UNIFYING GENERATIVE AND DENSE RETRIEVAL FOR SEQUENTIAL RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential dense retrieval models utilize advanced sequence learning techniques to compute item and user representations, which are then used to rank relevant items for a user through inner product computation between the user and all item representations. However, this approach requires storing a unique representation for each item, resulting in significant memory requirements as the number of items grow. In contrast, the recently proposed generative retrieval paradigm offers a promising alternative by directly predicting item indices using a generative model trained on semantic IDs that encapsulate items' semantic information. Despite its potential for large-scale applications, a comprehensive comparison between *generative retrieval* and *sequential dense retrieval* under fair conditions is still lacking, leaving open questions regarding performance, storage, and computation trade-offs. To address this gap, we conduct a thorough comparison of both approaches under identical conditions and propose LIGER (LeveragIng dense retrieval for GEnerative Retrieval), a hybrid model that combines the strengths of these two widely used paradigms. Our proposed model seamlessly integrates sequential dense into generative retrieval, effectively addressing performance disparities and improving cold-start item recommendation. This approach demonstrates significant improvements in both efficiency and effectiveness for recommendation systems.

1 INTRODUCTION

Sequential recommendation methods (Kang & McAuley, 2018b; Zhou et al., 2020), have predominantly relied on advanced sequential modeling techniques (Hochreiter & Schmidhuber, 1997; Vaswani et al., 2017; Radford et al., 2019) to learn dense embeddings for each item and user. These methods, often referred to as *dense retrieval*, involve computing the maximal inner product between user and item embeddings to identify the most relevant items for a user. However, this approach requires comparing every item in the dataset during the retrieval stage, which can be computationally expensive as the number of items grows. Furthermore, each item must be represented by a unique embedding, which needs to be learned and stored, adding to the complexity.

In contrast, *generative retrieval* (Rajput et al., 2024) is a new approach, which deviates from the traditional embedding-centric paradigm. Instead of generating embeddings, this approach utilizes a generative model to directly predict the item index. To better capture the sequential patterns within item interactions, items are indexed by "semantic IDs" (Lee et al., 2022a), which encapsulate their semantic characteristics. During the recommendation process, the model employs beam search decoding to predict the semantic ID (SID) of the next item based on the user's previous interactions. This method not only reduces the need for storing individual item embeddings but also enhances the ability to capture deeper semantic relationships within the data.

The *generative retrieval* paradigm is well-positioned for future scaling in industrial recommendation systems (Singh et al., 2023), offering significant savings in storage and inference time. However, while recent works continue to advance the dense retrieval paradigm (Hou et al., 2022c), generative retrieval methods are increasingly being integrated with pretrained models such as LLMs (Cao et al., 2024b) to improve item recommendation. Despite these advancements, there is a notable lack of direct comparisons under equivalent conditions, raising questions about which paradigm excels in performance when considering storage and computation trade-offs. In this study, we compare se-

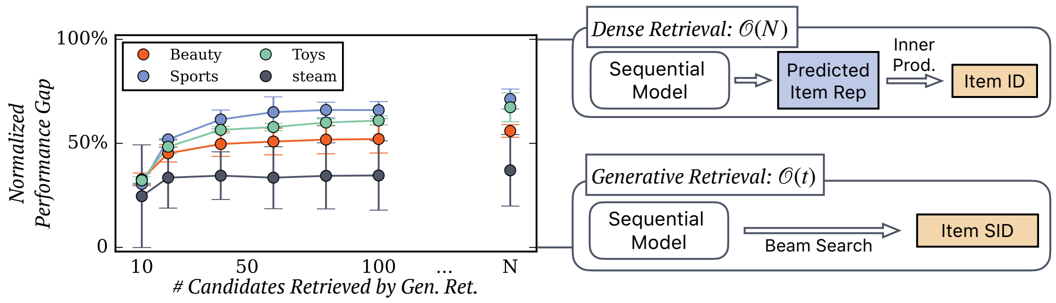

Figure 1: *Normalized Performance Gap Between Generative and Dense Retrieval Across Datasets, and How Our Method Bridges the Gap as the Number of Retrieved Candidates Increases.* The left panel illustrates the normalized performance gap between generative and dense retrieval models across several datasets (Beauty, Sports, Toys, Steam) for the Recall@10 metric. In this normalization, 0% represents the performance of the generative retrieval model, while 100% corresponds to the performance of the dense retrieval model, which is designed to utilizes the same amount of information as the generative approach. The figure highlights how our hybrid method progressively recovers the performance gap as the number of candidates retrieved by the generative model increases. Dense retrieval calculates the inner product between predicted item representations and the entire item set, scaling with $\mathcal{O}(N)$ and requiring storage of $\mathcal{O}(N)$ embeddings. In contrast, generative retrieval stores only $\mathcal{O}(t)$ learnable embeddings and predicts the next item using beam search, scaling with $\mathcal{O}(tK)$, where K is the beam search size and t is the number of Semantic IDs. Our hybrid method combines the strengths of generative retrieval for candidate generation with dense retrieval for ranking, reducing both storage and computational complexity compared to pure dense retrieval. This hybrid approach significantly improves the performance of generative retrieval, narrowing the performance gap with dense retrieval, as demonstrated by the saturating curve in the figure.

quential generative and dense retrieval models under identical conditions, revealing a performance gap between these two approaches. Furthermore, we identify a limitation in the generative retrieval methods' ability to handle cold-start items, indicating an area for improvement.

To address these challenges, we propose a novel hybrid model called LIGER that synergistically combines the strengths of generative and dense retrieval methods. Our SID-based hybrid model leverages the computational and storage efficiencies of generative retrieval, while enhancing its capabilities in generating cold-start items and improving ranking performance through the integration of dense retrieval techniques. By selectively applying dense retrieval to a limited set of candidates generated by an SID-based generative module, we maintain the minimal storage requirements of the learnable embedding in generative retrieval while significantly enhancing its performance.

In Figure 1, we present a key result comparing our method with dense and generative retrieval approaches: the generative retrieval approach we compare to adheres to the setup described in (Rajput et al., 2024), and the sequential dense retrieval method we use for this comparison has been designed to utilize the same amount of information as the generative retrieval method. For clarity of comparison, we compute the performance gap between generative and dense retrieval methods, normalizing it across different datasets to a scale of 0 to 100%. The figure illustrates the extent to which our hybrid approach closes this gap (see Section 3 for detailed discussions). To this end, we provide a comprehensive comparison between the sequential dense and generative retrieval paradigms. Specifically, our key contributions are as follows:

- We identify and analyze two primary limitations of the generative retrieval method: (1) Generative retrieval exhibits a performance gap compared to dense retrieval, given the same amount of information, and (2) it tends to overfit to items encountered during training, resulting in a lower probability of generating cold-start items.

