better interactivity by incorporating LLMs and LLM agents into the recommendation-serving stage (Friedman et al., 2023; He et al., 2023; Zhang et al., 2024b). However, these frameworks typically require specialized pipelines, whereas AgenticTagger focuses on feature generation, which potentially benefits diverse downstream recommendation models. Meanwhile, to the best of our knowledge, AgenticTagger is the first framework to study agentic feature generation for recommender systems.

3 Problem Statement

Consider the standard recommendation setting with item features. Given a set of users $\mathcal{U}$ and items $\mathcal{I}$, the objective is to predict unobserved items of interest for a user $u \in \mathcal{U}$ based on their history of interactions $S_u \subset \mathcal{I}$. Each item $i \in \mathcal{I}$ is associated with raw textual attributes $X_i \in \mathcal{X}$, where $\mathcal{X}$ is the original feature space. In the context of feature generation, our goal is to find an optimal one-to-many transformation function, $g : \mathcal{X} \rightarrow \mathcal{P}(A)$, that maps these original attributes to a new discrete feature space A . The optimal transformation, g^* , is one that maximizes the performance of a downstream recommender system, f :

$$g^* = \arg \max_g \mathcal{M}(f_{\theta^*(g)})$$

where $f_{\theta^*(g)}$ is the recommender model trained using the generated features, and $\mathcal{M}$ represents relevant performance metrics, e.g., offline metrics and key business metrics. Note that while AgenticTagger does not directly optimize for end performance due to the challenge of propagating gradients through an API-supported LLM, the best vocabulary should ideally, in our problem formulation, maximize downstream task performance. We provide an overview of notations in Table 12.

4 AgenticTagger

As shown in Figure 1, the overall pipeline of AgenticTagger involves two key stages: the vocabulary-

Algorithm 1 Vocabulary Building

```
1: Input: Item corpus  $\mathcal{I}$ , LLM Ensemble  $\Pi$ 
2: Hyperparameters: Max hierarchy depth  $D_{max}$ , split threshold  $\tau_{split}$ 
3: Output: Hierarchical descriptor set  $A_{all}$ 
4:
5:  $A_{all} \leftarrow \emptyset$ 
6: Init. root descriptor  $p_0$  with items  $\mathcal{I}_{p_0} \leftarrow \mathcal{I}$ 
7:  $Q \leftarrow \{p_0\}$ 
8: for  $d = 1$  to  $D_{max}$  do
9:   if  $|Q| = 0$  then
10:    break
11:    $Q_{next} \leftarrow \emptyset$ 
12:   for each parent descriptor  $p \in Q$  do
13:     if  $|\mathcal{I}_p| \geq \tau_{split}$  then
14:        $A_{children} \leftarrow \text{Algorithm 2}(\mathcal{I}_p, \Pi, p)$ 
15:        $A_{all} \leftarrow A_{all} \cup A_{children}$ 
16:        $Q_{next} \leftarrow Q_{next} \cup A_{children}$ 
17:    $Q \leftarrow Q_{next}$ 
18: return  $A_{all}$ 
```

building stage, where we produce a descriptor vocabulary by observing typical items in the dataset, and the vocabulary-assignment stage, where we assign the best set of descriptors to each item.

4.1 Vocabulary Building

During the vocabulary-building stage, AgenticTagger automatically learns a vocabulary from the item corpus. However, gradient-based vocabulary learning (Rajput et al., 2023; Hou et al., 2023, 2025a; Zhang et al., 2024c) is inapplicable to prompting-based LLM agents, and thus the common solution is prompting an LLM in mini-batches (Wan et al., 2024; Pham et al., 2024) and refining an in-context vocabulary via LLM self-refinement (Madaan et al., 2023). Yet, such an iterative approach involves a non-parallelizable loop over the whole item corpus, as LLMs must sequentially refine their in-context vocabulary after seeing each batch of items, which scales poorly to large datasets (Xie et al., 2025).

Hierarchical Vocabulary Construction. In light of the above issues, in AgenticTagger, we opt to scale existing LLM-based frameworks by making the vocabulary-building process parallelizable, where multiple LLMs are deployed concurrently to update a shared vocabulary that describes observed items. An overview of this process is illustrated in Algorithm 1 and Algorithm 2. Specifically, the

iterative feature generation follows a hierarchical-clustering-like pipeline (Algorithm 1), where we generate an initial set of features that partitions the whole item corpus $\mathcal{I}$, followed by a top-down procedure to generate finer-grained features that further partition items associated with each feature $\mathcal{I}_p$ into smaller subsets. For example, if the initial features include high-level category information such as “CDs and Vinyl”, all relevant items will then form a subcluster, for generating finer-grained features such as “Rock” and “Blues”, as shown in Figure 1-(i). By decomposing the task of generating an entire vocabulary into a top-down hierarchical procedure, AgenticTagger effectively limits the number of descriptors an LLM needs to produce at a time, which scales better to a large corpus. Meanwhile, the resulting hierarchical vocabulary naturally organizes items in a coarse-to-fine taxonomy, similar to hierarchical semantic IDs (Rajput et al., 2023; He et al., 2025), which are known to act as effective item representations.

Multi-agent Self-refinement. At the core of this top-down generation loop is a multi-agent iterative refinement (Figure 1-(ii) and Algorithm 2) subroutine reused for each feature generation job p , which aims to produce a set of features that describe common items in the corresponding subset. Our core insight is leveraging a stronger (though potentially slower) LLM, $\pi_{Architect}$, to generate and refine the vocabulary subset relevant to each parent descriptor p , and copies of a faster (and potentially weaker) LLM, $\pi_{Annotator}$, which attempt to annotate the current item corpus using the available vocabulary and provide feedback to $\pi_{Architect}$ to improve the vocabulary. Unlike in single-LLM self-refinement (Wan et al., 2024; Pham et al., 2024; Madaan et al., 2023), the most costly $\pi_{Annotator}$ operations are parallelized in AgenticTagger, while we can still use their feedback to prompt $\pi_{Architect}$ to refine the vocabulary.

As shown in Algorithm 2, during the subroutine execution, we first prompt the $\pi_{Architect}$ LLM to generate the initial set of descriptors by observing a small subset of items, $\text{DISTILL}(\mathcal{I}_{sub})$, in its context, and then deploy copies of (or batched queries to) the annotator LLMs $\pi_{Annotator}$ to determine if any of the currently known features apply to each item in $\mathcal{I}_{sub}$, generating a set of error suggestions $\mathcal{E}$. The specific prompts are as shown in Appendix C.1. The error reports $\mathcal{E}$ then act as inductive biases for prompting the vocabulary

Algorithm 2 Multi-agent Self-refinement

```
1: Input: Item corpus subset  $\mathcal{I}_{sub} \subseteq \mathcal{I}$ ; Parent descriptor  $p$ ;  
   LLMs  $\Pi = \{\pi_{\text{Architect}}, \pi_{\text{Annotator}}\}$   
2: Hyperparameters: Max refinement cycles  $C_{max}$ , anomaly threshold  $\tau_{anom}$   
3: Output: Descriptor vocabulary  $A$  (with partitioned subsets  $\mathcal{I}_a$  for each  $a \in A$ )  
4:  
5:  $A \leftarrow \pi_{\text{Architect}}(p, \text{DISTILL}(\mathcal{I}_{sub}))$   
6: for  $c = 1$  to  $C_{max}$  do  
7:    $(\mathcal{I}_{assigned}, \mathcal{I}_{unassigned}, \mathcal{E}) \leftarrow \text{PARALLELASSIGN}(\mathcal{I}_{sub}, A, \pi_{\text{Annotator}})$   
8:   if  $|\mathcal{I}_{unassigned}| < \tau_{anom}$  then  
9:     break  
10:   $A \leftarrow \pi_{\text{Architect}}(A, \text{DISTILL}(\mathcal{E}))$   
11: Update each  $a \in A$  with its associated items from  $\mathcal{I}_{assigned}$  to form  $\mathcal{I}_a$   
12: return  $A$ 
```

maintainer LLM $\pi_{\text{Architect}}$ to refine the current set of features. To avoid overcrowding $\pi_{\text{Architect}}$'s context, $\text{DISTILL}(\cdot)$ refers to a general operation that selects diverse representative samples from a larger set of item descriptions, such as the centroids from K-Medoids clustering (Kaufman and Rousseeuw, 1987), as we use in our experiments. We repeat this iterative refinement loop until a fixed number of optimization steps has been reached, or a desired number of items in $\mathcal{I}_{sub}$ can be described by the generated features A . The union of features generated for each subset of the item corpus then forms the final feature space A_{all} , providing a coarse-to-fine description for the whole item corpus.

