École Normale Supérieure Paris-Saclay

Criteo AI Lab

Unifying Recommender Systems and Product Search with Generative Retrieval

A master report submitted for the degree of Master of Science

Mathématiques, Vision, Apprentissage (MVA)

in the

Department of Mathematics

September 2024

Author:
Steven ZHENG

Supervisors: Patrick GALLINARI, Alberto LUMBRERAS, Alain RAKOTOMAMONJY
MVA Referee: Rachel BAWDEN

Abstract

This report presents the outcomes of a six-month internship at Criteo AI Lab, focused on the development and exploration of a unified framework for integrating Recommender Systems (RS) and Product Search using Generative Information Retrieval (GenIR). Traditionally, Information Retrieval (IR) and RS have evolved along parallel paths, with IR systems emphasizing keyword-based retrieval and RS focusing on collaborative filtering and content-based recommendations. However, the rise of deep learning and, more recently, generative models, has opened the possibility of merging these fields to enhance both retrieval and recommendation accuracy.

In this work, we propose a novel approach that leverages the generative capabilities of models like T5 to unify IR and RS within a single framework. By generating document identifiers (DocIDs) and item identifiers directly from user queries, the proposed system simplifies the retrieval pipeline, enabling more personalized and context-aware recommendations. The report details the technical foundations, model structure, and training methodologies employed, as well as the empirical results that demonstrate the potential of GenIR to revolutionize both search and recommendation tasks. The findings suggest that a unified GenIR framework can significantly enhance the efficiency and relevance of retrieval and recommendation systems, paving the way for more sophisticated and user-centric applications.

Acknowledgement

This report concludes my six-month internship at Criteo AI Lab and marks the completion of my studies.

My internship at Criteo AI Lab was a transformative experience, providing me with the opportunity to work on integrating Recommender Systems and Product Search using Generative Information Retrieval. The challenging process of developing and refining models for unified retrieval and recommendation significantly enhanced my technical skills in deep learning, natural language processing, and generative models. Additionally, the collaborative environment at Criteo offered invaluable insights into research in the area of AI.

I would also like to extend my gratitude to Criteo for providing not only the robust infrastructure and access to GPU resources, which were essential for the successful completion of my research and development work, but also for fostering an inspiring and supportive environment that greatly contributed to my growth both professionally and personally.

I am deeply grateful to Patrick Gallinari, Alberto Lumbreras, and Alain Rakotomamonjy for their exceptional mentorship, their generosity in sharing their expertise, and their unwavering dedication to this project.

I would also like to express my sincere thanks to Rachel Bawden for her academic supervision throughout this internship.

I extend my appreciation to my professors at the Mathématiques, Vision, Apprentissage (MVA) master's program at ENS Paris-Saclay for their rigorous training and for instilling high standards in both academic and industrial work.

As I close this chapter, I want to offer my heartfelt thanks to my loved ones. Their unwavering support and belief in me have been among the greatest gifts on this journey.

Remerciements

Ce rapport conclut mon stage de six mois au Criteo AI Lab et marque la fin de mes études.

Mon stage au Criteo AI Lab a été une expérience transformatrice, m'offrant l'opportunité de travailler sur l'intégration des systèmes de recommandation et de la recherche de produits à l'aide de la récupération d'information générative. Le processus exigeant de développement et de perfectionnement des modèles pour unifier la recherche et la recommandation a considérablement amélioré mes compétences techniques en apprentissage profond, en traitement du langage naturel et en modèles génératifs. De plus, l'environnement collaboratif chez Criteo m'a offert des perspectives inestimables en recherche dans le domaine de l'IA.

Je tiens également à exprimer ma gratitude à Criteo pour avoir fourni non seulement une infrastructure solide et un accès aux ressources GPU, qui ont été essentiels à la réussite de mon travail de recherche et de développement, mais aussi pour avoir cultivé un environnement inspirant et favorable qui a grandement contribué à mon développement, tant sur le plan professionnel que personnel.

Je suis profondément reconnaissant à Patrick Gallinari, Alberto Lumbreras et Alain Rakotomamonjy pour leur mentorat exceptionnel, leur générosité dans le partage de leur expertise, et leur dévouement sans faille à ce projet.

Je tiens également à exprimer mes sincères remerciements à Rachel Bawden pour sa supervision académique tout au long de ce stage.

J'exprime ma gratitude à mes professeurs du programme de master Mathématiques, Vision, Apprentissage (MVA) à l'ENS Paris-Saclay pour leur formation rigoureuse et pour avoir instauré des normes élevées dans les travaux académiques et industriels.

Enfin, alors que je tourne cette page, je tiens à offrir mes remerciements les plus sincères à mes proches. Leur soutien indéfectible et leur foi en moi ont été parmi les plus beaux cadeaux tout au long de ce parcours.

Contents

1 Introduction

The fields of Information Retrieval (IR) and Recommender Systems (RS) have significantly evolved, driven by the rapid increase of data and the demand for personalized experiences. Initially, IR systems relied on simple keyword-based methods, leveraging statistical word relationships through models like the Bag-of-Words and Vector Space Model (VSM) (Salton and McGill, 1983). These early approaches were limited in their ability to understand semantic nuances, focusing primarily on matching keywords between queries and documents. Over the past two decades, Term Frequency-Inverse Document Frequency (TF-IDF) (Sammut and Webb, 2010) and BM25 (Robertson and Zaragoza, 2009) algorithms improved retrieval effectiveness by considering the importance of terms in a document relative to their occurrence across the corpus, thus addressing some of the shortcomings of earlier models. However, these methods still struggled with issues like vocabulary mismatch and lacked the ability to capture deeper semantic relationships between words.

The landscape changed with the advances in deep learning. These methods introduced dense vector representations of words, as seen in models like Word2Vec (Mikolov et al., 2013) and BERT (Devlin et al., 2019) which capture deeper semantic meanings and contextual relationships. This shift towards dense retrieval methods improved the accuracy and relevance of search results, paving the way for more sophisticated IR systems.

Parallel to these advancements, Recommender Systems have also evolved, incorporating both collaborative filtering and content-based techniques, as well as hybrid approaches that combine the strengths of both. These methods have been developed concurrently, with the choice of approach often depending on the specific context or application. In recent years, deep learning methods have enabled more sophisticated models capable of predicting user preferences and suggesting items by leveraging complex patterns in historical data, user interactions, and item similarities.

Deep learning has similarly revolutionized Information Retrieval systems, where two primary approaches have emerged: sparse retrieval and dense retrieval. Sparse retrieval methods, like those seen in traditional keyword-based systems, rely on representing documents and queries using sparse vectors, where a small subset of terms or features (e.g., words or tokens) are matched directly. An example of deep learning applied to sparse retrieval is the Deep Semantic Matching model (Lu and Li, 2013).

On the other hand, dense retrieval methods (Burges et al., 2005; Karpukhin et al., 2020; Xiong et al., 2020; Khattab and Zaharia, 2020) involve embedding both queries and documents into dense vectors within a continuous vector space, allowing for efficient and highly relevant search results through vector similarity. These techniques allow IR systems to better understand user queries and documents at a deeper level, enabling more accurate retrieval in a wide range of contexts.