- We propose LIGER (LeveragIng dense retrieval for GEnerative Retrieval), a novel method that synergistically combines the strengths of dense and generative retrieval to significantly enhance the performance of sequential recommender systems. By integrating these methodologies, LIGER effectively reduces the performance gap between dense and generative retrieval while improving the generation of cold-start items.

- We conduct extensive experiments to validate the effectiveness of LIGER. Our results demonstrate that LIGER not only outperforms existing sequential dense and generative retrieval models on standard benchmark datasets but also shows better performance in scenarios involving cold-start items. Additionally, we analyze the computational efficiency and scalability of LIGER, suggesting its potential for large-scale applications.

2 Related Work

Generative Retrieval. The concept of generative retrieval was first proposed by Tay et al. (2022) within the domain of document retrieval. This paradigm shifts from traditional search and retrieval methods by encoding document information directly into the weights of a Transformer model. Subsequent studies (De Cao et al., 2020; Bevilacqua et al., 2022; Feng et al., 2022) have expanded on this foundation, enhancing document retrieval through improvements in indexing (Lee et al., 2022b;c; Wang et al., 2022), and the efficient continual database updates (Mehta et al., 2022; Kishore et al., 2023; Chen et al., 2023).

In the realm of sequential recommendation systems, Rajput et al. (2024) is the first work to leverage the generative retrieval techniques. The target item is directly generated given a user's interaction history, rather than selecting top items by ranking all relevant user-item pairs. A key challenge in generative retrieval is striking a balance between memorization and generalization when encoding items. To address this, semantic IDs have been proposed by leveraging RQ-VAE models (Lee et al., 2022a; Van Den Oord et al., 2017). These models encode content-based embeddings into a compact, discrete semantic indexer that captures the hierarchical structure of concepts within an item's content, proving to be scalable in industrial applications (Singh et al., 2023). Recent developments (Hou et al., 2022a) have expanded semantic-ID-based generative retrieval to include contrastive learning (Jin et al., 2024), multimodal integration (Liu et al., 2024), tokenization techniques (Sun et al., 2024), and learning-to-rank methods (Li et al., 2024).

Sequential Dense Recommendation. Traditional sequential dense recommender models follow the paradigm of learning representations of users, items, and their interactions with multimodal data. Early work (Hidasi et al., 2015) proposed architectures based on traditional Recurrent Neural Networks (RNNs), while later studies (Kang & McAuley, 2018a; Sun et al., 2019; de Souza Pereira Moreira et al., 2021) have shifted towards the Transformer architecture to enhance performance. Besides capturing the user-item interaction history pattern with the sequential modeling, extra features such as item attributes (Zhang et al., 2019; Zhou et al., 2020) has been utilized to further improve the performance. With the recent advancements in Large Language Models (LLMs), several works have explored using these models as the backbone for recommender systems, aligning item representations with LLMs to improve recommendation performance (Li et al., 2023b; Hou et al., 2022c; Cao et al., 2024a; Zheng et al., 2024). In this work, we aim to merge the sequential dense recommendation approach with generative retrieval techniques, assessing performance gaps and computational costs, and proposing a hybrid method that combines the strengths of both paradigms.

Cold-start Problem. Traditional challenges such as long-tail and cold-start items continue to hinder recommendation systems. The long-tail items issue arises from skewed distributions where a few popular items dominate user interactions (Zhang et al., 2022; 2020), while the cold-start problem arises when new items are introduced without any historical interaction data. Recent studies (Hou et al., 2022c; Li et al., 2023b) have shown that textual embeddings can provide a robust prior for tackling the cold-start issue, and further improvements have been achieved by integrating pretrained LLMs (Huang et al., 2024; Sanner et al., 2023) and knowledge graphs (Frej et al., 2024). In this work, we explore the cold-start problem within the context of generative retrieval and propose a hybrid method that combines dense retrieval with textual embeddings to effectively mitigate this issue.

3 Analysis of Generative and Dense Retrieval Paradigms

In this section, we delve into the methodology of the generative retrieval approach (Rajput et al., 2024) (see Section 3.1), as well as the sequential dense retrieval methods such as (Hou et al., 2022b) (see Section 3.2). In Section 3.3, we examine the performance gap between generative retrieval and

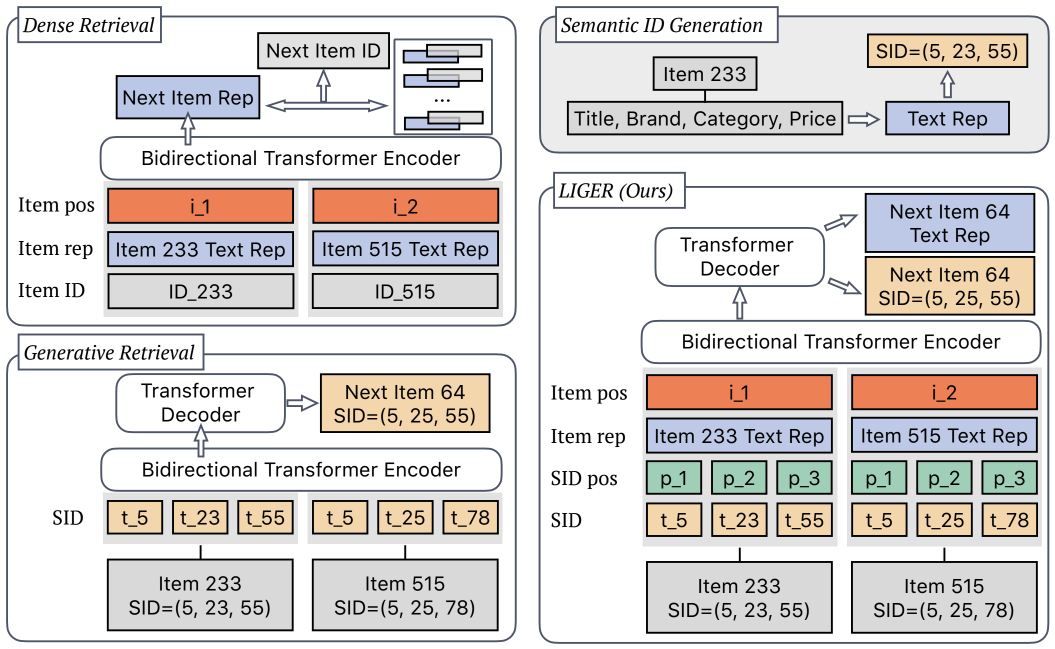

Figure 2: *Overview of Sequential Dense Retrieval, Generative Retrieval, and Our Hybrid Retrieval Method,* LIGER. Dense Retrieval (*upper left*) uses an encoder model to map item IDs and text representations into dense embeddings, which are used to predict the next item in the sequence based on similarity. Generative Retrieval (*lower left*) employs an encoder-decoder Transformer to generate the next item's semantic ID from the given semantic ID trajectory. These semantic IDs are derived from item features such as title, brand, price, and category (*upper right*). Our proposed Hybrid Retrieval, LIGER (*lower right*) combines both semantic ID input and item text representations, integrating dense and generative retrieval techniques. By taking item positions, text representations, and semantic IDs as input, and outputs both the predicted item embedding and the next item's representation.

traditional sequential dense retrieval methods, and then discuss the challenges generative retrieval faces in handling item cold-start scenarios in Section 3.4.