4.2 Vocabulary Assignment

Once the feature vocabulary A_{all} is built, the vocabulary assignment stage for AgenticTagger is straightforward, where we simply constrain the descriptors output by the LLMs to fall into the vocabulary by prompting, i.e., $g(X_i) = \{a \in A \mid \pi(X_i, A) \rightarrow a\}$ for any raw item attribute X_i . For example, in our experiments, for the sake of implementation simplicity, we opt to reuse the same operation from building the vocabulary, $\text{PARALLELASSIGN}(\mathcal{I}, A_{all}, \pi_{\text{Annotator}})$, where an LLM is prompted to assign features to an item by observing all features in its context, exploiting modern LLMs' long-context and reasoning abilities (Zhao et al., 2025). As we later show in Section 5, the generated descriptors are indeed of suffi-

cient quality to support good performance in downstream recommendation tasks. We provide further analysis and qualitative examples in Appendix C.4.

4.3 Complexity Analysis

The complexity of AgenticTagger mainly occurs in vocabulary building, as the assignment stage simply involves looping through all items of interest, with a one-time, parallelizable, and offline $O(|\mathcal{I}|)$ LLM batch query job. In its most basic version, the vocabulary building algorithm will incur $O(C_{max} \cdot N \cdot b^{D_{max}})$ LLM calls, where N is the number of items from a representative subset of the entire item corpus. This calculation assumes that an item will be assigned to an average of b features at each depth level $d \in \{1, \dots, D_{max}\}$, which grows quickly as the depth of the feature hierarchy grows. However, the expressiveness of the feature hierarchy also grows exponentially w.r.t. the depth of the hierarchy, similar to other hierarchical discrete item representation methods (Rajput et al., 2023; Zhang et al., 2024c), and thus a few layers are often sufficient; as we later show in our experiments, 3-6 layers of features are sufficient for good performance (Table 2).

Further, since the goal of vocabulary building is solely to generate a set of feature descriptors, we can also heuristically constrain the branching factor by limiting how many features an item is assigned to at each layer, such as randomly assigning an item to one of its associated features during the branching operation. In practice, we observe that a branching factor of 1 is also sufficient for good performance (Section 6.4), in which case the complexity w.r.t. LLM calls becomes $O(C_{max} \cdot N \cdot D_{max})$. Finally, compared to prior LLM-based document categorization methods (Wan et al., 2024; Pham et al., 2024), since AgenticTagger does not have sequential dependency between most of its dominant LLM operations (specifically, the $\pi_{\text{Annotator}}$ annotations), the overall execution time can be further reduced w.r.t. the number of concurrent threads in PARALLELASSIGN . We provide wall-clock runtime discussion in Appendix D, where AgenticTagger builds its vocabulary on over 60k items in a few hours with a modest API concurrency of 32.

5 Main Experiments

We evaluate AgenticTagger-produced descriptors by deploying them as discrete features in recommendation models, specifically for generative

Table 2: AgenticTagger for Generative Recommendation on academic benchmarks. Our model’s performance is reported across 3 runs. We further discuss improvements against against crossing-based IDs (Hou et al., 2025b; Singh et al., 2024) and free-form generations in Figure 2 and Appendix A.1.

Datasets	Metric	ID-based		Feature + ID			Generative				
		BERT4Rec	SASRec	FDSA	S ³ -Rec	VQ-Rec	LMIndexer	HSTU	P5-CID	TIGER	AgenticTagger
Sports	R@5	0.0115	0.0233	0.0182	0.0251	0.0181	0.0222	0.0258	0.0287	0.0264	0.0299 ± 0.0010
	N@5	0.0075	0.0154	0.0122	0.0161	0.0132	0.0142	0.0165	0.0179	0.0181	0.0194 ± 0.0007
	R@10	0.0191	0.0350	0.0288	0.0385	0.0251	—	0.0414	0.0426	0.0400	0.0474 ± 0.0002
	N@10	0.0099	0.0192	0.0156	0.0204	0.0154	—	0.0215	0.0224	0.0225	0.0251 ± 0.0004
Beauty	R@5	0.0203	0.0387	0.0267	0.0387	0.0434	0.0415	0.0469	0.0468	0.0454	0.0492 ± 0.0007
	N@5	0.0124	0.0249	0.0163	0.0244	0.0311	0.0262	0.0314	0.0315	0.0321	0.0332 ± 0.0008
	R@10	0.0347	0.0605	0.0407	0.0647	0.0741	—	0.0704	0.0701	0.0648	0.0728 ± 0.0010
	N@10	0.0170	0.0318	0.0208	0.0327	0.0372	—	0.0389	0.0400	0.0384	0.0408 ± 0.0008
CDs	R@5	0.0326	0.0351	0.0226	0.0213	0.0314	—	0.0417	0.0505	0.0492	0.0544 ± 0.0005
	N@5	0.0201	0.0177	0.0137	0.0130	0.0209	—	0.0275	0.0326	0.0329	0.0358 ± 0.0002
	R@10	0.0547	0.0619	0.0378	0.0375	0.0485	—	0.0638	0.0785	0.0748	0.0836 ± 0.0003
	N@10	0.0271	0.0263	0.0186	0.0182	0.0264	—	0.0346	0.0416	0.0411	0.0453 ± 0.0002

recommendation (Section 5.1) and ranking (Section 5.2).

5.1 Generative Recommendation

Datasets and Evaluation. We conduct our experiments across three domains from the Amazon Reviews dataset (McAuley et al., 2015): “Sports and Outdoor” (**Sports**), “Beauty” (**Beauty**), and “CDs and Vinyl” (**CDs**); the dataset statistics are shown in Table 8. For the larger CDs domain only, we constrain AgenticTagger’s branching factor to 1 for scalability and apply the vanilla AgenticTagger algorithm to the other two domains (see Section 6.4 for discussions). We adopt the commonly used “last-out” splitting (Kang and McAuley, 2018; Rajput et al., 2023; Hou et al., 2025a), where we hold out the last chronologically interacted item for each user as the test set, and the second-last item for validation. Following standard convention (Rajput et al., 2023; Wang et al., 2024d), we use T5 (Raffel et al., 2020) trained from scratch as the backbone model, and report Recall@ K and NDCG@ K , where $K \in \{5, 10\}$. We repeat our experiments with 3 random seeds and report the average metrics. We use Gemini-2.5-flash and Gemini-2.5-flash-lite for the architect and annotator LLMs, respectively, in our main experiment. We discuss results from Gemini-1.5-flash and Gemini-1.5-flash-lite (specifically, the 001 series), which achieved similarly competitive performance, as discussed in Appendix A.3.

Baselines. In our main experiment (Table 2), we compare our method with several families of models: models with item IDs only, such

as SASRec (Kang and McAuley, 2018) and BERT4Rec (Sun et al., 2019); models enhanced by embedded features: FDSA (Zhang et al., 2019), S³-Rec (Zhou et al., 2020); and models with discrete features (i.e., semantic IDs) derived from semantic information or collaborative information: VQ-Rec (Hou et al., 2023), TIGER (Rajput et al., 2023), HSTU (Zhai et al., 2024), and LLM-Indexer (Jin et al., 2024). In the main paper, we opt to prioritize fair comparison with content-based semantic IDs, to evaluate the quality of AgenticTagger. We further discuss our improvements against models that enhance raw semantic IDs via feature crossing, namely SPM-CID (Singh et al., 2024) and Action-Piece (Hou et al., 2025b) in Section 6.2.

Results. We summarize our results on the Amazon Reviews benchmarks in Table 2. Overall, models with sets of discrete item features outperform other competing methods that make use of single-embedding-based item features, while AgenticTagger helps achieve the best overall performance compared to various baselines, demonstrating its effectiveness. Interestingly, the performance improvements are more significant in the Sports and CDs domains, with 8.9%-11.6% improvements over the closest competing model in NDCG@10, whereas the improvement is moderate (2%) in the Beauty domain. Importantly, we observe AgenticTagger achieves strong performance with a highly compact vocabulary, with 2567, 2248, and 934 unique descriptors on Sports, Beauty, and CDs domain, respectively, showcasing its effectiveness in constructing vocabulary from item semantics.

Table 3: Ranking performance comparison between the Item ID baseline and AgenticTagger.