Generative Information Retrieval (GenIR) is a novel paradigm in information retrieval that reimagines the retrieval process by directly generating relevant document identifiers (docids) for a given user query. Unlike traditional retrieval methods that rely on separate stages for indexing, retrieval, and ranking, GenIR relies on generative models—often based on sequence-to-sequence (seq2seq) architectures—to autoregressively produce document identifiers as output. This enables a more unified and end-to-end approach where the entire retrieval process, including indexing and query matching, can be optimized within a single model. The Differentiable Search Index (DSI) introduced by Tay et al. (2022) is the first model to implement this approach, marking a significant shift in how information retrieval is conceptualized and implemented. By generating document identifiers directly from the model's parameters, DSI eliminates the need for traditional query processing and document reranking, simplifying the retrieval pipeline and enhancing both efficiency and adaptability.

GenIR has also demonstrated significant potential in advancing Generative Recommender Systems, as highlighted by several promising works (Rajput et al., 2023; Zheng et al., 2024; Wang et al., 2024b; Si et al., 2024). These advancements demonstrate the potential of GenIR to revolutionize how recommendations

are generated by directly leveraging generative models.

While GenIR has already begun to influence both search and recommendation systems independently, the convergence of these fields through GenIR is still in its early stages. To our knowledge, no existing work has yet tackled the integration of both tasks through the prism of GenIR in a unified framework. However, there is Jin et al. (2023), which fine-tunes a model on both downstream tasks separately, illustrating the potential for cross-domain applications. Nevertheless, a fully integrated approach remains unexplored. This integration holds the promise of not only enhancing the relevance and accuracy of both search and recommendation results but also enabling more complex and sophisticated tasks, such as personalized content generation and nuanced natural language understanding. Given the inherent synergy between information retrieval and recommendation systems, a unified approach that make use of GenIR appears both logical and inevitable. Such an approach could seamlessly blend the strengths of IR and RS, leading to more powerful, adaptable, and personalized user experiences.

In this report, we aim to explore and advance this integration by examining the potential of a unified GenIR framework. We will begin with an overview of traditional paradigms in both fields, followed by a discussion of modern approaches that adopt generative models. Our exploration will delve into the technical foundations, implementation details, and empirical results of our proposed approach, ultimately highlighting how GenIR can unify and enhance both information retrieval and recommendation systems.

2 Traditional Paradigms

2.1 Deep Learning based Search

The initial wave of deep learning methods for Information Retrieval (IR) focused on neural network architectures such as Convolutional Neural Networks (CNNs) (Kim, 2014) and Recurrent Neural Networks (RNNs) (Lai et al., 2015) and especially Long Short-Term Memory (LSTM) (Mueller and Thyagarajan, 2016) networks have been successful in representing sentences as fixed-length feature vectors. Other approaches focusing on sparse retrieval, such as DocT5Query (Cheriton, 2019), further improve classic sparse retrieval methods by expanding documents with predicted queries. This technique helps bridge the vocabulary gap between user queries and document content, effectively increasing the relevance of search results in keyword-based systems. These methods laid the groundwork for more advanced IR systems by introducing neural approaches to handle the complexities of natural language.

More recently, the focus in deep learning for IR has shifted towards transformer-based models (Vaswani et al., 2017), most notably BERT (Devlin et al., 2019). Transformers represent a significant leap forward in the ability of IR systems to understand and process natural language. Unlike earlier architectures, which had access to the entire context but processed it less directly, transformer models use self-attention mechanisms to more effectively capture deeper semantic relationships by considering all words in a sentence simultaneously. This shift towards transformers has enabled more sophisticated and accurate document retrieval and ranking, setting new benchmarks across various IR tasks. The adoption of transformers reflects the ongoing evolution in the field, where more context-aware and efficient models are becoming the standard for cutting-edge IR applications.

Modern IR systems typically employ a two-stage process known as "retrieve-then-rank" to efficiently handle large-scale document collections. In the first stage, known as retrieval, a broad set of potentially relevant documents is quickly identified based on a user query. This initial retrieval often relies on traditional methods like term matching or basic neural network models to ensure efficiency.

In the second stage, known as ranking, these documents are re-evaluated and re-ordered using more sophisticated models to ensure that the most relevant documents are presented at the top of the list. In dense retrieval, both the query and the documents are transformed into dense vector representations using neural network-based functions. Specifically, the query is mapped to a vector $\mathbf{q}$ using a function $\phi(\cdot)$, and each document is mapped to a vector $\mathbf{d}$ using a function $\psi(\cdot)$:

$$\mathbf{q} = \phi(\text{Query}), \quad \mathbf{d} = \psi(\text{Document}) \tag{1}$$

These functions, $\phi(\cdot)$ and $\psi(\cdot)$, represent the encoding process and can be implemented in various ways depending on the retrieval method used:

- **Dual Encoder Method:** In this approach, $\phi(\cdot)$ and $\psi(\cdot)$ are typically implemented as separate neural network encoders that independently encode the query and the document into dense vectors. These encoders can be configured in different ways depending on the task:
  - **Siamese Dual Encoder:** In this setup, both $\phi(\cdot)$ and $\psi(\cdot)$ share the same neural network architecture and weights, effectively making them identical. This is commonly used in models like Sentence-T5 (Ni et al., 2021), where both the query and document are encoded using the same model, ensuring that the embeddings are aligned in the same semantic space.
  - **Asymmetric Dual Encoder:** Here, $\phi(\cdot)$ and $\psi(\cdot)$ can have different architectures or different weights, allowing for more flexibility in encoding queries and documents differently. An example of this approach is Dense Passage Retrieval (DPR) (Karpukhin et al., 2020), where $\phi(\cdot)$ (query encoder) and $\psi(\cdot)$ (document encoder) are often based on the same BERT model but may have distinct parameters or fine-tuning strategies to better capture the specific characteristics of queries and documents.

  Typically, dual encoder methods first employ BM25 as an initial retriever to select a set of candidate documents, which are then re-ranked based on the similarity of their embeddings to the query. However, these methods can also be used for direct ranking by computing the embeddings of all documents in advance and comparing them with the query embedding during retrieval.

- **Cross-Attentional Models:** In this approach, we use an encoder such as BERT on the combined input of the query and document, separated by a special $[SEP]$ token. The model then computes a relevance score based on a learned linear function applied to the encoding of the $[CLS]$ token. Given its high computational cost, Cross-Attentional BERT is typically utilized only for reranking purposes, as demonstrated in previous research by Nogueira and Cho (2020) and Yang et al. (2019).

- **Dense Retriever Models:** Dense retrievers, like the dual encoder, use dense vector representations. However, they may involve more sophisticated methods like late interaction models (e.g. ColBERT from Khattab and Zaharia (2020)) where the query and document representations interact late in the process rather than being simply concatenated.