3.1 Generative Retrieval Methodology

The generative retrieval approach such as TIGER (Rajput et al., 2024) typically follows a two-stage training process. The first stage involves collecting textual descriptions for each item based on their attributes. These descriptions serve as inputs to a content model (e.g., a language encoder) that produces item embeddings, subsequently quantized by an RQ-VAE (Lee et al., 2022a) to attribute a semantic ID for each item.

In the second stage of training, the item embeddings and the trained RQ-VAE model are discarded, retaining only the semantic IDs. These IDs replace the original item indices in the item interaction trajectory. The Transformer is then trained on these trajectories to predict the semantic IDs of subsequent items. During inference, a set of candidate items is retrieved using beam search and the trained Transformer, based on their semantic IDs. A visual representation of the generative retrieval method is provided in Figure 2 *(Lower Left)*. Notably, although the textual item embeddings are excluded from the second stage of training, they continue to play a crucial role in generating semantic IDs.

3.2 Textual-Informed Sequential Dense Retrieval Method

Sequential dense retrieval methods typically consist of the following components: (1) learning of item embeddings through sequence modeling, and (2) dot-product search for retrieval. To enhance the learning of item representation, several dense retrieval methods such as Hou et al. (2022b) integrate textual information with sequential interactions to facilitate transferable representation learning.

Building on these insights, we design the dense retrieval to incorporate both textual information and sequential interaction data as follows: The item index is input into the embedding layer, and its representation is enriched by adding the item's textual embedding to each corresponding item index. The sequential model is then trained to output a predicted embedding. For retrieval, we employ an inner product search against the item embedding set, where each item's representation consists of both learnable item embeddings and the added textual embeddings. During training, the predicted item embedding is compared to all other item embeddings, and a cross-entropy loss is applied to optimize the model's accuracy in predicting the correct items. Figure 2 *(Upper Left)* provides a detailed illustration of this dense retrieval model design.

3.3 The Identified Performance Gap

As discussed in Section 3.1 and Section 3.2, both the generative retrieval model and our designed sequential dense retrieval methods utilize item textual embeddings and sequential interaction information. To ensure a fair comparison between the generative retrieval and dense retrieval methods, we maintain consistency in model architecture, data preprocessing, and information utilization. The specific details of our experiment setup are described in Section 5.1.

The result is shown in Figure 1 *(Left)*, where a notable performance gap exists between the generative and dense retrieval methods. In this section, we investigate whether the cause for this discrepency steps from the relative inefficiency of semantic ID embedding representation compared to item ID embedding representation. To dissect this effect, we have conducted ablation studies on the Amazon Beauty dataset using the dense retrieval paradigm, specifically examining this factor. The studies are structured as follows. Recall that a full-fledged dense retrieval method takes both item ID and projected text representation as input, and outputs an embedding that matches the item ID embedding with projected text embedding. We modify this setup in two ways:

- *Input:* Item ID and learnable text representation; *Output*: learnable text representation;
- *Input:* Semantic ID and learnable text representation; *Output*: learnable text representation.

The results, depicted in Figure 3, show that using semantic IDs as input and predicting the item text representation as the target allows the dense retrieval method to recover approximately 75% of the performance gap between the generative and dense retrieval methods with item ID representation. This indicates that the primary contributor to the performance gap is the inefficiency of the next-token prediction loss in generating retrieved items, rather than the semantic ID representation itself.

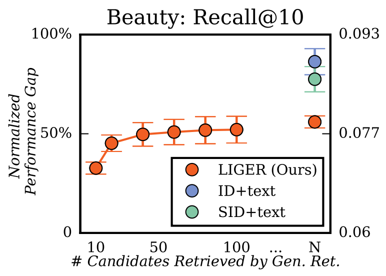

Figure 3: *Comparison of Recall@10 on Amazon Beauty across different ablation methods.* The left y-axis shows the normalized performance gap between generative and dense retrieval, and the right y-axis shows the actual Recall@10. The ablation studies (ID+text and SID+text) recover approximately 75% of the performance gap, highlighting the inherent gap between dense and generative retrieval. Our proposed hybrid method, LIGER, partially bridges this gap. As the number of retrieved candidates increases (x-axis), our method consistently improves generative retrieval, further closing the gap with dense retrieval.

3.4 Challenges in Cold-Start Item Prediction with Generative Retrieval Models

In addition to the performance gap previously identified, our investigation extends to the cold-start item generation problem, a critical issue in the dynamic environment of real-world recommendation systems. As new items are continuously introduced, they often lack sufficient user interactions, which impedes their predictability until a significant amount of interaction data is gathered. For dense retrieval, the inclusion of item textual embeddings provide some prior information, thus partially retaining the ability to retrieve cold-start items.

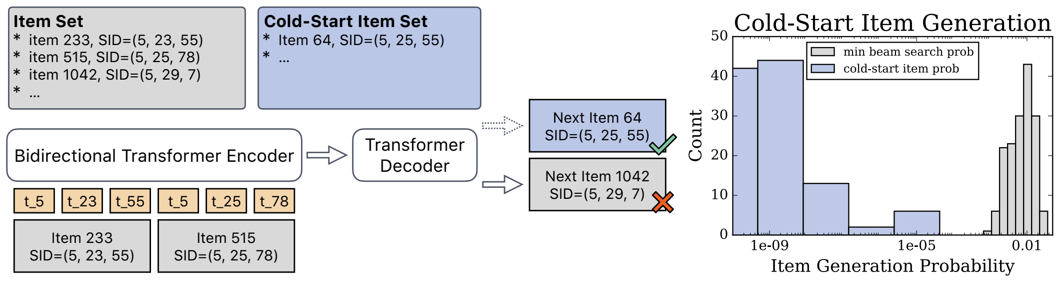

Figure 4: *Generative Retrieval Model Fails to Generate Cold-Start Items.* On the left, an example shows the generative retrieval model failing to predict the cold-start item with ID 64, instead predicting a previously seen item (ID 1042). On the right, we plot the probability of generating cold-start items on the Amazon Beauty dataset, compared to the minimal probability achievable by beam search when retrieving the top 10 items. Beam search predictions exhibit significantly higher probabilities than the ground-truth cold-start items.