Metric	Item ID (Baseline)	AgenticTagger (Ours)
R@5	0.2655 $\pm$ 0.0037	0.2823 $\pm$ 0.0038
N@5	0.1966 $\pm$ 0.0029	0.1870 $\pm$ 0.0027
R@10	0.3456 $\pm$ 0.0040	0.4300 $\pm$ 0.0041
N@10	0.2224 $\pm$ 0.0029	0.2346 $\pm$ 0.0027
R@20	0.4427 $\pm$ 0.0042	0.6082 $\pm$ 0.0041
N@20	0.2470 $\pm$ 0.0029	0.2796 $\pm$ 0.0025
R@50	0.5761 $\pm$ 0.0041	0.8514 $\pm$ 0.0030
N@50	0.2733 $\pm$ 0.0027	0.3280 $\pm$ 0.0022

5.2 Ranking

Dataset, Baseline, and Evaluation. Besides generative recommendation, another frequent use case for structured item features is ranking (Singh et al., 2024; Zhu et al., 2021). To this end, we evaluate the effect of AgenticTagger descriptors on an anonymized private dataset collected from a news-feed service over a random subset of 47,358 users across an 8-day period. This results in 67,097 items and 206,345 interactions. Following prior work in the literature (He et al., 2017; Kang and McAuley, 2018), we randomly sample 100 negative items per user to reduce computational overhead, and report performance with Recall@ K and NDCG@ K , where $K \in \{5, 10, 20, 50\}$. The backbone recommendation model is a neural feature-aware model aligned with the deployed model in the platform, where we use either one-hot item IDs or AgenticTagger descriptors as IDs during training. For good scalability, we set hierarchy depth to 3 for AgenticTagger, and build the vocabulary using 30,000 randomly subsampled items. We break item collisions using an extra digit of a collision-resolver token (Rajput et al., 2023; Hou et al., 2025a).

Results. We report our results in Table 3; as shown, AgenticTagger brings significant improvements to ranking accuracy compared to raw item IDs, showcasing the effectiveness of AgenticTagger features. Meanwhile, AgenticTagger achieves strong performance with a much lower-cardinality vocabulary, utilizing only 2.7k unique descriptors compared to the full one-hot item table of over 67k item embeddings, achieving effective vocabulary compression, corroborating our observation on the compression effectiveness of AgenticTagger.

On the other hand, an interesting observation is that the item ID baseline shows slightly stronger

performance for NDCG@5. We hypothesize that this is because when the dataset is sufficiently large for good convergence, memorization of item-specific collaborative filtering signals via unique per-item embeddings helps fine-grained ranking among top candidates. This is aligned with prior findings in the literature (Kang et al., 2021), where compressing one-hot item representations into hashed embedding tables sometimes requires (albeit marginally) a performance trade-off. We discuss this idea in the appendix, where in smaller datasets, AgenticTagger yields larger improvements compared to item IDs (Appendix C.2).

6 Ablations and Discussions

6.1 Interpretability and Controllability

Table 4: Performance improvements after single-turn critique on 5000 samples.

Dataset	Metric	Vanilla	+Critique	Improve.
Sports	R@5	0.0318	0.0428	+34.59%
	N@5	0.0206	0.0293	+42.09%
	R@10	0.0500	0.0670	+34.00%
	N@10	0.0264	0.0371	+40.31%
Beauty	R@5	0.0504	0.0864	+71.43%
	N@5	0.0350	0.0602	+72.17%
	R@10	0.0738	0.1284	+73.98%
	N@10	0.0425	0.0738	+73.65%
CDs	R@5	0.0546	0.0666	+21.98%
	N@5	0.0374	0.0445	+19.21%
	R@10	0.0816	0.0990	+21.32%
	N@10	0.0460	0.0549	+19.42%

In this section, we discuss the value AgenticTagger features add to the interpretability and controllability of a recommender system via a single-turn conversational recommendation setting, where a user gives feedback to the recommender system, providing information on the item they would like to interact with next, a widely studied interaction modality in conversational recommendation community (Li et al., 2018; Wang et al., 2022; He et al., 2023; Zhu et al., 2025).

Following recent work on using LLMs for user simulation in recommender systems (Bougie and Watanabe, 2025; Park, 2025; Zhang et al., 2025b; Wang et al., 2024c), we make use of an LLM as a user simulator to evaluate whether AgenticTagger can enable effective user control over recommender system behavior. Specifically, we adopt a communication game setting, where an LLM (Gemini-2.5-flash in practice) observes the descriptions of target items in its prompt, and is asked

to select all potentially relevant top-level (i.e., level 1) features in the vocabulary, which generally contains a small number of keywords (e.g., 16 for the CDs and Vinyl domain, the largest public benchmark we evaluate on). We then generate the next item from a pre-trained generative recommender backbone using beam search following standard practice, except we constrain its search space to be within the allowed top-level features provided by the user simulator.

As shown in Table 4, this constrained recommendation setting increases model performance significantly across datasets, showcasing that AgenticTagger features effectively communicate the nuances of items in the dataset and, importantly, in an interpretable manner, such that the behavior of the corresponding recommender system using AgenticTagger features can be straightforwardly controlled by natural-language-based critiques.

6.2 Comparisons with ID Crossing in Generative Recommendation.

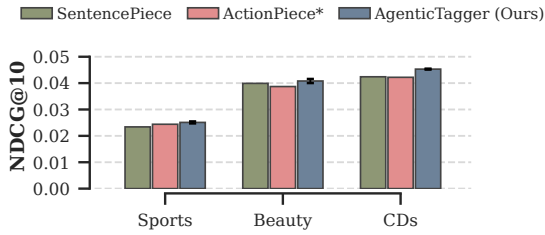


Figure 2: Comparison against feature-crossing-based methods in NDCG@10. For ActionPiece, we compare against its ablated variant *without* inference time model ensembling, to isolate the effect on item-content understanding.

In light of recent advances in demonstrating the benefit of incorporating dataset collaborative filtering statistics directly into the tokenization process via feature crossing (Hou et al., 2025b; Singh et al., 2024), we also compare our method with two recent tokenization methods that merge frequently co-occurring semantic IDs into new IDs based on observed user behavior sequences, SentencePiece (Singh et al., 2024) and ActionPiece (Hou et al., 2025b). To be specific, these methods represent an alternative path towards better features for recommender systems: automated feature crossing based on user behavior. As shown in Figure 2, AgenticTagger remains a strong tokenization method, demonstrating the effectiveness of leveraging LLMs’ content understanding abilities in tokenization in addition to collaborative filtering

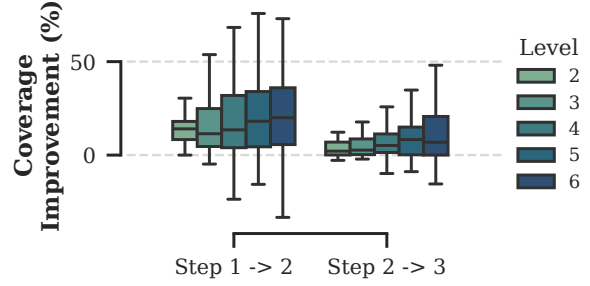


Figure 3: **AgenticTagger effectively refines its vocabulary, covering more observed items after each refinement round.** Here we show the delta of item coverage on Sports dataset (see Appendix C.5 for all datasets), where positive deltas indicates effective improvement.

signals. Meanwhile, we note that AgenticTagger is orthogonal to these crossing-based techniques, and we leave the crossing of AgenticTagger features to future study.

6.3 Effect of Multi-agent Self-refinement

In this ablation, we examine the effect of the iterative refinement loop in AgenticTagger on improving item coverage. To verify its effect, we plot the average percentage differences in item coverage (i.e., items that are assigned at least one feature among the current vocabulary) for each clustering job in Figure 3. As shown, the overall trend is that iterative refinement helps improve item coverage, leading to a reduced number of items that cannot be described by the vocabulary as the number of optimization steps grows. While we report the difference between steps 1 and 2 on the Sports dataset, the trend remains consistent across datasets and optimization steps. On the other hand, we observe that iterative optimization tends to bring less steady coverage improvements to later-layer (i.e., finer-grained) AgenticTagger features, which we hypothesize is due to item differences become more nuances in deeper layers of the hierarchy.

6.4 Scaling AgenticTagger to Larger Datasets

In AgenticTagger, a key decision to make when scaling to larger datasets is the balance between controlling the maximum branching factor b , i.e., how many features an item can be associated with at each granularity level, and how much data to subsample for generating the vocabulary using Algorithm 1. While these two techniques are not mutually exclusive, it is nevertheless helpful to understand which tends to yield better performance.