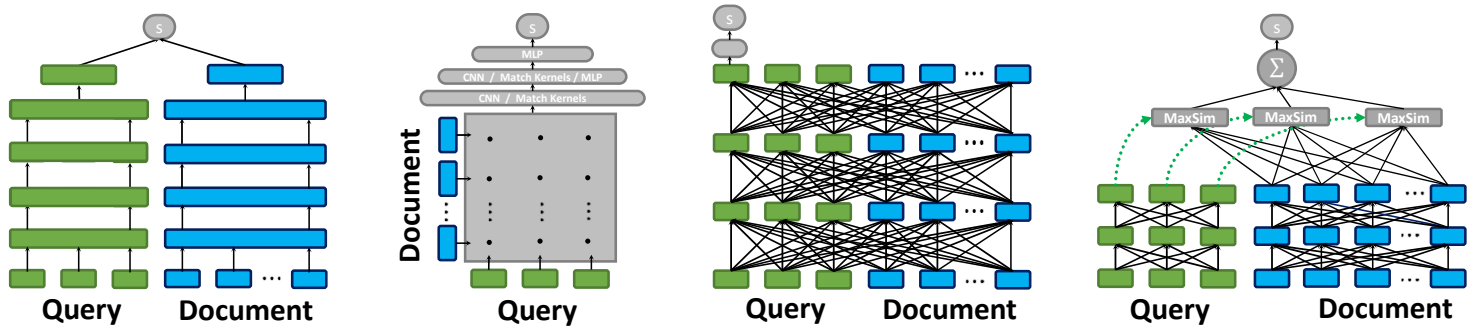
We illustrate these different types of IR models in Figure 1.

The relevance or similarity score $Sim(\mathbf{q}, \mathbf{d})$ between a query and a document is then computed using a similarity function, such as the cosine similarity:

$$Sim(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} \tag{2}$$

Here, $\mathbf{q} \cdot \mathbf{d}$ represents the dot product of the query vector $\mathbf{q}$ and the document vector $\mathbf{d}$, while $\|\mathbf{q}\|$ and $\|\mathbf{d}\|$ are the norms of these vectors. The cosine similarity score ranges from -1 to 1, where a score of 1 indicates perfect similarity (i.e. the vectors are aligned), and a score of 0 indicates no similarity (i.e. the vectors are orthogonal).

This similarity score is then used to rank the documents, with those having higher scores being considered more relevant to the query. This two-stage process allows IR systems to balance speed and accuracy, providing users with precise and contextually appropriate search results.



(a) Representation-based Similarity *(e.g., DSSM, SNRM)*

(b) Query-Document Interaction *(e.g., DRMM, KNRM, Conv-KNRM)*

(c) All-to-all Interaction *(e.g., BERT)*

(d) Late Interaction *(i.e., the proposed ColBERT)*

Figure 1: The figure from (Khattab and Zaharia, 2020) illustrates different neural interaction models used in information retrieval, focusing on how queries and documents interact within these models. Figure 1a demonstrates a **Representation-based Similarity** model (e.g. DSSM (Huang et al., 2013), SNRM (Zamani et al., 2018)), where query and document representations are independently computed and then compared. Figure 1b depicts a **Query-Document Interaction** model (e.g. DRMM (Guo et al., 2016), KNRM (Xiong et al., 2017), Conv-KNRM (Dai et al., 2018)), where each token in the query interacts directly with each token in the document. Figure 1c shows an **All-to-all Interaction**model (e.g. BERT), where every token in the query interacts with every token in the document, capturing comprehensive token-level relationships. Finally, Figure 1d illustrates a **Late Interaction** model (e.g. ColBERT), where queries and documents are first independently encoded, with interaction occurring later through vector comparison. Each model represents a different strategy for handling the interaction between queries and documents in neural retrieval tasks.

2.2 Deep Learning-based recommendation

In this section, we will mainly focus on one task of recommendation which is sequential recommendation. Assume that we have a set of items, denoted $\mathcal{X}$, where $x \in \mathcal{X}$ represents an item. A user has a sequential sequence interaction with items: $\{x_1, x_2, x_3, \ldots, x_n\}$ called the user's history with n the number of interactions and x_i the i-th item that the user has interacted with. The goals of sequential recommendation is to predict the next item x_{n+1}.

In modern deep learning methods for sequential recommendation, models have evolved to capture complex patterns in user interactions by leveraging advanced neural architectures. Initially, techniques such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) were employed to model user sequences, with RNNs excelling in capturing long-term dependencies through their hidden states (Hidasi et al., 2016) , and CNNs like Caser (Tang and Wang, 2018) treating sequences as images to extract high-order transitions. However, these methods faced challenges with efficiency and scalability, particularly in handling long sequences or large datasets.

To address these challenges, recent advancements have shifted the focus towards attention-based models, which provide greater flexibility and efficiency. A notable example is the Self-Attentive Sequential Recommendation (SASRec) (Kang and McAuley, 2018) model, which introduced the use of self-attention mechanisms for sequential recommendation. Unlike RNNs, SASRec does not rely on recurrent structures, making it more efficient and capable of parallel processing. The self-attention mechanism in SASRec allows the model to adaptively focus on the most relevant items in a user's history, effectively capturing both short-term and long-term dependencies. This approach not only enhances performance across various datasets, particularly those with varying sparsity levels, but also improves scalability, making SASRec significantly faster than previous CNN and RNN-based models.

Furthermore on the success of attention-based models, transformer-based architectures like BERT have also been adapted for recommendation tasks. One such model, BERT4Rec (Sun et al., 2019),

applies BERT to sequential recommendation by predicting the next item in a sequence using masked item modeling. Unlike traditional sequential models that consider only past interactions, BERT4Rec exploits bidirectional context from both preceding and following items in the sequence. This bidirectional approach allows for a more nuanced understanding of user behavior, capturing complex item interactions and dependencies within a user's history. BERT4Rec has demonstrated superior performance across various benchmark datasets, making it a powerful tool in modern recommendation systems, particularly for tasks that require deep contextual understanding.

The transition towards attention-based methods, exemplified by models like SASRec and BERT4Rec, reflects a broader trend in modern recommendation systems. These models utilize powerful and scalable architectures that can adapt to the intricacies of user behavior over time, ultimately offering more personalized and accurate recommendations.

2.3 Incorporating Large Language Models for IR and Recommendation

Both search and recommendation systems have increasingly adopted attention-based methods, particularly with models like BERT, due to their ability to capture deep semantic relationships. With the growing popularity and effectiveness of Large Language Models (LLMs) such as GPT-3 and GPT-4 (OpenAI, 2024) and open-source model such as Llama2 models (Touvron et al., 2023) and Mistral's models (Jiang et al., 2023), the transition to incorporating these models in search was inevitable. LLMs have brought transformative capabilities to IR, allowing systems to better understand and process complex queries, thereby generating more context-aware and accurate search results.

The integration of LLMs into search systems has extended beyond traditional query matching. LLMs are now being utilized as zero-shot rankers (Brown et al., 2020), where they rank documents or responses without requiring specific task fine-tuning, leveraging their extensive pre-training across diverse data sources. This capability allows search engines to improve the relevance of results even in the absence of task-specific training data. Additionally, LLMs enhance various components of search systems, such as query rewriting, retrieval, reranking, and even summarization of results, thereby streamlining the search process and improving the overall user experience.

This capacity of LLMs were also highlighted for recommendations in Hou et al. (2024b). This approach applies the broad generalization capabilities of LLMs to perform recommendation tasks in a zero-shot manner, providing a significant leap in flexibility and reducing the need for extensive training data. While promising, these zero-shot methods have also highlighted some limitations, such as the need for further alignment between LLMs and specific recommendation tasks to achieve optimal performance.