Hence, a natural research question arises: *Can generative retrieval models, which also leverage item textual embeddings in their process to generate semantic indices, predict cold-start items?* To explore this, we monitored the cold-start item generation probability for the trained generative model and present the results in Figure 4 (Right). The results reveal that the model's learned conditional probability tends to overfit to items encountered during training, resulting in a significantly diminished ability to generate cold-start items. In fact, the generation probability often falls below the minimal probability achievable by the beam search. This limitation underscores the generative retrieval model's challenges in generalizing to unseen item sets, highlighting a crucial area for further improvement and research.

It is worth noting that Rajput et al. (2024), propose an alternative solution to mitigate the issue of cold-start item generation. Their approach involves setting a predefined threshold ε for cold-start item within the retrieved candidate set of K items, effectively generating $K \cdot \varepsilon$ cold-start items. However, this method relies on prior knowledge of the ratio between recommended cold-start and non-cold-start items, which may not always be available. Moreover, the cold-start item with the minimal generation probability may no longer retain its relevance or intended impact. Therefore, we argue that the challenges in cold-start item generation persist for generative retrieval models, indicating a need for more robust solutions that do not depend heavily on predefined parameters or assumptions.

4 METHODOLOGY

The notion that "there is no free lunch" holds true in the context of retrieval methods. As concluded from the previous section, a performance gap exists between generative retrieval and traditional dense retrieval, and generative retrieval struggles to generate cold-start items. The advanced performance of dense retrieval comes at the expense of increased storage, learning, and inference cost. Conversely, generative retrieval excels in efficiency but lags behind in performance.

The trade-offs between approaches are summarized in Table 1, where N represents the total number of items, t denotes the total number of semantic IDs, and K is the number of candidates to be retrieved during inference.

Table 1: *Comparison of Dense Retrieval, Generative Retrieval, and Our Hybrid Retrieval Methods Across Different Costs.* Here N represents the total number of items, t denotes the total number of semantic IDs used by generative retrieval method, and K indicates the number of candidates retrieved during inference.

	Dense Retrieval	Generative Retrieval	LIGER (Ours)
Learnable Embedding	$\mathcal{O}(N)$	$\mathcal{O}(t)$	$\mathcal{O}(t)$
(Fixed) Item Embedding Stored During Training	$\mathcal{O}(N)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$
Inference Cost	$\mathcal{O}(N)$	$\mathcal{O}(tK)$	$\mathcal{O}(tK)$
Cold-Start Item Generation	Yes	No	Yes

In this section, we propose a hybrid method, called LIGER, that combines the strengths of both approaches. Our goal is to utilize efficient learnable embeddings and reduce inference costs while enabling the generative retrieval model to generate cold-start items and bridge the gap with dense retrieval. To achieve this, we integrate textual item embeddings into the sequential model training phase of the generative retrieval method. Although this increases the storage cost for item embeddings during training, it significantly improves the performance of generative retrieval and enables cold-start item generation. The associated costs of LIGER are detailed in the last column of Table 1.

LIGER builds upon the semantic-ID-based input format and the beam search decoding used in generative retrieval. Following the design choice in dense retrieval, we augment the input of the generative retrieval model with the textual item embeddings. Additionally, we modify the model to output predicted embeddings for each item, enriching the information processed during retrieval. Figure 2 (*Lower Right*) provides a detailed illustration of LIGER, showcasing the integration and workflow of the enhanced model. During inference, the semantic ID prediction head retrieves K items, which are then supplemented with natural cold-start items and ranked using the output embedding head.

The efficacy of LIGER is demonstrated in Figure 1 (*left*), where it consistently improves upon the generative retrieval method across datasets, significantly narrowing the performance gap with dense retrieval. Furthermore, as shown in Figure 3 on the Beauty dataset, our method approaches the performance upper bound set by the dense retrieval method with semantic ID as input. We hypothesize that the remaining performance gap may be attributed to the weight sharing mechanism for multi-objective optimization in our model (Lakkapragada et al., 2023; Yu et al., 2020; Javaloy & Valera, 2021). In the next section, we will demonstrate the effectiveness of our method across various datasets and baseline methods.

5 Experimental Setup and Results

In this section, we present the experimental results across various datasets and baseline methods, showcasing the performance on both in-set and cold-start items. Specifically, we assess the cold-start performance by testing on items that are naturally unseen during training, which is determined by the dataset statistics.

5.1 Experimental Setup

Datasets. We evaluate LIGER on the following 4 datasets. We preprocess the datasets using the standard 5-core filtering method (Zhang et al., 2019; Zhou et al., 2020) which removes items with fewer than 5 users and users with fewer than 5 interactions. We also truncate sequences to a maximum length of 20. The resulting dataset statistics are detailed in Table 2.

• *Amazon Beauty, Sports, and Toys* (He & McAuley, 2016): We use the Amazon Review dataset (2014), focusing on three categories: Beauty, Sports and Outdoors, and Toys and Games. For each item, we construct embeddings by incorporating four key attributes: title, price, category, and description.

• *Steam* (Kang & McAuley, 2018b): The dataset comprises online reviews of video games, from which we extract relevant attributes to construct item embeddings. Specifically, we utilize the following attributes: title, genre, specs, tags, price, and publisher. To reduce the dataset size and make it more manageable, we apply subsampling by selecting every 7th sequence, thereby retaining a representative subset of the data.

When generating the item textual embedding, the item attributes are processed using the sentence-T5 model Ni et al. (2021).

Semantic ID Generation. Utilizing the textual embeddings generated from the sentence-T5 model, we employ a 3-layer MLP for both the encoder and decoder in the RQ-VAE Lee et al. (2022a). The RQ-VAE features three levels of learnable codebooks, each with a dimension of 128 and a cardinality of 256. We use the AdamW optimizer to train the RQ-VAE, setting the learning rate at 0.001 and the weight decay at 0.1. To prevent collisions (i.e., the same semantic ID representing different items), following Rajput et al. (2024) we append an extra token at the end of the ordered semantic codes to ensure uniqueness.

Table 2: Dataset statistics after applying 5-core filtering to both users and items. The first three datasets (Beauty, Sports, and Toys) are subsets of the Amazon review dataset.