To investigate this trade-off, we compare the

Dataset	Cor.	Interp.	Precise (%)		Cov.
	(1–5)	(1–5)	Coarse	Fine	
Beauty	4.87±0.44	4.70±0.52	99.0	90.0	97.7
Sports	4.60±0.74	4.42±1.06	100.0	71.0	95.8
CDs	4.11±0.89	4.09±0.88	95.0	95.0	95.6

Table 5: Qualitative evaluation results across three datasets, where AgenticTagger descriptors are of strong semantic quality and item coverage.

quality of AgenticTagger features generated from a random subset of 20k items with that of the original scaling choice we made on the CDs and Vinyl domain, where we constrained the branching factor to 1. That is, we only consider one randomly selected associated feature at each feature level for an item when generating finer-grained features in the next level. We found that, surprisingly, constraining the branching factor yields significantly superior performance compared to subsampling. For the sub-sampling variant, the $N@10$ performance drops from 0.0836 to 0.0787. We hypothesize that constructing the vocabulary from more items helps AgenticTagger capture more diverse patterns, improving downstream performance. However, the performance of the sub-sampling approach nevertheless exceeds that of the second best baseline method, indicating that building the vocabulary on a subset of items is often good enough for strong performance, as we observe in Section 5.2.

6.5 Qualitative Evaluation

Beyond downstream performance, we also directly evaluate the quality of AgenticTagger’s descriptors. Specifically, we conduct evaluation across the following dimensions.

Correctness (Cor). We use LLM-as-a-judge (Gu et al., 2025), asking Gemini-3.1-pro-preview to rate 100 item-and-tag pairs, to judge the tag correctness on a 1–5 scale, where 5 is perfectly correct, 4 is accurate but slightly broad, 3 is fair and related, 2 is poor, and 1 is incorrect. We report the mean and std for the metrics.

Interpretability (Interp). Similar to correctness, we prompt LLM for ratings on 100 item-and-tag pairs, then let LLM rate the tags in its readability and interpretability, reporting the mean and std. In this setting, 5 means perfect readability and 1 means tags of poor quality, similar to the correctness experiment.

For calibration purposes for LLM-as-a-judge

for point-wise scoring in correctness/interpretability evaluation, we manually annotate 50 samples on the Beauty dataset and check agreement with LLM judgement. In this case, human annotation yielded an average correctness score of 4.79 and interpretability score of 4.87, which aligns well with LLM judge decisions.

Preciseness (Precise). We adopt a communication game setting. We sample 100 pairs of similar items sharing the same coarse descriptors (the first 1 or 2 descriptors). We then ask LLM to look at the full set of descriptors of each item as well as item descriptions, and let the LLM guess which set of descriptors belong to which item. Random guess gets 50% accuracy, and precise/un-ambiguous descriptors should make LLM guess which set of tags describe which item perfectly. We report two settings: *Coarse-easy*: item pairs are sampled to share the first tag (e.g., both are “Makeup - Eyes”), and *Fine-grained-hard*: item pairs share the first two tags (e.g., both are “Nail Care & Polish → Nail Files & Buffers”), but differs subtly).

Coverage (Cov). We report the ratio of perfectly assignable items after the highest level descriptors are settled during the vocabulary building stage; these are items that can be covered by at least the highest level descriptors. 100% means full coverage of the item set.

As shown in Table 5, AgenticTagger achieves strong qualitative results across dimensions, which corroborates the performance observed in quantitative evaluations.

7 Conclusion

In this work, we introduce AgenticTagger, a general framework for prompting LLMs to generate natural language item-descriptor representations for recommender systems. We showcase the wide applicability of AgenticTagger across various applications, including generative retrieval, ranking, and controllability-oriented, critique-based recommendation. We also discuss the properties of AgenticTagger, such as its complexity and feature cardinality. In the future, we plan to investigate methods for improving effectiveness by introducing AgenticTagger implementations that consider collaborative filtering signals during the feature generation phase, following the recent success of feature crossing.

Limitations

In this work, our goal is to unleash the power of frontier LLMs to generate discrete text-based features in recommender systems. Thus we focus on the setting of querying API-based LLMs. However, the AgenticTagger framework, especially in its assignment stage, would potentially benefit from distilled, light-weight annotator models, such as a small language model with a classification head. Therefore, we focus primarily on the performance of the structured natural language descriptors in this work, leaving the optimization of AgenticTagger’s inference efficiency to future work (we provide additional discussion on this in Appendix D). We also acknowledge that the public dataset used are uniformly based on the Amazon dataset, and further generalization experiments to new domains would be helpful. Finally, we clarify that our use of the term Agent refer to generalized, prompt-based agents, and for the scope of this work we do not consider agents with memory and toolcalls, which we leave to future explorations.

References

- Nicolas Bougie and Narimawa Watanabe. 2025. [SimUSER: Simulating user behavior with large language models for recommender system evaluation](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 43–60, Vienna, Austria. Association for Computational Linguistics.
- Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. [Wide & deep learning for recommender systems](#). In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA. Association for Computing Machinery.
- Benjamin Coleman, Wang-Cheng Kang, Matthew Fahrback, Ruoxi Wang, Lichan Hong, Ed H. Chi, and Derek Zhiyuan Cheng. 2023. [Unified embedding: Battle-tested feature representations for web-scale ML systems](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, Brian Chu, Zexi Chen, and Manoj Tiwari. 2023. [Leveraging large language models in conversational recommender systems](#). *Preprint*, arXiv:2305.07961.
- Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. [Optimized product quantization](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:744–755.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. [A survey on llm-as-a-judge](#). *Preprint*, arXiv:2411.15594.
- F. Maxwell Harper and Joseph A. Konstan. 2015. [The movielens datasets: History and context](#). *ACM Trans. Interact. Intell. Syst.*, 5(4).
- Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, Xinyang Yi, Lexi Baugher, Baykal Cakici, Ed Chi, Cristos Goodrow, Ningren Han, He Ma, Romer Rosales, Abby Van Soest, and 4 others. 2025. [Plum: Adapting pre-trained language models for industrial-scale generative recommendations](#). *Preprint*, arXiv:2510.07784.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. [Neural collaborative filtering](#). *Preprint*, arXiv:1708.05031.
- Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. [Large language models as zero-shot conversational recommenders](#). In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM ’23*, page 720–730, New York, NY, USA. Association for Computing Machinery.
- Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *TheWebConf*.
- Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. 2025a. Generating long semantic ids in parallel for recommendation. In *KDD*.
- Yupeng Hou, Jianmo Ni, Zhankui He, Naveen Sachdeva, Wang-Cheng Kang, Ed H. Chi, Julian McAuley, and Derek Zhiyuan Cheng. 2025b. [Actionpiece: Contextually tokenizing action sequences for generative recommendation](#). In *Forty-second International Conference on Machine Learning*.
- Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2025. [Recommender ai agent: Integrating large language models for interactive recommendations](#). *ACM Trans. Inf. Syst.*, 43(4).
- Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. [Product quantization for nearest neighbor search](#). *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

- Yangqin Jiang, Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. 2025. [RecLM: Recommendation instruction tuning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15443–15459, Vienna, Austria. Association for Computational Linguistics.
- Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, Suhang Wang, Jiawei Han, and Xianfeng Tang. 2024. Language models as semantic indexers. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.
- Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvish Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. 2025. Generative recommendation with semantic ids: A practitioner’s handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*.
- Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H. Chi. 2021. [Learning to embed categorical features without embedding tables for recommendation](#). In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD ’21*, page 840–850, New York, NY, USA. Association for Computing Machinery.
- Wang-Cheng Kang and Julian McAuley. 2018. [Self-attentive sequential recommendation](#). *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206.
- Leonard Kaufman and Peter J. Rousseeuw. 1987. [Clustering by means of medoids](#).
- Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. [Concept bottleneck models](#). In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR.
- Raymond Li, Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, page 9748–9758, Red Hook, NY, USA. Curran Associates Inc.
- Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025a. [Improving llm-powered recommendations with personalized information](#). *SIGIR ’25*, page 2560–2565, New York, NY, USA. Association for Computing Machinery.
- Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. [Once: Boosting content-based recommendation with both open- and closed-source large language models](#). In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM ’24*, page 452–461, New York, NY, USA. Association for Computing Machinery.
- Weihao Liu, Zhaocheng Du, Haiyuan Zhao, Wenbo Zhang, Xiaoyan Zhao, Gang Wang, Zhenhua Dong, and Jun Xu. 2025b. [Inference computation scaling for feature augmentation in recommendation systems](#). *Preprint*, arXiv:2502.16040.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA. Curran Associates Inc.
- Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Taffjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. 2024. [CLIN: A continually learning language agent for rapid task adaptation and generalization](#). In *First Conference on Language Modeling*.
- Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. [Image-based recommendations on styles and substitutes](#). *SIGIR ’15*, page 43–52, New York, NY, USA. Association for Computing Machinery.
- Choongwon Park. 2025. [Llm as user simulator: Towards training news recommender without real user interactions](#). In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’25*, page 3080–3084, New York, NY, USA. Association for Computing Machinery.
- Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. 2024. [TopicGPT: A prompt-based topic modeling framework](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2956–2984, Mexico City, Mexico. Association for Computational Linguistics.
- 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(1).
- 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 Maheswaran Sathiamoorthy. 2023. [Recommender systems with generative retrieval](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.