To address these challenges, frameworks like TALLRec (Bao et al., 2023) have been proposed, which tune LLMs specifically for recommendation tasks by aligning them with recommendation data. This approach allows LLMs to better understand and predict user preferences by leveraging their vast language understanding capabilities, combined with domain-specific fine-tuning.

Moreover, models like RecFormer (Li et al., 2023a) further adress this issue by leveraging Pre-trained Language Models (PLMs) to encode textual information directly. Instead of relying on traditional collaborative filtering techniques that depend heavily on user-item interactions or matrix factorization, the authors modify the embeddings produced by PLMs to adapt them specifically for recommendation tasks.

The incorporation of LLMs into search has marked a new era in IR and Recommendation, where the paradigm has shifted from traditional matching techniques to generative AI. This shift combines the strengths of traditional methods with the power of advanced neural architectures, leading to more dynamic and user-friendly search experiences. As LLMs continue to evolve, their role in not only matching but also generating relevant, context-aware content will become increasingly central, driving the next generation of search technologies toward even greater accuracy, relevance, and efficiency.

3 Generative Information Retrieval

3.1 Preliminaries: From Matching to Generation

Generative Information Retrieval (GenIR) marks a significant evolution in the field of IR, transitioning from traditional matching-based methods to a paradigm where relevant information is generated directly by advanced models. This approach uses the capabilities of LLMs to generate document identifiers or entire responses to user queries, rather than relying on predefined indices and similarity metrics.

In GenIR, the retrieval process is reimagined. Instead of merely matching a query against a pre-indexed document list, generative models, such as T5 (Raffel et al., 2023), BART (Lewis et al., 2020), and GPT (Radford and Narasimhan, 2018), can be trained to generate relevant document identifiers (DocIDs) or responses directly from the input query. Document identifiers (DocIDs) are unique codes or strings assigned to documents within a corpus, which allow them to be efficiently retrieved. This generative approach enables a more seamless and integrated retrieval process, bypassing the traditional need for extensive document indexing and matching.

Two key areas of focus within GenIR are **Generative Document Retrieval (GDR)** and **Reliable Response Generation**:

1. Generative Document Retrieval (GDR): In this approach, generative models learn to produce document identifiers corresponding to the most relevant documents based on a given query. This method utilizes the model's internal parameters as a form of memory, allowing it to recall and generate relevant DocIDs without the need for external indexing structures. This direct generation of identifiers leads to more efficient and context-aware retrieval, capitalizing on the deep understanding and generative capabilities of LLMs.

2. Reliable Response Generation: Beyond simply retrieving documents, another crucial aspect of GenIR is generating reliable and contextually appropriate responses directly from the model. This involves using generative models to not only identify relevant documents but also synthesize coherent and accurate answers to user queries. Reliable response generation is particularly important in applications where users expect precise and informative answers, such as in conversational agents, customer support, or educational tools. Ensuring the reliability of these generated responses is critical, as it impacts user trust and the overall effectiveness of the IR system.

In this report, we will specifically focus on GDR as a pivotal element of GenIR. By exploring GDR, we aim to enhance its capabilities to support multi-task applications, including product search and sequential recommendation. This approach seeks to simplify and elevate the document retrieval process, making it more adaptable and efficient across various tasks. Additionally, we acknowledge the importance of reliable response generation as a complementary focus within GenIR, ensuring that the information retrieved and presented to users is both relevant and trustworthy. Together, these areas represent the forefront of innovation in modern IR systems.

3.2 Model Training

Model training is a critical component in the development of generative retrieval models, designed to enhance the model's ability to memorize and accurately retrieve documents. The training process typically involves mapping queries to relevant DocIDs using sequence-to-sequence (seq2seq) training methods. This approach is foundational to generative retrieval models such as DSI, NCI (Wang et al., 2023a), and SEAL (Bevilacqua et al., 2022), which have demonstrated significant performance improvements in retrieval tasks.

3.2.1 Indexing

In traditional IR systems, indexing involves the creation of a structured repository where documents are assigned unique identifiers, known as DocIDs. These identifiers allow for efficient retrieval by matching user queries against a pre-constructed index of documents. The index is a vital component, typically built using various techniques such as inverted indexing, where each term in the document corpus is associated with the documents that contain it. This approach enables fast retrieval by allowing the system to quickly locate relevant documents based on the presence and frequency of terms within a query.

However, in GenIR, the concept of indexing is fundamentally reimagined. Instead of relying on a static index with predefined DocIDs, GenIR systems dynamically generate these identifiers in response to user queries. This shift is driven by the capabilities of Large Language Models (LLMs), which can synthesize document identifiers or even entire responses based on their understanding of the query and the corpus. The generative approach allows these models to create DocIDs that are contextually relevant and adaptable, eliminating the need for a traditional, static index.

The process of indexing in GenIR involves two primary steps: **indexing methods** (e.g. how to index) and **indexing targets** (e.g. what to index). In **indexing methods**, the focus is on how the content of a document is linked to its generated identifier. Unlike traditional systems where this link is established through predefined rules and structures, in GenIR, models learn these associations through training on document-query pairs. This method exploits the model's ability to internalize and recall document content, enabling it to generate identifiers that are both unique and relevant to the query at hand.

In **indexing targets**, the emphasis shifts to how documents are represented within the generative model. Given the constraints of model capacity and the computational cost of processing entire documents, GenIR systems often use various strategies to represent documents effectively. These strategies may include selecting key sections of a document, generating synthetic queries as proxies for document content, or employing compressed representations that capture the essence of the document without overwhelming the model.

The indexing process can be formally described as: $r_{d_j} \rightarrow j$, where r_{d_j} represents the document $d_j \in \mathcal{D}$, and j is its corresponding identifier. The representation r_{d_j} can be, for example, the first L tokens of the document. This process is typically trained using a simple sequence-to-sequence (seq2seq) approach. In this setup, the model takes the tokens from r_j as input and generates the associated identifier as output.

3.2.2 Retrieval

In the retrieval phase of generative retrieval, the model is trained using labeled query-DocID pairs. The goal during inference is for the model to retrieve the correct document given a specific query. In this phase, the model generates DocIDs in response to queries by utilizing the associations it learned during the indexing phase. The main objective is to ensure that the most relevant documents are retrieved and ranked according to their relevance to the user's query.

The generation of DocIDs is performed in an autoregressive fashion, meaning that the model generates each identifier sequentially, conditioned on the previously generated identifiers. Specifically, the model is optimized using a seq2seq cross-entropy loss, and training is conducted with teacher forcing. The loss function is defined as:

$$\mathcal{L}_\theta = \sum_{(q,d)\in\mathcal{D}} \sum_{t}^{m} \log P(id_t | q, id_{<t}; \theta) \tag{3}$$

Here, θ are the trainable parameters of the model, id_t is the t-th token in the currently given DocID id of length m.

The retrieval process can be enhanced through several data augmentation methods such as:

1. **Sampling Document Pieces as Pseudo Queries:** DynamicRetriever (Zhou et al., 2022a) utilizes passages, sampled terms and n-grams to serve as pseudo queries to enhance the model's memorization of DocIDs.