Dataset	# users	# items	# actions	# cold-start items
Beauty	22,363	12,101	198,502	43
Toys and Games	19,412	11,924	167,597	56
Sports and Outdoors	35,598	18,357	296,337	81
Steam	47,761	12,012	599,620	400

Sequential Modeling Architecture. For the generative model, we utilize the T5 (Raffel et al., 2020) encoder-decoder model, configuring both the encoder and decoder with 6 layers, an embedding dimension of 128, 6 heads, and a feed-forward network hidden dimension of 1024. The dense retrieval model designed in Section 3.2 employs only the T5-encoder with 6 layers, while maintaining the same hyper-parameters. We use the AdamW optimizer with a learning rate of 0.0003, a weight decay parameter of 0.035, and a cosine learning rate scheduler.

Evaluation Metrics. We assess the model's performance using Normalized Discounted Cumulative Gain (NDCG)@10 and Recall@10. For dataset splitting, we adopt the leave-one-out strategy following (Kang & McAuley, 2018b; Zhou et al., 2020; Rajput et al., 2024), designating the last item as the test label, the preceding item for validation, and the remainder for training. During training, early stopping is applied based on the in-set NDCG@10 validation metric. For our method, we assess the validation performance of the two components: (A) the semantic ID prediction head and (B) the output embedding head. To ensure a balanced evaluation, we implement early stopping based on the sum of in-set NDCG@10 for component (A) and in-set Recall@10 for component (B). To ensure fair evaluation of cold-start items, we exclude them from semantic ID generation to prevent data contamination.

Baselines. We compare our methods with five state-of-the-art Item-ID-based dense retrieval methods, including:

1. SASRec (Kang & McAuley, 2018b). A self-attention based sequential recommendation model that learns to predict the next item ID based on the user's interaction history.
2. FDSA (Zhang et al., 2019) [*feature-informed*]. This method extends SASRec by incorporating item features into the self-attention model, allowing it to leverage prior information about cold-start items through their attributes.
3. S^3-Rec (Zhou et al., 2020) [*feature-informed*]. A self-attention based model that utilizes data correlation to create self-supervision signals, improving sequential recommendation through pre-training.
4. UnisRec (Hou et al., 2022b) [*modality-based*]. A model that learns universal item representations by utilizing associated description text and a lightweight encoding architecture that incorporates parametric whitening and a mixture-of-experts adaptor. We fine-tune the released pretrained model in the transductive setting.
5. Recformer (Li et al., 2023a) [*modality-based*]. A bidirectional Transformer-based model that encodes item information using key-value attributes described by text. We fine-tune the pre-trained model on the downstream datasets.

We also compare LIGER against TIGER (Rajput et al., 2024), a semantic ID-based generative retrieval method. Although subsequent works have built upon this paradigm using large language models (LLMs) (Zheng et al., 2023; Cao et al., 2024b), they rely on pre-trained LLMs, which are outside the scope of our comparisons.

Experimental Results. The results from the benchmark dataset are presented in Table 3, where the mean and standard deviation are calculated across three random seed runs. Traditional item-ID-based methods, such as SASRec exhibit poor in-set performance compared to semantic-ID-based models. However, when attribute information is included, models like FDSA and S^3-Rec show improved in-set performance. Nevertheless, their performance on cold-start items remains subpar due to the static nature of item embeddings. In contrast, models that utilize text representations

Table 3: *Performance Comparison Across Baseline Methods on Amazon Beauty, Sports, Toys, and Steam Datasets.* The best performance is highlighted in bold, and the second-best performance is underlined. Our method consistently achieves either the best or second-best performance across all datasets, closely followed by modality-based baselines (UniSRec or RecFormer). We report LIGER's results where generative retrieval is used to retrieve 20 items, followed by the sequential dense retrieval.

	Methods	Inference Cost	**NDCG@10↑**		**Recall@10↑**	
			In-set	Cold	In-set	Cold
Beauty	SASRec	$\mathcal{O}(N)$	0.02179 ± 0.00023	0.0 ± 0.0	0.05109 ± 0.00042	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.02244 ± 0.00135	0.0 ± 0.0	0.04530 ± 0.00357	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.02279 ± 0.00058	0.0 ± 0.0	0.05226 ± 0.00229	0.0 ± 0.0
	UniSRec	$\mathcal{O}(N)$	$\underline{0.03346 \pm 0.00057}$	0.01422 ± 0.00128	$\underline{0.06937 \pm 0.00110}$	0.03704 ± 0.00000
	RecFormer	$\mathcal{O}(N)$	0.02880 ± 0.00085	$\underline{0.01955 \pm 0.00433}$	0.06265 ± 0.00196	$\underline{0.04733 \pm 0.00943}$
	TIGER	$\mathcal{O}(tK)$	0.03216 ± 0.00084	0.0 ± 0.0	0.06009 ± 0.00204	0.0 ± 0.0
	LIGER (Ours)	$\mathcal{O}(tK)$	$\mathbf{0.03879} \pm \mathbf{0.00070}$	$\mathbf{0.02483} \pm \mathbf{0.00131}$	$\mathbf{0.07500} \pm \mathbf{0.00137}$	$\mathbf{0.07407} \pm \mathbf{0.00617}$
Sports	SASRec	$\mathcal{O}(N)$	0.01160 ± 0.00038	0.0 ± 0.0	0.02696 ± 0.00102	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.01391 ± 0.00162	0.0 ± 0.0	0.02699 ± 0.00312	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.01097 ± 0.00033	0.0 ± 0.0	0.02557 ± 0.00034	0.0 ± 0.0
	UniSRec	$\mathcal{O}(N)$	0.01814 ± 0.00041	0.00676 ± 0.00244	0.03753 ± 0.00106	0.01559 ± 0.00447
	RecFormer	$\mathcal{O}(N)$	0.01318 ± 0.00053	$\mathbf{0.01797} \pm \mathbf{0.00000}$	0.02921 ± 0.00167	$\mathbf{0.03801} \pm \mathbf{0.00000}$
	TIGER	$\mathcal{O}(tK)$	$\underline{0.01989 \pm 0.00085}$	0.00064 ± 0.00056	0.03822 ± 0.00109	0.00195 ± 0.00169
	LIGER (Ours)	$\mathcal{O}(tK)$	$\mathbf{0.02437} \pm \mathbf{0.00070}$	$\underline{0.01254 \pm 0.00256}$	$\mathbf{0.04642} \pm \mathbf{0.00127}$	$\underline{0.03314 \pm 0.00338}$
Toys	SASRec	$\mathcal{O}(N)$	0.02756 ± 0.00079	0.0 ± 0.0	0.06314 ± 0.00178	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.02375 ± 0.00277	0.0 ± 0.0	0.04684 ± 0.00483	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.02942 ± 0.00071	0.0 ± 0.0	0.06659 ± 0.00135	0.0 ± 0.0
	UniSRec	$\mathcal{O}(N)$	0.03622 ± 0.00056	0.01090 ± 0.00084	0.07472 ± 0.00058	0.02477 ± 0.00195
	RecFormer	$\mathcal{O}(N)$	$\underline{0.03697 \pm 0.00052}$	$\mathbf{0.04432} \pm \mathbf{0.00094}$	$\mathbf{0.07971} \pm \mathbf{0.00170}$	$\mathbf{0.10023} \pm \mathbf{0.00516}$
	TIGER	$\mathcal{O}(tK)$	0.02949 ± 0.00049	0.0 ± 0.0	0.05782 ± 0.00163	0.0 ± 0.0
	LIGER (Ours)	$\mathcal{O}(tK)$	$\mathbf{0.03864} \pm \mathbf{0.00042}$	$\underline{0.03527 \pm 0.00518}$	$\underline{0.07591 \pm 0.00132}$	$\underline{0.09347 \pm 0.01086}$
Steam	SASRec	$\mathcal{O}(N)$	0.14763 ± 0.00051	0.0 ± 0.0	0.18259 ± 0.00055	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.08236 ± 0.00152	0.0 ± 0.0	0.14773 ± 0.00234	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.14437 ± 0.00127	0.0 ± 0.0	0.18025 ± 0.00222	0.0 ± 0.0
	UniSRec	$\mathcal{O}(N)$	-	-	-	-
	RecFormer	$\mathcal{O}(N)$	-	-	-	-
	TIGER	$\mathcal{O}(tK)$	$\mathbf{0.15034} \pm \mathbf{0.00064}$	0.0 ± 0.0	$\underline{0.18980 \pm 0.00135}$	0.0 ± 0.0
	LIGER (Ours)	$\mathcal{O}(tK)$	$\underline{0.14876 \pm 0.00077}$	$\mathbf{0.00377} \pm \mathbf{0.00100}$	$\mathbf{0.19281} \pm \mathbf{0.00149}$	$\mathbf{0.01083} \pm \mathbf{0.00292}$