- Yankun Ren, Zhongde Chen, Xinxing Yang, Longfei Li, Cong Jiang, Lei Cheng, Bo Zhang, Linjian Mo, and Jun Zhou. 2024. [Enhancing sequential recommenders with augmented knowledge from aligned large language models](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24*, page 345–354, New York, NY, USA. Association for Computing Machinery.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09*, page 452–461, Arlington, Virginia, USA. AUAI Press.
- Kainan Shi, Peilin Zhou, Ge Wang, Han Ding, and Fei Wang. 2025. [What matters in llm-based feature extractor for recommender? a systematic analysis of prompts, models, and adaptation](#). *Preprint*, arXiv:2509.14979.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Yubo Shu, Haonan Zhang, Hansu Gu, Peng Zhang, Tun Lu, Dongsheng Li, and Ning Gu. 2023. [Rah! recsys-assistant-human: A human-centered recommendation framework with llm agents](#). *Preprint*, arXiv:2308.09904.
- Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, Ed Chi, and Xinyang Yi. 2024. [Better generalization with semantic ids: A case study in ranking for recommendations](#). In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys '24*, page 1039–1044, New York, NY, USA. Association for Computing Machinery.
- Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. 2025. [Concept bottleneck large language models](#). In *The Thirteenth International Conference on Learning Representations*.
- Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. [Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer](#). In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19*, page 1441–1450, New York, NY, USA. Association for Computing Machinery.
- Param Thakkar and Anushka Yadav. 2024. [Personalized recommendation systems using multi-modal, autonomous, multi agent systems](#). *Preprint*, arXiv:2410.19855.
- Mahsa Valizadeh, Xiangjue Dong, Rui Tuo, and James Caverlee. 2025. [Language models as semantic augmenters for sequential recommenders](#). *Preprint*, arXiv:2510.18046.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 6309–6318, Red Hook, NY, USA. Curran Associates Inc.
- Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W. White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. 2024. [Tnt-llm: Text mining at scale with large language models](#). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, page 5836–5847, New York, NY, USA. Association for Computing Machinery.
- Jianling Wang, Haokai Lu, James Caverlee, Ed H. Chi, and Minmin Chen. 2024a. [Large language models as data augmenters for cold-start item recommendation](#). In *Companion Proceedings of the ACM Web Conference 2024, WWW '24*, page 726–729, New York, NY, USA. Association for Computing Machinery.
- Jianling Wang, Haokai Lu, James Caverlee, Ed H. Chi, and Minmin Chen. 2024b. [Large language models as data augmenters for cold-start item recommendation](#). In *Companion Proceedings of the ACM Web Conference 2024, WWW '24*, page 726–729, New York, NY, USA. Association for Computing Machinery.
- Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. 2024c. [User behavior simulation with large language model-based agents](#). *Preprint*, arXiv:2306.02552.
- Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. [Deep & cross network for ad click predictions](#). In *Proceedings of the ADKDD'17, ADKDD'17*, New York, NY, USA. Association for Computing Machinery.
- Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. [Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems](#). In *Proceedings of the Web Conference 2021, WWW '21*, page 1785–1797, New York, NY, USA. Association for Computing Machinery.
- Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024d. Learnable item tokenization for generative recommendation. In *International Conference on Information and Knowledge Management*.

- Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Towards unified conversational recommender systems via knowledge-enhanced prompt learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1929–1937.
- Yancheng Wang, Ziyang Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Yanbin Lu, Xiaojiang Huang, and Yingzhen Yang. 2024e. [RecMind: Large language model powered agent for recommendation](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4351–4364, Mexico City, Mexico. Association for Computational Linguistics.
- Yu Xia, Sungchul Kim, Tong Yu, Ryan A. Rossi, and Julian McAuley. 2026. [Multi-agent collaborative filtering: Orchestrating users and items for agentic recommendations](#). *Preprint*, arXiv:2511.18413.
- Zhouhang Xie, Tushar Khot, Bhavana Dalvi Mishra, Harshit Surana, Julian McAuley, Peter Clark, and Bodhisattwa Prasad Majumder. 2025. [Latent factor models meets instructions: Goal-conditioned latent factor discovery without task supervision](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11114–11134, Albuquerque, New Mexico. Association for Computational Linguistics.
- Zhouhang Xie, Bodhisattwa Prasad Majumder, Mengjie Zhao, Yoshinori Maeda, Keiichi Yamada, Hiromi Wakaki, and Julian McAuley. 2024. [Few-shot dialogue strategy learning for motivational interviewing via inductive reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13207–13219, Bangkok, Thailand. Association for Computational Linguistics.
- Yuki Yada and Hayato Yamana. 2024. [News recommendation with category description by a large language model](#). *Preprint*, arXiv:2405.13007.
- Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. 2024. [Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58484–58509. PMLR.
- Chao Zhang, Shiwei Wu, Haoxin Zhang, Tong Xu, Yan Gao, Yao Hu, and Enhong Chen. 2024a. [Notellm: A retrievable large language model for note recommendation](#). *WWW '24*, page 170–179, New York, NY, USA. Association for Computing Machinery.
- Chao Zhang, Haoxin Zhang, Shiwei Wu, Di Wu, Tong Xu, Xiangyu Zhao, Yan Gao, Yao Hu, and Enhong Chen. 2025a. [Notellm-2: Multimodal large representation models for recommendation](#). In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, page 2815–2826, New York, NY, USA. Association for Computing Machinery.
- Jizhi Zhang, Keqin Bao, Wenjie Wang, Yang Zhang, Wentao Shi, Wanhong Xu, Fuli Feng, and Tat-Seng Chua. 2024b. [Prospect personalized recommendation on large language model-based agent platform](#). *Preprint*, arXiv:2402.18240.
- 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 *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19*, page 4320–4326. AAAI Press.
- Yabin Zhang, Wenhui Yu, Erhan Zhang, Xu Chen, Lantao Hu, Peng Jiang, and Kun Gai. 2024c. [Recgpt: Generative personalized prompts for sequential recommendation via chatgpt training paradigm](#). *ArXiv*, abs/2404.08675.
- Zhiyang Zhang, Junda She, Kuo Cai, Bo Chen, Shiyao Wang, Xinchun Luo, Qiang Luo, Ruiming Tang, Han Li, Kun Gai, and Guorui Zhou. 2026. [Unleashing the native recommendation potential: Llm-based generative recommendation via structured term identifiers](#). *Preprint*, arXiv:2601.06798.
- Zijian Zhang, Shuchang Liu, Ziru Liu, Rui Zhong, Qingpeng Cai, Xiangyu Zhao, Chunxu Zhang, Qidong Liu, and Peng Jiang. 2025b. [Llm-powered user simulator for recommender system](#). In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'25/IAAI'25/EAAI'25*. AAAI Press.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2025. [A survey of large language models](#). *Preprint*, arXiv:2303.18223.
- Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten de Rijke. 2024. [Let me do it for you: Towards llm empowered recommendation via tool learning](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24*, page 1796–1806, New York, NY, USA. Association for Computing Machinery.
- Ruiqi Zhong, Heng Wang, Dan Klein, and Jacob Steinhardt. 2024. [Explaining datasets in words: Statistical models with natural language parameters](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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 *Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20*, page 1893–1902, New York, NY, USA. Association for Computing Machinery.

Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. [Open benchmarking for click-through rate prediction](#). In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 2759–2769. ACM.