2. **Pseudo Query Generation:** Pseudo queries generated from document content can also be used during retrieval training to further improve the model's ability to generalize and handle a wide range of queries. NCI (Wang et al., 2023a) uses DocT5Query (Cheriton, 2019). Similarly, DSI-QG (Zhuang et al., 2023) trains a Query-Generation model and then uses a random sampling strategy to generate a set of queries for each document in order to have more diverse and creative queries.

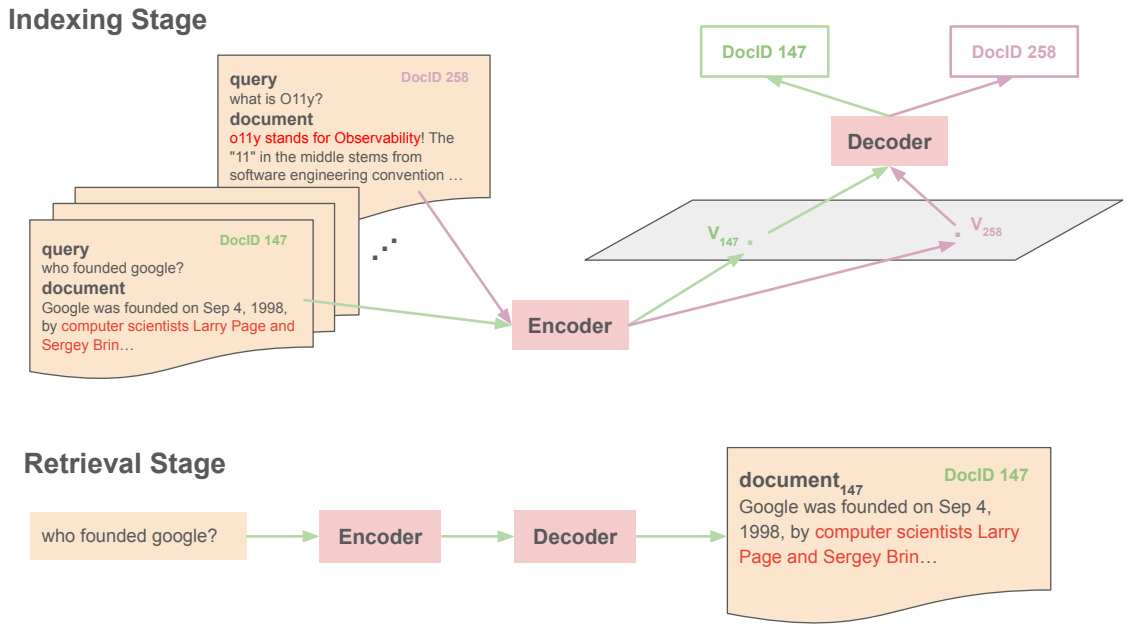


Figure 2: The figure from (Kuo et al., 2024) illustrates the two training stages. During the Indexing Stage, a sequence-to-sequence (seq2seq) learning system is used to link specific queries, such as "What is O11y?" and "Who founded Google?", with their corresponding document identifiers (DocIDs 258 and 147, respectively). This ensures precise associations between queries, DocIDs, and documents. In the Retrieval Stage, when a user submits a query like "Who founded Google?", the system autoregressively generates the relevant DocID directly. This eliminates the need for further query processing or document reranking, demonstrating the system's efficiency in end-to-end retrieval based on the learned query-to-DocID relationships.

3.3 Model Structure

The retrieval process in Generative Document Retrieval (GDR) involves autoregressively generating document identifiers (DocIDs). During inference, one of the most common decoding methods used in GDR is Constrained Beam Search. This method restricts the generated DocIDs to valid entries within the corpus, ranking them based on their generation probabilities to produce an ordered list of DocIDs. While Constrained Beam Search is widely used in natural language processing (NLP), specific decoding methods have been developed to better suit the unique needs of GDR.

For example, the semantic structured DocIDs proposed by models like DSI led to the development of the Prefix-Aware Weight-Adaptive (PAWA) decoder structure by NCI (Wang et al., 2023a). PAWA

is designed to perceive the hierarchical prefixes of DocIDs and adjust the weights at different positions accordingly, enabling it to capture the semantic hierarchy embedded within DocIDs more effectively. Additionally, the Planning-Ahead Generation (PAG) (Zeng et al., 2024) approach, proposed by Zeng et al., introduces a method where DocIDs are constructed based on both set-based and sequence-based decoding. PAG first decodes the set-based DocID to approximate document-level scores and then continues with sequence-based decoding to refine the identifier, enhancing the accuracy and relevance of the retrieved documents.

As discussed in Section 3, most GDR models have employed an encoder-decoder transformer architecture, similar to models like T5, which is specifically designed to handle sequence-to-sequence (seq2seq) tasks. The encoder-decoder structure, exemplified by models like T5, has been highly effective in GenIR tasks due to its ability to manage complex language understanding and generation processes.

Despite the widespread adoption of Large Language Models (LLMs), the exploration of decoder-only architectures for generative retrieval remains relatively limited. Notably, (Li et al., 2024a) utilized LLaMA-2 to achieve superior performance across a range of downstream tasks. This highlights the potential advantages of using LLMs in generative retrieval, likely due to their enhanced reasoning abilities and instruction-following capabilities, which are enhanced by extensive pre-training on diverse datasets. The large scale of LLMs also offers the possibility of indexing larger corpora, thereby improving the scalability of generative retrieval systems. Moreover, the decoder-only architecture presents the opportunity to simplify the generative retrieval framework by unifying input and output representations.

Similarly, (Wang et al., 2024a) explored the application of LLMs in retrieval with the LLM2GR model, which eliminates the need for additional retrieval models by directly generating relevant document titles and passages from queries. This approach employs constrained decoding along with a scoring mechanism, demonstrating efficient and effective retrieval performance.

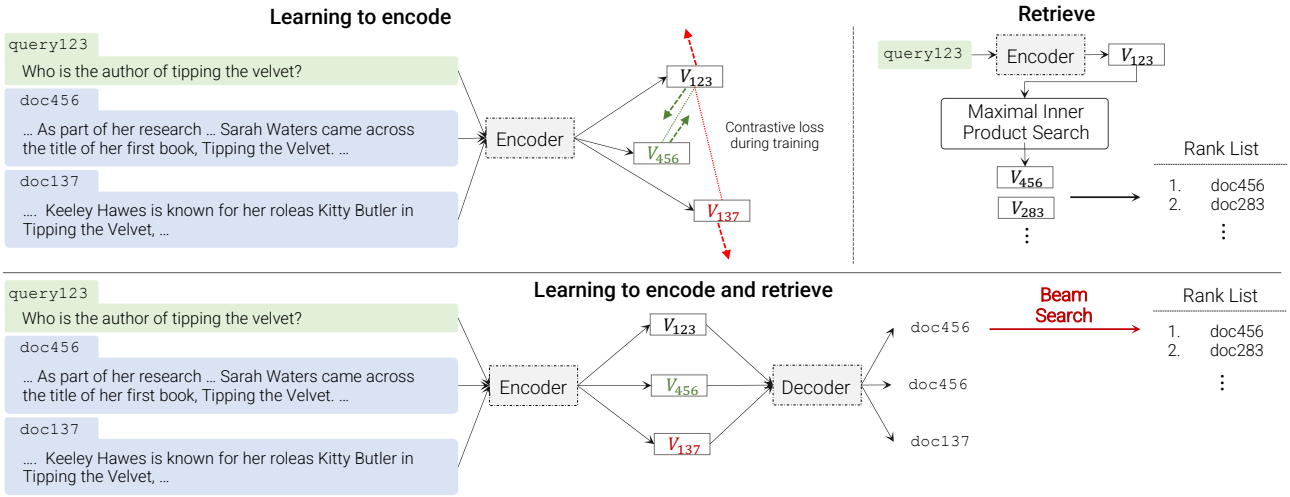


Figure 3: The figure from (Tay et al., 2022) illustrates the difference between dual encoders (top) and DSI (bottom).