and pre-training, such as UniSRec and RecFormer, demonstrate enhanced capabilities in handling cold-start item scenarios. The inclusion of text embeddings pre-training enables these models to better handle unseen items. TIGER, which is a semantic-ID-based generative retrieval model, outperforms item-ID-based methods in terms of in-set performance but still struggles with cold-start item generation.

Our model, LIGER, builds upon TIGER by using semantic-ID-based inputs and combining dense retrieval with semantic ID generation as outputs. This approach significantly improves upon the TIGER method and enables effective generation of cold-start items. Across all datasets, our method consistently achieves either the best or second-best performance, closely followed by modality-based baselines such as UniSRec and RecFormer. We adopt a hybrid approach for our reporting, where we use generative retrieval to retrieve 20 items and then rank them using dense retrieval, including cold-start items. Comprehensive result including performance of LIGER with different number of retrieved items from generative retrieval is presented in Table 4.

6 DISCUSSION

Addressing Cold-Start Items with Hybrid Retrieval Models. The generative retrieval method's struggle with cold-start items primarily stems from overfitting to familiar semantic IDs during training, as discussed in Section 3.4. To mitigate this issue, LIGER efficiently combines dense retrieval with generative retrieval. Specifically, LIGER first generates a small set of K candidates (where $K \ll N$) using generative retrieval, and then supplements these candidates with a set of cold-start items, similar to the approach used in dense retrieval methods. This significantly reduces the candidate set for LIGER, thanks to its generative retrieval module. By integrating dense retrieval, we ensure that even if the generative retrieval retrieves fewer than N items, the model still achieves a baseline level of performance for cold-start scenarios, comparable to considering all item sets. As shown in Table 4, the dense retrieval component guarantees a minimum level of cold-start performance by leveraging item text embeddings as prior information.

Comparative Performance with Current Dense Retrieval Methods. While LIGER outperforms existing baselines, its primary objective is to strike a balance between the two frameworks under discussion. As highlighted in previous sections, there are notable performance discrepancies between generative and dense retrieval methods, despite using the same input information and model architecture. Our primary goal is to bridge the gap between these two distinct recommendation paradigms, enhancing a deeper understanding of their respective strengths, weaknesses, and associated costs. The results presented should illuminate potential future directions for integrating these paradigms more effectively. By exploring the synergy between generative and dense retrieval, we hope to inspire future directions for integrating these paradigms more effectively, leading to development of more robust recommendation systems.

7 Conclusion

In this work, we conducted a comprehensive comparison between traditional dense retrieval methods and the emerging generative retrieval approach. Our analysis revealed the limitations of dense retrieval, including high computational and storage requirements, while highlighting the advantages of generative retrieval, which uses semantic IDs and generative models to enhance efficiency and semantic understanding. Furthermore, we have identified the challenges faced by generative retrieval, particularly in handling cold-start items and matching the performance of dense retrieval. To address these challenges, we introduced a novel hybrid model, LIGER, that combines the strengths of both approaches. Our findings demonstrate that our hybrid model surpasses existing models in handling cold-start scenarios and achieves advanced overall performance on benchmark datasets. Furthermore, it offers scalability and computational efficiency, making it suitable for large-scale applications.

Looking ahead, the fusion of dense and generative retrieval methods holds tremendous potential for advancing recommendation systems. Our research provides a foundation for further exploration into hybrid models that capitalize on the strengths of both retrieval types. As these models continue to evolve, they will become increasingly practical for real-world applications, enabling more personalized and responsive user experiences.

References

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. Autoregressive search engines: Generating substrings as document identifiers. *ArXiv*, abs/2204.10628, 2022. URL `https://api.semanticscholar.org/CorpusID:248366293`.

Yuwei Cao, Nikhil Mehta, Xinyang Yi, Raghunandan Keshavan, Lukasz Heldt, Lichan Hong, Ed H Chi, and Maheswaran Sathiamoorthy. Aligning large language models with recommendation knowledge. *arXiv preprint arXiv:2404.00245*, 2024a.

Yuwei Cao, Nikhil Mehta, Xinyang Yi, Raghunandan H. Keshavan, Lukasz Heldt, Lichan Hong, Ed H. Chi, and Maheswaran Sathiamoorthy. Aligning large language models with recommendation knowledge. *ArXiv*, abs/2404.00245, 2024b. URL `https://api.semanticscholar.org/CorpusID:268819967`.

Jiangui Chen, Ruqing Zhang, J. Guo, M. de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. Continual learning for generative retrieval over dynamic corpora. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023. URL `https://api.semanticscholar.org/CorpusID:261277063`.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM conference on recommender systems*, pp. 143–153, 2021.

Chao Feng, Wu Li, Defu Lian, Zheng Liu, and Enhong Chen. Recommender forest for efficient retrieval. In *Neural Information Processing Systems*, 2022. URL `https://api.semanticscholar.org/CorpusID:258509354`.