Yaochen Zhu, Harald Steck, Dawen Liang, Yinhan He, Nathan Kallus, and Jundong Li. 2025. [LLM-based conversational recommendation agents with collaborative verbalized experience](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 2207–2220, Suzhou, China. Association for Computational Linguistics.

A Further Ablations

A.1 Comparison with Free-form Generation in Generative Recommendation

As discussed earlier, it is difficult to directly use free-form generated features for generative recommendation, as this method assigns sets of features to items with no constraints on vocabulary choice. To illustrate this point, we discuss an earlier attempt aimed at directly using free-form generated features as semantic IDs in the Sports domain.

We first prompt an LLM to generate free-form descriptors and treat all ever-generated descriptors as part of the valid vocabulary. We then train a dual-encoder model that predicts which descriptors are generated for each item among the possible descriptor vocabulary (i.e., every descriptor that has been generated by the LLM), following prior work in LLM-based document topic mining (Xie et al., 2025), where positive signals come from LLM generations. This process ensures that the potential match between each descriptor-item pair is sufficiently explored, even when the LLM did not generate a particular descriptor initially. Finally, we perform retrieval-augmented descriptor generation with another pass of prompting, except that the LLM this time is asked to select all relevant descriptors from the top 30 suggested descriptors predicted by the aforementioned dual-encoder model. This process yields a maximum of 27, a minimum of 0, and an average of 12.25 descriptors per item.

Table 6: Comparison on AgenticTagger performance with different LLMs on generative recommendation.

Backbone	Sports		Beauty		CDs	
	N@5	N@10	N@5	N@10	N@5	N@10
Gemini-1.5	0.0184	0.0237	0.0319	0.0397	0.0349	0.0439
Gemini-2.5	0.0194	0.0251	0.0332	0.0408	0.0358	0.0453

Due to the large number of descriptors per item, we then prune the descriptors assigned to each item via two methods. (1) Frequency: we split the tags by their occurrence across the corpus into four equal bins, ignoring tags with frequencies smaller than 10 or larger than 2,000, since these are overly rare or common. Each item’s descriptors are then selected by taking the most frequent tag from each popularity bin, where available, creating a coarse-to-fine tag sequence similar to semantic IDs. (2) K-means: we select 5,000 centroids by running K-means clustering over the text embeddings of the descriptor vocabulary, and swap the descriptors with their assigned centroid IDs, reducing the number of descriptors per item.

However, this framework results in poor overall performance: NDCG@10 for frequency-based pruning is 0.185, and for K-means-based pruning, it is 0.184, neither of which is comparable to modern semantic ID performance. This demonstrates, in contrast, the effectiveness of AgenticTagger in generating hierarchically structured, semantic-ID-like descriptors.

A.2 Comparison with Clustering of Free-form Generated Descriptors.

Another natural alternative to AgenticTagger is generating free-form descriptors for each item using an LLM, followed by clustering descriptors, and then replacing each tag with its corresponding cluster medoid. To evaluate this baseline, we prompted an LLM (Gemini-3.5-flash¹) to generate three free-form descriptors for each item in the Beauty domain. We embedded these descriptors into a latent space, extracted 900 medoids, and mapped each original descriptor to its nearest medoid.

To ensure a fair comparison, we evaluated these clustered tags in a generative recommendation setting using the exact training paradigm, scripts, and random seeds detailed for our main experiment. We find that AgenticTagger demonstrates significantly

¹To maintain consistency with the Gemini-2.5-flash model originally used, which was subsequently deprecated.

Table 7: Case Study: AgenticTagger and free-form generation as features. Item content rewritten to preserve privacy.

Item Context	Free-form	AgenticTagger
Ground Truth	Skipper Limited,	L0: Business & Finance
<i>Transformer stock, 820% 3-yr returns</i>	Order Book Growth, Power Transmission.	L1: Stock Market L2: Individual Performance
User History		L0: Business & Finance
(1) <i>Defence bidder, 7k% 5-yr returns</i>	(1) Defence PSU, DRDO Bidding.	L1: Stock Market
(2) <i>Industrial PSU, 200% 5-mo returns</i>	(2) PSU Stocks, Market Returns.	L2: Individual Performance, Order Book Analysis

stronger downstream performance: the clustering baseline achieves an NDCG@10 of 0.0354, compared to 0.0408 for the equivalent 3-descriptor version of AgenticTagger. This $\sim 13\%$ performance degradation indicates that our proposed multi-LLM workflow produces higher-quality, more performant features that cannot be replaced by simple post-hoc clustering.

A.3 Performance with Different LLMs

In our generative recommendation experiments on public benchmarks, we opted to use the Gemini-2.5 family of models, as they were released closest to the time the experiments were conducted. However, to investigate if our framework is robust to the choice of base models, we also experimented with Gemini-1.5-flash and Gemini-1.5-pro as annotator and architect LLMs, respectively. In our experiments, we find both models yield competitive performance, while Gemini-2.5 maintains a slight advantage across datasets, likely due to its stronger overall abilities in text generation and understanding.

A.4 Case Study

Finally, to understand the improvement brought by AgenticTagger compared to free-form generation, we took an example from our private dataset to analyze the effect of features from both frameworks on connecting the user’s recent interests to the ground-truth item. As shown, the free-form-generated features, while of generally high coherence, failed to generalize across items due to random surface phrase choices; in contrast, AgenticTagger successfully identified the user’s broad interest in the stock

market and detailed interest in stock performance. While in both cases LLMs are effective at extracting general user interest in stocks, AgenticTagger is the only method that can organically connect user history to the ground-truth interacted item, demonstrating the value of vocabulary-constrained, structured descriptor generation.

B Implementation Details

In this section, we cover our implementation details on public benchmark experiments disclosed in this work.

B.0.1 Baselines and pre-processing.

We directly reuse the data processing implementation open-sourced by Hou et al. 2025b for fair and straightforward comparison. We reuse the reported baseline performance in Hou et al. 2025b, where baseline results for BERTRec, SASRec, FDSA, and S³-Rec on the Sports and Beauty domains are taken directly from each respective work, and the rest are re-implementations by Hou et al..

B.0.2 Hyperparameter for GR Training.

We train the generative recommender model from scratch with AgenticTagger as semantic IDs for a maximum of 200 epochs, breaking the training if there are no improvements in $N@10$ for 20 epochs. Following Hou et al. 2025b, we set training batch size to 256 and tune learning rate from $\{1 \times 10^{-3}, 3 \times 10^{-3}, 5 \times 10^{-3}\}$ and weight-decay from $\{0.07, 0.1, 0.15, 0.2\}$. We use a standard T5 (Raffel et al., 2020) backbone trained from scratch as the generative recommendation model, following prior works (Rajput et al., 2023; Hou et al., 2025b), where the non-embedding portion of the model totals to 13M parameters. We conduct three repeated experiments with random seeds $\{2025, 2026, 2027\}$, and report the test set performance with the model checkpoints with best average $N@10$ across runs. We train the GR on NVIDIA A100 40GB gpu.

B.0.3 GR Inference.

We use standard auto-regressive decoding with a beam size of 50 for inference following baseline methods (Hou et al., 2025b). However, since AgenticTagger can produce variable length semantic IDs, we do not sample from the base generative recommender for a fixed number of steps; instead, we generate until we reach the end-of-sequence token.

Table 8: Statistics of public datasets used. “Avg.” denotes the average sequence length.

Datasets	#Users	#Items	#Interactions	Avg.
Sports	35,598	18,357	260,739	8.32
Beauty	22,363	12,101	176,139	8.87
CDs	75,258	64,443	1,022,334	14.58

Table 9: Hyperparameter settings for each dataset.

Hyperparameter	Sports	Beauty	CDs
learning_rate	0.005	0.005	0.01
n_semid	6+1	3+1	6+1
vocab_size	2487+80	1103+1145	742+192
warmup_steps	10,000	10,000	10,000
dropout_rate	0.1	0.1	0.1
weight_decay	0.07	0.10	0.07
beam_size	50	50	50
num_layers	4	4	4
d_model	128	128	256
d_ff	1,024	1,024	2,048
num_heads	6	6	6
d_kv	64	64	64
optimizer	adamw	adamw	adamw
lr_scheduler	cosine	cosine	cosine
train_batch_size	256	256	256
max_epochs	200	200	200
early_stop_patience	20	20	20

The generated sequences are then ranked by their beam scores.