3.4 Design of Document Identifiers

The design of document identifiers (DocIDs) is a critical aspect of generative retrieval models, as these identifiers serve as the link between the user's query and the relevant document. In traditional IR systems, DocIDs are often arbitrary or sequential numbers assigned to documents in a database. However, in GenIR systems, the design of these identifiers must be carefully considered to optimize the model's ability to generate them accurately.

In Generative Retrieval (GR), the design of document identifiers (DocIDs) is a critical aspect that directly influences the efficiency and effectiveness of the retrieval process. Unlike traditional retrieval methods that rely on pre-existing query-document relationships or complex ranking algorithms, GR uses generative models to map queries directly to document identifiers. This section explores the different strategies used for designing DocIDs, emphasizing the importance of their structure and representation in achieving high retrieval performance.

Numerical identifiers in GR involve representing each document with one or more tokens. These identifiers can be classified into two categories: **single-token identifiers** and **sequential-token identifiers**.

1. **Single-Token Identifiers:** In this simple approach, each document is assigned a unique token, typically a random integer or a class label. The generative model learns to decode these tokens to identify the relevant documents. This method, while straightforward, may not fully capture the semantic relationships between documents, potentially limiting retrieval accuracy. The main drawback is that this approach is not scalable, as it requires a unique token for each document, and datasets typically contain hundreds of thousands of documents, making it impractical for large-scale retrieval tasks. Examples of works/approaches that uses single-token identifiers are: (Tay et al., 2022; Mehta et al., 2023; Nguyen and Yates, 2023; Chen et al., 2023).

2. **Sequential-Token Identifiers:** This approach uses a sequence of tokens to represent each document, with the tokens being decoded sequentially. Sequential-token identifiers can further be divided into:

    - **Arbitrary Structured Identifiers:** These use random sequences of tokens without inherent semantic relationships, making it difficult for the model to generalize based on partial token sequences. Examples of work that uses arbitrary structured identifiers are: (Tay et al., 2022; Mehta et al., 2023; Zhuang et al., 2023; Nadeem et al., 2022).
    - **Semantically Structured Identifiers:** In this approach, sequences of tokens are designed so that documents with similar semantic content share the same initial subsequence, effectively clustering semantically related documents. This method reduces the search space and enhances retrieval efficiency by embedding semantic relationships directly into the identifier structure. One common technique, used in (Tay et al., 2022), employs k-means clustering to group similar documents hierarchically, allowing the generative model to retrieve documents sequentially within these clusters. Another approach, proposed in (Sun et al., 2023), uses an autoencoder architecture to encode documents into DocIDs and regenerate them by decoding these identifiers. Additionally, advanced methods like those in (Zhou et al., 2022b), (Ren et al., 2023), and (Zhang et al., 2023a) further refine this process by leveraging techniques such as Product Quantization (PQ) (Jégou et al., 2011) and Residual Quantization (RQ) (Martinez et al., 2014) to create semantically structured identifiers, thereby optimizing large-scale document retrieval.

3. **Learnable DocIDs:** Recent approaches have introduced learnable document representations to improve retrieval performance. GenRet (Sun et al., 2023) encodes documents into short, discrete DocIDs using a discrete autoencoder, with optimization through progressive training and diversity clustering. Tied-Atomic (Nguyen and Yates, 2023) links document text with token embeddings and uses contrastive loss for DocID creation, combining generative models' expressiveness with dense retrieval's efficiency. LMIndexer (Jin et al., 2023) and ASI (Yang et al., 2023) learn optimal DocIDs through semantic indexing, ensuring similar content aligns under common DocIDs. RIPOR (Zeng et al., 2023) encodes document content into vectors, splits them via Residual Quantization (RQ), and uses prefix-guided ranking optimization to enhance search precision. These methods use dense retriever embeddings and techniques like k-means, PQ, and RQ to create semantically meaningful DocID sequences, combining the strengths of encoder-decoder generative retrieval and bi-encoder dense retrieval models.

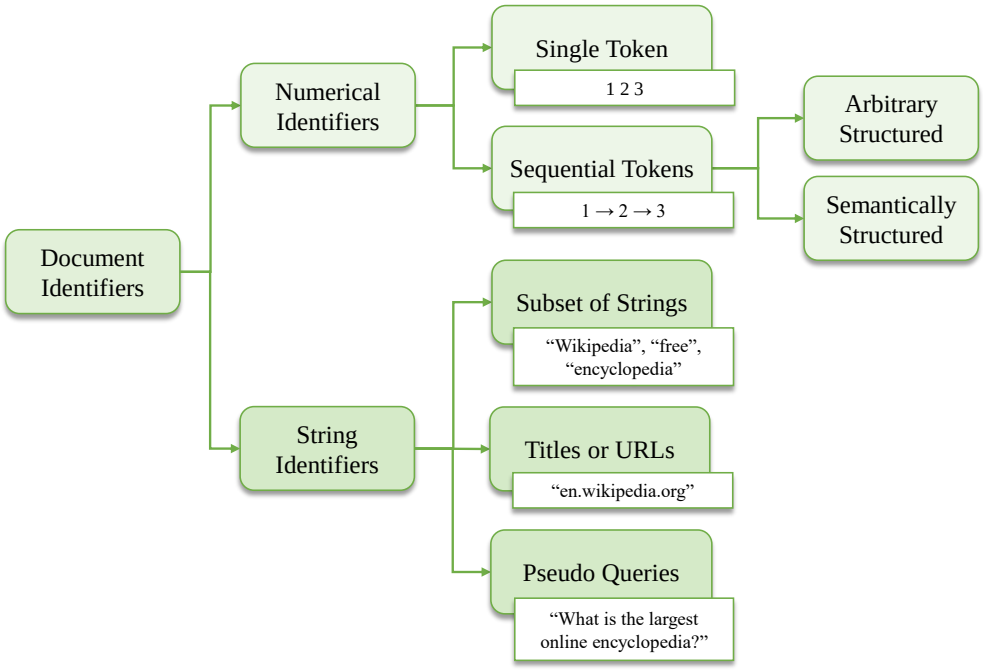


Figure 4: Different types of document identifiers, figure from (Kuo et al., 2024).