Jibril Frej, Marta Knezevic, and Tanja Käser. Graph reasoning for explainable cold start recommendation. *ArXiv*, abs/2406.07420, 2024. URL `https://api.semanticscholar.org/CorpusID:270379698`.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th International Conference on World Wide Web*, 2016. URL `https://api.semanticscholar.org/CorpusID:1964279`.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL `https://api.semanticscholar.org/CorpusID:1915014`.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. *Proceedings of the ACM Web Conference 2023*, 2022a. URL `https://api.semanticscholar.org/CorpusID:253098091`.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji rong Wen. Towards universal sequence representation learning for recommender systems. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022b. URL `https://api.semanticscholar.org/CorpusID:249625869`.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 585–593, 2022c.

Feiran Huang, Zhen Yang, Junyi Jiang, Yuan-Qi Bei, Yijie Zhang, and Hao Chen. Large language model interaction simulator for cold-start item recommendation. *ArXiv*, abs/2402.09176, 2024. URL `https://api.semanticscholar.org/CorpusID:267657482`.

Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. *arXiv preprint arXiv:2103.02631*, 2021.

Mengqun Jin, Zexuan Qiu, Jieming Zhu, Zhenhua Dong, and Xiu Li. Contrastive quantization based semantic code for generative recommendation. *arXiv preprint arXiv:2404.14774*, 2024.

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

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pp. 197–206. IEEE Computer Society, 2018b. doi: 10.1109/ICDM.2018.00035.

Varsha Kishore, Chao gang Wan, Justin Lovelace, Yoav Artzi, and Kilian Q. Weinberger. Incdsi: Incrementally updatable document retrieval. *ArXiv*, abs/2307.10323, 2023. URL `https://api.semanticscholar.org/CorpusID:259991824`.

Anish Lakkapragada, Essam Sleiman, Saimourya Surabhi, and Dennis P Wall. Mitigating negative transfer in multi-task learning with exponential moving average loss weighting strategies (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 16246–16247, 2023.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11523–11532, 2022a.

Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. Contextualized generative retrieval. *ArXiv*, abs/2210.02068, 2022b. URL https://api.semanticscholar.org/CorpusID:252715634.

Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. Nonparametric decoding for generative retrieval. In *Annual Meeting of the Association for Computational Linguistics*, 2022c. URL https://api.semanticscholar.org/CorpusID:258959550.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023a. URL https://api.semanticscholar.org/CorpusID:258841284.

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

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8716–8723, 2024.

Han Liu, Yin wei Wei, Xuemeng Song, Weili Guan, Yuan-Fang Li, and Liqiang Nie. Mmgrec: Multimodal generative recommendation with transformer model. *ArXiv*, abs/2404.16555, 2024. URL https://api.semanticscholar.org/CorpusID:269362930.

Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. Dsi++: Updating transformer memory with new documents. *ArXiv*, abs/2212.09744, 2022. URL https://api.semanticscholar.org/CorpusID:254854290.

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

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

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36, 2024.

Scott Sanner, Krisztian Balog, Filip Radlinski, Benjamin D. Wedin, and Lucas Dixon. Large language models are competitive near cold-start recommenders for language- and item-based preferences. *Proceedings of the 17th ACM Conference on Recommender Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:260164942.

Anima Singh, Trung Vu, Nikhil Mehta, Raghunandan Keshavan, Maheswaran Sathiamoorthy, Yilin Zheng, Lichan Hong, Lukasz Heldt, Li Wei, Devansh Tandon, et al. Better generalization with semantic ids: A case study in ranking for recommendations. *arXiv preprint arXiv:2306.08121*, 2023.

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

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. Learning to tokenize for generative retrieval. *Advances in Neural Information Processing Systems*, 36, 2024.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index. *ArXiv*, abs/2202.06991, 2022. URL `https://api.semanticscholar.org/CorpusID:246863488`.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL `https://api.semanticscholar.org/CorpusID:13756489`.

Yujing Wang, Ying Hou, Hong Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. A neural corpus indexer for document retrieval. *ArXiv*, abs/2206.02743, 2022. URL `https://api.semanticscholar.org/CorpusID:249395549`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

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

Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H. Chi. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. *Proceedings of the Web Conference 2021*, 2020. URL `https://api.semanticscholar.org/CorpusID:226221746`.

Yin Zhang, Ruoxi Wang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, James Caverlee, and Ed H. Chi. Empowering long-tail item recommendation through cross decoupling network (cdn). *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. URL `https://api.semanticscholar.org/CorpusID:253117145`.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448, 2023. URL `https://api.semanticscholar.org/CorpusID:265213194`.

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

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1893–1902, 2020.

A Appendix

In Table 4, we present the full results on the benchmark, where our method with different number of retrieved candidates from generative retrieval are shown.

Table 4: *Performance Table for Amazon Beauty, Sports, Toys, and Steam Datasets Across Various Baseline Methods.* In this table, we present our method with different number of retrieved candidates from generative retrieval.