B.0.4 AgenticTagger

By default, we set the initial cluster size to 15 across all datasets and the number of refinement cycles to 3 (while the architect LLM can choose to exceed this number). The minimum cluster size for sub-clustering is set to 30, i.e., any group of more than 30 items would trigger the procedure for producing sub-category descriptors for the said group. We break the iterative refinement loop if the item coverage for any cluster is greater than 95 or if the refiner LLM fails to generate more than 20 error cases. Due to the large number of items in the CDs domain, we set the sub-clustering threshold to 100 for the branching-controlled version of the algorithm and 500 for the subsampled variant. While the final vocabulary size of AgenticTagger varies due to the stochasticity of program execution, we report the base vocabulary and the number of conflict resolver descriptors (i.e., “stub” descriptors used when two items hash to the same set of AgenticTagger tags) as “base descriptor + resolvers”

in Table 9 under “vocab_size.” As shown, AgenticTagger is significantly more efficient than its semantic ID counterparts, using fewer than 3,000 words across the board on academic benchmarks.

C Further Related Works

ML Models with Natural Language Parameters.

Beyond recommender systems, there have been continuing efforts to introduce parameters tied to explicit natural language concepts into ML models, e.g., concept-bottleneck models (Koh et al., 2020; Sun et al., 2025), typically for their improved interpretability (Zhong et al., 2024). However, this stream of work generally does not aim for improved accuracy and frequently assumes there exist sets of concepts (in our case, features) that are readily available (Koh et al., 2020). The most relevant work to ours is perhaps the label-free concept bottleneck model (Sun et al., 2025), which prompts an LLM with a task description for concepts. However, our aim is to infer a set of features from large sets of items for recommendation, the nature of which is typically difficult to infer from the task description itself, unlike domains such as image classification with a fixed set of labels.

LLM for Text Mining At Scale

Another relevant stream of work to AgenticTagger, as discussed throughout the paper, consists of recent advances in leveraging LLMs for scalable text mining, typically manifesting as topic/latent theme mining (Xie et al., 2025; Pham et al., 2024) or taxonomy mining (Wan et al., 2024). However, these frameworks generally do not consider downstream applications and usually impose sequential dependencies on the interaction between LLMs and text corpora (Wan et al., 2024; Pham et al., 2024), whereas AgenticTagger uses parallelized operations, making scaling to recommendation-scale item corpora feasible.

LLM-generated Semantic IDs

Concurrent to this

work, we note there are also recent explorations of LLM-generated natural language semantic IDs, dubbed Term IDs (Zhang et al., 2026), where one prompts an LLM for free-form generated semantic IDs over a batch of similar items for adapting pre-trained LLMs for recommendation. To this end, Term IDs impose fewer constraints on the vocabulary itself, since a pre-trained LLM is able to generalize information across semantically similar Term IDs. Indeed, in our exploratory experiment, we find that under a fair comparison with the same hyper-parameter search procedure and on the same from-scratch backbone generative recommendation model (as described in Table 9), Term IDs only achieve an NDCG@10 of 0.017 on Sports dataset, confirming our hypothesis that less structured IDs imposes stricter constraints on the recommendation model used. On the other hand, AgenticTagger aims to develop a descriptor vocabulary with minimal constraints on the downstream model. To illustrate this point, AgenticTagger enables a 2-layer transformer model (the same as Tiger (Rajput et al., 2023)) to perform better than the 0.4B variant of LLM-based GR with Term IDs on the Sports and Beauty domains, showcasing the effectiveness of AgenticTagger.

C.1 Sample Prompts (Sports Domain)

Architect - Generate Initial Vocabulary

You are an expert taxonomy architect. Your task is to create a set of fine-grained sub-categories. The items you are categorizing all belong to the parent category: "{parent_rule_description}".

Your new categories MUST be more specific subdivisions of this parent category. Do not simply repeat the parent category's name. For example, if the parent is "Firearm Accessories", your sub-categories should be "Optics", "Holsters", "Cleaning Kits", etc., not "Firearm Accessories" again.

Primary Goal: Create a set of ~{n_target_rules} mutually exclusive sub-categories that logically partition the parent.

{context_prompt}

Input Data:

Below are diverse product examples. Use them to understand the product landscape you need to partition.

{sample_text}

Task & Strict Output Format:

Devise a set of categories. For each, provide a "name" and a "description" with "INCLUDES" and "EXCLUDES" clauses. Respond ONLY with a single JSON object: {"categories": [{...}]}.

Your JSON Response:

Annotator

{context_prompt}

Above, I have provided the parent category and some product examples.

Your Goal:

Propose a list of exactly {n_target_rules} sub-categories.

Constraint - Overlap:

Sub-categories must be mutually exclusive.

Constraint - Coverage:

Together, they should cover most common items found in "{parent_rule_description}".

Constraint - Specificity:

Do NOT create a "Miscellaneous" or "Other" category. Every category must have a specific, meaningful name.

Constraint - Style:

Use professional, industry-standard terminology.

Output Format:

Your response must be a JSON object with the following structure:

```
{
  "categories": [
```

```
{
  "name": "Category Name",
  "description": "Definition of what
  is included",
  "includes": ["Example 1", "Example
  2"],
  "excludes": ["Non-example 1"]
}
```

Annotator - Generating Error Feedbacks

You are a data analyst specializing in taxonomy quality control. Analyze a cluster of `{len(ticket_cluster)}` uncategorized products and synthesize a single, structured change proposal.

Existing Category Rules:
`{existing_rules_text}`

Analysis of Uncategorized Items:
`{ticket_examples_text}`

Task: Decide the most logical action and formulate a proposal. Respond ONLY with a single JSON object. Your response MUST strictly follow one of the formats below.

1. To create a new category:

```
{ "change_type":
  "CREATE_NEW_CATEGORY",
  "problem_summary": "<summary>",
  "suggested_change":
  { "new_rule_description": "New
  Category Name: INCLUDES: ...
  EXCLUDES: ..." }}
```

2. To expand an existing category (THIS IS A VERY STRICT FORMAT):
The "refined_description" MUST be a single complete string, starting with a name, then "INCLUDES", then "EXCLUDES".

GOOD EXAMPLE:

```
{
  "change_type":
```

```
"EXPAND_EXISTING_CATEGORY",
  "problem_summary": "The current
  'Outdoor Gear' rule is too generic
  and should explicitly include
  tactical accessories.",
  "suggested_change": {
    "rule_id_to_refine":
    "rule_a4368cef",
    "refined_description": "Outdoor &
    Tactical Gear: INCLUDES: Tents,
    backpacks, sleeping bags, and
    tactical accessories like gloves,
    belts, and pouches. EXCLUDES:
    Specialized sporting equipment,
    firearms, and knives."
  }
}
```

3. To ignore outliers:

```
{ "change_type":
  "IGNORE_AS_OUTLIERS",
  "problem_summary": "<summary>",
  "suggested_change": { "reason":
    "<reason>" }}
```

Your JSON Response:

Architect: Taxonomy Refinement

You are a senior taxonomy manager. Review the following change proposals and decide whether to approve or reject each. Your goal is a clean, non-overlapping taxonomy. But if the proposal is reasonable, you should try to accomodate that. You should also reject proposals that are out-of-place by common sense. Respond ONLY with a JSON list of objects, each with "proposal_id", "decision" ('APPROVED' or 'REJECTED'), and "reasoning".

Proposals for Review:
`{proposals_text}`

Your JSON Decision List:

C.2 Ranking Performance on Smaller Dataset

In this section, we discuss further performance comparisons in ranking experiments, where we re-

Table 10: Ranking performance comparison between Item ID baseline and AgenticTagger.

Metric	Item ID	AgenticTagger
R@5	0.0833 $\pm$ 0.0068	0.1205 $\pm$ 0.0080
N@5	0.0589 $\pm$ 0.0051	0.0853 $\pm$ 0.0060
R@10	0.1163 $\pm$ 0.0079	0.1751 $\pm$ 0.0093
N@10	0.0695 $\pm$ 0.0052	0.1028 $\pm$ 0.0061

port both Recall and NDCG in Table 10. This subset contains 1,791 users, 15,830 items, and 47,260 interactions, making the collaborative filtering signal less salient due to the smaller dataset size. As shown, our method consistently outperforms the ID-based method by a larger margin than in the main experiment, verifying our intuition: when the dataset is large and the collaborative filtering signal is more stable, assigning more item-specific parameters helps in memorizing item-specific information, sometimes leading to better ranking performance. Conversely, when the dataset is sparse, content-aware structured discrete features shared among items help convergence significantly, making AgenticTagger features potentially applicable to cold-start settings, where they lead to better generalization.