String identifiers differ from numerical ones by using text-based identifiers that inherently contain semantic information. These can be categorized as follows:

1. **Sub-strings of Documents:** To enhance the flexibility of DocIDs, SEAL (Bevilacqua et al., 2022) introduced a sub-string identifier approach, representing documents with any N-grams within them. Utilizing the FM-Index, SEAL retrieves all documents containing the generated N-grams, scoring and ranking them based on the frequency and importance of these N-grams in the query. This approach has been adopted by various GR models (Li et al., 2023b; Nguyen and Yates, 2023; Li et al., 2024b, 2023c). For more comprehensive document representation, MINDER (Li et al., 2023c) proposed multi-view identifiers that combine generated pseudo queries, titles, and sub-strings, although this method requires more memory and inference time. Similar multi-DocID approaches were used in LTRGR (Sun et al., 2023) and DGR (Li et al., 2024b), while SE-DSI (Tang et al., 2023) demonstrated the effectiveness of using pseudo queries alone as DocIDs.

2. **Document Titles:** Document titles are the most straightforward text-based identifiers, where each title uniquely represents a document within the corpus. This approach works well with datasets like the Wikipedia corpus used in the KILT benchmark (Petroni et al., 2021), where each document has a unique, manually annotated title. GENRE (Cao et al., 2021) adopts titles as DocIDs, using the generative model BART with pre-built DocID prefixes, achieving superior retrieval performance across 11 KILT datasets. Subsequent works, such as GERE (Chen et al., 2022a), CorpusBrain (Chen et al., 2022b), Re3val (Song et al., 2024), and CorpusBrain++ (Guo et al., 2024), also utilized title-based DocIDs for Wikipedia-based tasks. LLM-URL (Ziems et al., 2023) extended this approach by generating URLs directly using ChatGPT, yielding strong results after filtering out invalid URLs. However, in web search scenarios, document titles often suffer from duplication and

lack of meaningful content, making them unsuitable as standalone DocIDs. Ultron (Zhou et al., 2022b) effectively addressed this issue by combining URLs and titles as DocIDs, uniquely identifying documents through key elements in web page URLs and titles.

3. **Term Sets:** Unlike the sequential DocIDs, term set-based document representations use keywords extracted from titles and content rather than predefined sequences. AutoTSG (Zhang et al., 2023b) proposed this approach, enabling retrieval of a target document as long as the generated term set matches the document's identifiers. During inference, AutoTSG employs a constrained greedy search to ensure each generated term effectively identifies a specific document. Similarly, PAG (Zeng et al., 2024) constructed DocIDs based on sets of key terms, ignoring term order, which allows for approximated document-level scoring during decoding. This approach enhances flexibility in document retrieval by focusing on the presence of key terms rather than their sequence.

4. **Synthetic Pseudo-Queries:** Tang et al. (2023) utilized a generation model (Doc2query by (Nogueira et al., 2019)) to create pseudo-queries for each document, using the top generated query as the document identifier for retrieval. During retrieval, the rank list is generated via beam search, and to handle cases where identifiers are not unique, a random order is applied to documents sharing the same DocID during inference. Additionally, Li et al. (2023c) proposed a multi-view identifier that combines titles, substrings, and pseudo-queries for more accurate document identification, an approach also utilized in subsequent work. Li et al. (2023b) and Ren et al. (2023) noted that string-based identifiers are advantageous as they align closely with the tokens used in pretraining language models. While arbitrary DocIDs are easier to create, identifiers containing semantic information, such as pseudo-queries, generally offer more reliable performance across various retrieval scenarios.

5. **Learnable DocIDs:** Text-based identifiers can also be designed to be learnable, allowing the model to dynamically adapt and optimize document representations. For instance, NOVO (Wang et al., 2023b) introduced learnable continuous N-grams that form term-set DocIDs. The model employs a Denoising Query Modeling (DQM) task, which teaches it to generate queries from documents with added noise, implicitly learning to filter out and emphasize N-grams that are more relevant to the queries. Additionally, NOVO updates the semantic representations of document identifiers by refining the embeddings of N-grams during the retrieval task. Following this, GLEN (Lee et al., 2023) advances the concept by designing dynamic lexical DocIDs through a two-phase index learning strategy. In the first phase, keyword-based DocID assignment, GLEN extracts and learns keywords from documents using self-supervised signals. In the second phase, ranking-based DocID refinement, GLEN directly integrates query-document relevance into dynamic DocIDs through two specialized loss functions. During inference, GLEN employs a collision-free inference mechanism, ranking documents based on DocID weights efficiently, without adding extra computational overhead.

The design of document identifiers in Generative Retrieval is a highly dynamic and evolving area of research, with no clear consensus on the best approach. Various strategies have been proposed, ranging from numerical identifiers to string-based identifiers. The introduction of learnable DocIDs, which dynamically adapt to the content and context of documents, represents a significant advancement in this field, offering new possibilities for improving retrieval performance.

Despite these innovations, the optimal design of DocIDs remains an open question, with each approach presenting its own strengths and challenges. As GenIR systems continue to develop, it is likely that new methodologies will emerge, further refining our understanding of how best to link queries with relevant documents. The ongoing exploration of different DocID designs underscores the complexity of the task and highlights the importance of continued research in this critical aspect of information retrieval.

3.5 Applications to recommendations

Generative models in recommendation systems represent a paradigm shift from traditional discriminative approaches, moving away from ranking and indexing items based on predefined scores. Instead, generative models directly produce item identifiers based on user queries or historical interaction sequences, enabling more flexible and adaptive recommendations. These models, such as P5 (Geng et al., 2023), TIGER (Rajput et al., 2023), and IDGenRec (Tan et al., 2024), leverage natural language processing and semantic information to create unique, content-rich identifiers that align closely with both user preferences and item attributes. By integrating collaborative filtering with semantic information, as seen in models like ColaRec (Wang et al., 2024b) and SEATER (Si et al., 2024), generative recommendation systems offer a more holistic approach to predicting user preferences, making them increasingly effective in delivering personalized and contextually relevant recommendations.

3.6 Evaluation

3.6.1 Metrics

Evaluating the performance of generative models in information retrieval (IR) and recommendation systems requires robust metrics that can capture both the accuracy and relevance of the results. Several standard evaluation metrics are widely used to assess how well these models perform in ranking and retrieving the most relevant items or documents. Below is an overview of the key metrics commonly used in the evaluation of generative retrieval and recommendation systems.

Recall at Rank K (Recall@K) measures the proportion of relevant items that are successfully retrieved within the top K results. It is particularly useful in scenarios where the emphasis is on ensuring that as many relevant items as possible are included in the retrieved set. Recall@K is calculated as:

$$\text{Recall@K} = \frac{\text{Number of relevant items in top } K \text{ results}}{\text{Total number of relevant items in the dataset}} \tag{4}$$

This metric is crucial for understanding the completeness of the retrieval system, especially in contexts like recommendation systems where capturing as many relevant items as possible within a short list is important for user satisfaction.

Normalized Discounted Cumulative Gain (nDCG@K) is a metric that considers the position of relevant items in the ranked list, assigning higher importance to items appearing earlier in the list. nDCG@K is particularly valuable because it reflects both the relevance and the ranking order, rewarding systems that rank relevant items higher. The metric is calculated in two steps:

1. **Discounted Cumulative Gain (DCG@K)**: This measures the gain of each relevant item, discounted logarithmically based on its position in the list.

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{\text{rel}_i}{\log_2(i+1)} \tag{5}$$

   where rel_i is the relevance score of the item at position i.