Datasets	Methods	Inference Cost	NDCG@10↑		Recall@10↑	
			In-set	Cold	In-set	Cold
	SASRec	$\mathcal{O}(N)$	0.02179 ± 0.00023	0.0 ± 0.0	0.05109 ± 0.00042	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.02244 ± 0.00135	0.0 ± 0.0	0.04530 ± 0.00357	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.02279 ± 0.00058	0.0 ± 0.0	0.05226 ± 0.00229	0.0 ± 0.0
Beauty	UniSRec	$\mathcal{O}(N)$	0.03346 ± 0.00057	0.01422 ± 0.00128	0.06937 ± 0.00110	0.03704 ± 0.00000
	RecFormer	$\mathcal{O}(N)$	0.02880 ± 0.00085	0.01955 ± 0.00433	0.06265 ± 0.00196	0.04733 ± 0.00943
	TIGER	$\mathcal{O}(tK)$	0.03216 ± 0.00084	0.0 ± 0.0	0.06009 ± 0.00204	0.0 ± 0.0
	Ours ($K = 20$)	$\mathcal{O}(tK)$	0.03879 ± 0.00070	0.02483 ± 0.00131	0.07500 ± 0.00137	0.07407 ± 0.00617
	Ours ($K = 40$)	$\mathcal{O}(tK)$	0.03938 ± 0.00093	0.01730 ± 0.00295	0.07647 ± 0.00196	0.05144 ± 0.01285
	Ours ($K = 60$)	$\mathcal{O}(tK)$	0.03945 ± 0.00099	0.01689 ± 0.00320	0.07686 ± 0.00210	0.05144 ± 0.01285
	Ours ($K = 80$)	$\mathcal{O}(tK)$	0.03958 ± 0.00105	0.01619 ± 0.00249	0.07717 ± 0.00225	0.04938 ± 0.01069
	Ours ($K = 100$)	$\mathcal{O}(tK)$	0.03960 ± 0.00098	0.01551 ± 0.00345	0.07726 ± 0.00222	0.04733 ± 0.01426
	Ours ($K = N$)	$\mathcal{O}(N)$	0.04003 ± 0.00057	0.01165 ± 0.00277	0.07854 ± 0.00100	0.03498 ± 0.01285
	SASRec	$\mathcal{O}(N)$	0.01160 ± 0.00038	0.0 ± 0.0	0.02696 ± 0.00102	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.01391 ± 0.00162	0.0 ± 0.0	0.02699 ± 0.00312	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.01097 ± 0.00033	0.0 ± 0.0	0.02557 ± 0.00034	0.0 ± 0.0
Sports	UniSRec	$\mathcal{O}(N)$	0.01814 ± 0.00041	0.00676 ± 0.00244	0.03753 ± 0.00106	0.01559 ± 0.00447
	RecFormer	$\mathcal{O}(N)$	0.01318 ± 0.00053	0.01797 ± 0.00000	0.02921 ± 0.00167	0.03801 ± 0.00000
	TIGER	$\mathcal{O}(tK)$	0.01989 ± 0.00085	0.00064 ± 0.00056	0.03822 ± 0.00109	0.00195 ± 0.00169
	Ours ($K = 20$)	$\mathcal{O}(tK)$	0.02437 ± 0.00070	0.01254 ± 0.00256	0.04642 ± 0.00127	0.03314 ± 0.00338
	Ours ($K = 40$)	$\mathcal{O}(tK)$	0.02485 ± 0.00081	0.00719 ± 0.00314	0.04791 ± 0.00155	0.01852 ± 0.00675
	Ours ($K = 60$)	$\mathcal{O}(tK)$	0.02488 ± 0.00080	0.00580 ± 0.00235	0.04809 ± 0.00141	0.01462 ± 0.00506
	Ours ($K = 80$)	$\mathcal{O}(tK)$	0.02496 ± 0.00087	0.00564 ± 0.00210	0.04818 ± 0.00177	0.01462 ± 0.00506
	Ours ($K = 100$)	$\mathcal{O}(tK)$	0.02498 ± 0.00094	0.00515 ± 0.00217	0.04828 ± 0.00193	0.01365 ± 0.00609
	Ours ($K = N$)	$\mathcal{O}(N)$	0.02529 ± 0.00107	0.00350 ± 0.00183	0.04929 ± 0.00224	0.00975 ± 0.00447
	SASRec	$\mathcal{O}(N)$	0.02756 ± 0.00079	0.0 ± 0.0	0.06314 ± 0.00178	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.02375 ± 0.00277	0.0 ± 0.0	0.04684 ± 0.00483	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.02942 ± 0.00071	0.0 ± 0.0	0.06659 ± 0.00135	0.0 ± 0.0
Toys	UniSRec	$\mathcal{O}(N)$	0.03622 ± 0.00056	0.01090 ± 0.00084	0.07472 ± 0.00058	0.02477 ± 0.00195
	RecFormer	$\mathcal{O}(N)$	0.03697 ± 0.00052	0.04432 ± 0.00094	0.07971 ± 0.00170	0.10023 ± 0.00516
	TIGER	$\mathcal{O}(tK)$	0.02949 ± 0.00049	0.0 ± 0.0	0.05782 ± 0.00163	0.0 ± 0.0
	Ours ($K = 20$)	$\mathcal{O}(tK)$	0.03864 ± 0.00042	0.03527 ± 0.00518	0.07591 ± 0.00132	0.09347 ± 0.01086
	Ours ($K = 40$)	$\mathcal{O}(tK)$	0.03967 ± 0.00041	0.02778 ± 0.00628	0.07894 ± 0.00058	0.07207 ± 0.01407
	Ours ($K = 60$)	$\mathcal{O}(tK)$	0.03978 ± 0.00057	0.02463 ± 0.00381	0.07944 ± 0.00066	0.06419 ± 0.01014
	Ours ($K = 80$)	$\mathcal{O}(tK)$	0.03998 ± 0.00060	0.02369 ± 0.00454	0.08026 ± 0.00063	0.06194 ± 0.01365
	Ours ($K = 100$)	$\mathcal{O}(tK)$	0.04010 ± 0.00068	0.02323 ± 0.00403	0.08061 ± 0.00081	0.06081 ± 0.01218
	Ours ($K = N$)	$\mathcal{O}(tK)$	0.04089 ± 0.00136	0.01972 ± 0.00352	0.08300 ± 0.00260	0.05180 ± 0.00975
	SASRec	$\mathcal{O}(N)$	0.14763 ± 0.00051	0.0 ± 0.0	0.18259 ± 0.00055	0.0 ± 0.0
	FDSA	$\mathcal{O}(N)$	0.08236 ± 0.00152	0.0 ± 0.0	0.14773 ± 0.00234	0.0 ± 0.0
	S^3-Rec	$\mathcal{O}(N)$	0.14437 ± 0.00127	0.0 ± 0.0	0.18025 ± 0.00222	0.0 ± 0.0
Steam	UniSRec	$\mathcal{O}(N)$	-	-	-	-
	RecFormer	$\mathcal{O}(N)$	-	-	-	-
	TIGER	$\mathcal{O}(tK)$	0.15034 ± 0.00064	0.0 ± 0.0	0.18980 ± 0.00135	0.0 ± 0.0
	Ours ($K = 20$)	$\mathcal{O}(tK)$	0.14876 ± 0.00077	0.00377 ± 0.00100	0.19281 ± 0.00149	0.01083 ± 0.00292
	Ours ($K = 40$)	$\mathcal{O}(tK)$	0.14865 ± 0.00063	0.00227 ± 0.00053	0.19236 ± 0.00100	0.00637 ± 0.00110
	Ours ($K = 60$)	$\mathcal{O}(tK)$	0.14865 ± 0.00066	0.00177 ± 0.00101	0.19237 ± 0.00102	0.00510 ± 0.00292
	Ours ($K = 80$)	$\mathcal{O}(tK)$	0.14864 ± 0.00072	0.00156 ± 0.00080	0.19225 ± 0.00107	0.00446 ± 0.00221
	Ours ($K = 100$)	$\mathcal{O}(tK)$	0.14863 ± 0.00069	0.00156 ± 0.00080	0.19218 ± 0.00100	0.00446 ± 0.00221
	Ours ($K = N$)	$\mathcal{O}(tK)$	0.14870 ± 0.00070	0.00089 ± 0.00077	0.19239 ± 0.00111	0.00255 ± 0.00221