C.3 Cardinality of Free-form Generation

To investigate whether free-form generation indeed leads to vocabulary explosion, we conduct an ablation study where we prompt an LLM to generate three keywords over the private dataset discussed in Section 5. Indeed, there were approximately 90k unique keywords generated over 210k documents; however, only 21k (or 24%) of the descriptors appeared in more than one item. Consequently, these features are inapplicable for natively acting as compact, ID-like features, making downstream modeling challenging. On the other hand, descriptors generated by AgenticTagger have a much higher utilization rate (87%) across a vocabulary of over 2k, demonstrating their effectiveness.

C.4 Tags Generated

We show qualitative examples of the discovered vocabulary in Table 11. For presentation purposes, we pick two descriptors per level but only expand on one of them until no valid children are found. Note that in AgenticTagger, the depth of each hierarchy path can be of variable length, as shown in

the CDs domain example. As showcased in the table, AgenticTagger is able to find clear separations of products, especially towards the earlier layers of the hierarchy (e.g., “Skincare” and “Haircare” products). However, towards the end of the hierarchy, the boundary between products becomes fuzzy, and it is more challenging to come up with clean, disentangled separation rules (e.g., “Specialty Function” and “Ceramic Plate Irons”), making it harder to enforce a nominal constraint on features. While we partially address this issue by instructing the LLM to select the best-fitting descriptor for each product when multiple are technically correct, developing methods to fundamentally address such disentanglement issues remains an open problem. Nevertheless, even at later layers, AgenticTagger can still discover meaningful descriptors that separate the products, such as sleeping bags by their applicable temperature range, as shown in the Sports domain.

C.5 Effect of Multi-agent Self-refinement on Other Domains

We showcase the improvements in coverage by reporting the distribution of delta changes in the coverage rate in Figure 4. The trend is consistent across domains: (1) our designed multi-agent self-refinement loop effectively refines the vocabulary, improving item coverage; and (2) the trend of improvement (i.e., delta greater than zero) becomes less stable in later layers, possibly because it is more challenging to generate disentangled descriptors, as qualitatively analyzed in Appendix C.4. Regardless, the Wilcoxon signed-rank test shows that improvements are significant (i.e., delta greater than zero) with p-values smaller than 0.05, except for levels five and six in the CDs domain, corroborating the general effectiveness of the self-refinement mechanism.

D Additional Discussions

D.1 Further Discussion on the Challenge for Generating Structured Fix-vocabulary Features with LLMs

In this paragraph, we provide concrete instances on the challenge of fixed-vocabulary feature generation with LLMs, particularly for the case when the feature is nominal. For example, it is challenging to enforce mutual exclusivity between descriptions from free-form generation (Appendix A.1), making these descriptors unsuitable for our use case. Further, a good set of text descriptors should

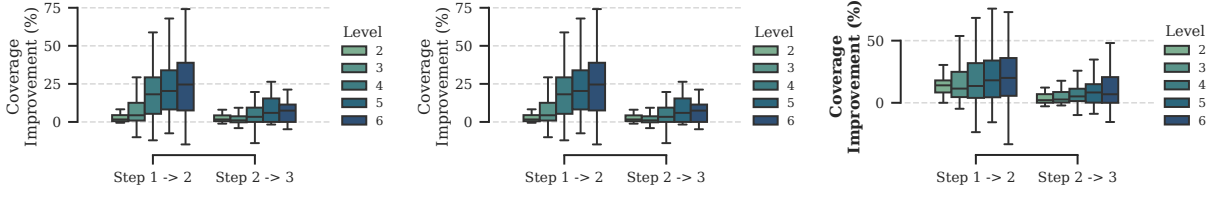


Figure 4: Coverage improvements across domains. We discuss further details in Appendix C.5.

Table 11: Discovered Vocabulary (Sub-sampled) Across Domains

Beauty Domain	Music (CDs) Domain	Sports Domain
L1: Skincare - Facial Treatments	L1: Jazz Music	L1: Fitness & Exercise Equipment
L1: Haircare - Treatments & Styling	L1: Electronic Dance Music (EDM)	L1: Camping & Hiking Gear
L2: Hair & Scalp Treatments	L2: Smooth Jazz	L2: Shelter Systems
L2: Hair Styling Tools & Acc.	L2: Free Jazz & Avant-Garde	L2: Sleeping Systems
L3: Hair Dryers	L3: Interdisc. Avant-Garde	L3: Sleeping Bag Liners
L3: Flat Irons & Straighteners	L3: Free Improv. & Avant-Garde	L3: Sleeping Bags
L4: Travel & Mini Flat Irons	L4: Free Jazz (Post-Bebop)	L4: Sleeping Bag Shapes
L4: Professional Salon Irons	L4: Avant-Garde (Exp.)	L4: Temp.-Rated Bags
L5: Specialty Function	L5: Atonal Exploration	L5: Mild to Cool Bags
L5: Ceramic Plate Irons	L5: Genre-Infused Exp.	L5: Cold Weather Bags
L6: Adv. Plate Tech.		L6: 10-30 Degree Bags
L6: Ionic Technology		L6: 0-Degree Rated Bags

ideally capture meaningful frequent patterns in the item corpus (Xie et al., 2025; Rajput et al., 2023; Singh et al., 2024). Yet, recommender system corpora generally exceed the context limit of LLMs (McAuley et al., 2015; Harper and Konstan, 2015), and thus it is challenging to incorporate the abstract constraint of generating descriptors that meaningfully highlight item similarities and differences w.r.t. other items, as this requires a global view of the entire item corpus. Finally, as LLMs can pick any arbitrary surface form (e.g., "Blues music" and "Blues great hits") for the same underlying concept in free-form generation, such an approach often leads to vocabulary explosion, where most generated descriptors occur only once across the item corpus, as we discuss in Appendix C.3. This renders many features unusable without further processing, compromising downstream recommendation performance (Appendix A.1).

D.2 Wall-clock and Cost Analysis

To understand the practical cost, we benchmark the runtime for CDs and Vinyl dataset, the largest public dataset to use in our experiments. We use the branching-constrained algorithm variant, and opt to build the vocabulary from *all* items, as this is the better performing method as discussed in Section 6.4. The total runtime for the algorithm is 7 hours, for our modest implementation that only allows for a maximum parallel API call of 32, show-

casing the efficacy of AgenticTagger.

Since the Gemini-2.5 family of models are deprecated, we opt to report costs using Gemini-3.1-flash-lite and Gemini-3-flash-preview as the annotator and architect model, respectively. The run-time of the algorithm is similar to that of the Gemini-2.5 family, totaling 8.78 hours with maximum parallel calls of 16, and a total cost of 180 dollars.

We note that the wall-clock runtime is measured through our implementation that uses a fairly modest concurrency (e.g., an API parallelism of 32), and in practice, this could be significantly sped-up by increasing the maximum number of allowed concurrent API calls.

E LLM Usage and AI Assistance

We use LLM for grammar suggestion, language improvements, and low-novelty texts (e.g., making a table) in parts of this manuscript. The low-novelty portion of the experiments of this work benefits from AI coding assistance.

F Notes on Private Dataset

We collect and use interaction information strictly adhering to relevant platform rules, following numerous prior arts in recommender system research that involves industrial datasets (Singh et al., 2024; He et al., 2025; Wang et al., 2017, 2021; Cheng

Table 12: Summary of Notations

Notation	Description
$\mathcal{U}, \mathcal{I}$	Sets of users and items
S_u	Interaction history for user $u \in \mathcal{U}$
X_i	Raw attributes for item $i \in \mathcal{I}$
Π, π	LLMs, and a specific instance
$\mathcal{E}$	Set of natural language error feedback
A	Descriptor feature vocabulary
a	A specific feature descriptor ($a \in A$)
$g(\cdot)$	Transformation mapping attributes to a subset of A
f_θ	Recommender system parameterized by θ

Table 13: Impact of the backbone model used in AgenticTagger on downstream performance, Recall and NDCG.

Backbone LLM	Sports				Beauty				CDs			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Gemini-1.5	0.0284	0.0184	0.0448	0.0237	0.0469	0.0319	0.0710	0.0397	0.0531	0.0349	0.0812	0.0439
Gemini-2.5	0.0299	0.0194	0.0474	0.0251	0.0492	0.0332	0.0728	0.0408	0.0544	0.0358	0.0836	0.0453

et al., 2016). We do not use any PII in our experiment.