2. **nDCG@K**: This normalizes the DCG score by the ideal DCG (IDCG), which is the maximum possible DCG score for the top K items.

$$\text{nDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{6}$$

nDCG@K is widely used in both IR and recommendation systems because it not only considers the presence of relevant items but also rewards their placement higher in the ranked list.

Mean Average Precision (MAP) provides a single-figure measure of quality across recall levels. It calculates the average precision for each query and then averages these values across all queries. Average Precision (AP) for a single query is computed by averaging the precision at the ranks where relevant items are found:

$$\text{AP} = \frac{1}{N} \sum_{k=1}^{N} \text{Precision@k} \times \text{rel}_k \tag{7}$$

where N is the number of relevant items, Precision@k is the precision at the k-th rank, and rel_k is a binary indicator of relevance (1 if the item at rank k is relevant, 0 otherwise).

Precision@k is defined as the proportion of relevant documents in the top k ranked results, and is given by:

$$\text{Precision@k} = \frac{\text{Number of relevant items in top } k \text{ results}}{k} \tag{8}$$

MAP is then calculated as:

$$\text{MAP} = \frac{1}{Q} \sum_{q=1}^{Q} \text{AP}_q \tag{9}$$

where Q is the total number of queries.

Mean Reciprocal Rank (MRR) evaluates the ranking of the first relevant item in the list. It is particularly useful in scenarios where finding one relevant item quickly is more important than finding all relevant items. MRR is calculated as:

$$\text{MRR} = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{\text{rank}_q} \tag{10}$$

where rank_q is the position of the first relevant item for the q-th query. MRR emphasizes the importance of returning relevant results as early as possible in the ranked list.

3.6.2 Datasets

The evaluation of generative models in information retrieval (IR) and recommendation systems heavily relies on large and diverse datasets that reflect real-world scenarios. These datasets provide the necessary benchmarks to assess the performance of different models across various tasks. Below, we discuss some of the most commonly used datasets in both search-based IR and recommendation systems.

MS MARCO (Bajaj et al., 2018) comprises 1,010,916 questions sourced from Bing, each paired with human-generated answers, 182,669 rewritten answers, and a collection of 8,841,823 passages extracted from web documents. Generative retrieval research often utilizes different subsets of this dataset, varying in size (10k, 100k, 300k, 1M, full), to conduct a range of experimental studies.

Natural Questions (NQ) (Kwiatkowski et al., 2019) includes real Google queries paired with annotated answers from Wikipedia. Generative retrieval studies typically use subsets of 10k, 100k, or 320k examples for various experiments.

TREC Deep Learning (TREC DL) (Craswell et al., 2020)(Craswell et al., 2021) is a key dataset used in evaluating deep learning models for information retrieval. It provides a variety of query types and

document collections, making it a comprehensive benchmark for testing the effectiveness and scalability of generative retrieval models.

Amazon Reviews (He and McAuley, 2016)(Ni et al., 2019) (Hou et al., 2024a) is a large-scale collection of product reviews, metadata, and user interactions sourced from Amazon. It is widely employed in the evaluation of recommendation systems, particularly in tasks involving product recommendations and user preference prediction. There are many versions of the datasets, containing users reviews and items metadata from 1996 to 2014, 2018 and 2023. Each version has separated their reviews by domains such as "Beauty", "Games" or "Sports and Outdoors".

MovieLens (Harper and Konstan, 2015) is a widely used dataset in the recommendation systems domain, consisting of user ratings and metadata for movies. The dataset is available in various versions, ranging from 100,000 to 20 million ratings, with user counts ranging from 943 to 138493 and movie counts ranging from 1682 to 27278.

ESCI (Exact, Substitute, Complement, Irrelevant) (Reddy et al., 2022) dataset is a large-scale, multilingual dataset designed for improving product search quality. It contains around 130,000 unique shopping queries and 2.6 million query-product pairs manually labeled with relevance judgments based on four categories: Exact, Substitute, Complement, and Irrelevant. These categories indicate the relevance of a product in relation to a user's query.

Amazon C4 (Amazon Complex Contexts Created by ChatGPT) (Hou et al., 2024a) is a semi-synthetic dataset designed to evaluate models' ability to handle complex natural language in product search scenarios. It is constructed by rephrasing user reviews into more detailed and context-rich queries using ChatGPT. The dataset is meant to assess models' performance in retrieving relevant items based on long, complex queries, in contrast to traditional keyword-based search datasets.

4 Model Implementation

4.1 Overview of the Approach

Our approach builds upon the Transformer Index for Generative Recommenders (TIGER) (Rajput et al., 2023) framework, adapting it to a multitask setting using a pretrained T5-small/base model (Raffel et al., 2023). The core innovation lies in unifying various tasks, including sequential recommendation, semantic indexing, and product search by leveraging the generative capabilities of the T5 model.

4.1.1 Multitask Learning Paradigm

The proposed system is designed to perform the following tasks:

- **Next Item Prediction (Sequential Recommendation)**: Our model predicts the Semantic ID of the next item a user might interact with based on their interaction history. This is formulated as a generative task where the model learns to output Semantic IDs sequentially, capturing the temporal dynamics of user interactions.
- **Product Search**: As a downstream task, the model uses the generated Semantic IDs to perform product search. This task involves retrieving relevant items based on a user query, which could be a natural language description or specific attributes. The use of a pretrained T5 model enables the system to understand and process complex queries more effectively, improving the relevance of the search results.

Semantic ID Indexing: Following LC-Rec (Zheng et al., 2024), we add one tuning task to enhance the association between the Semantic ID and its description.

- **Item Description to Index**: This task involves generating the Semantic ID from an item description.

These indexing tasks are crucial for understanding and verifying the content associated with each identifier, thereby ensuring accurate retrieval and recommendation. To evaluate the impact of these indexing tasks, we conducted an ablation study, the details of which are presented in Section

4.1.2 Main systems' components - retrieval versus recommendation

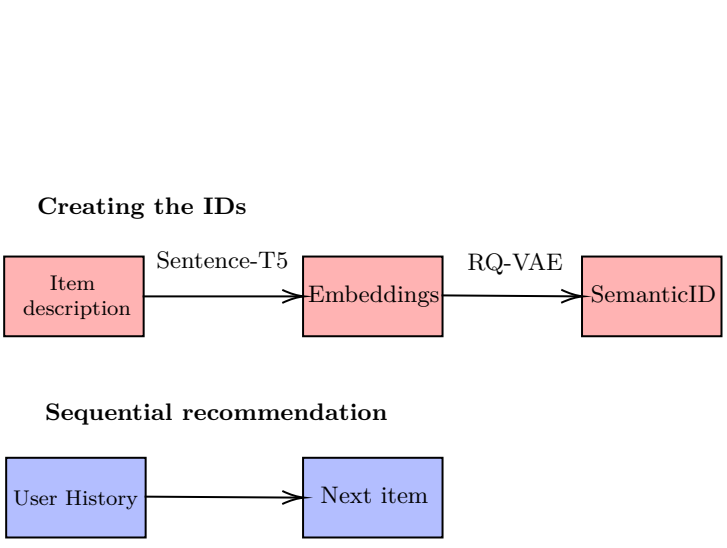


Figure 5: Training framework of TIGER for sequential recommendation.

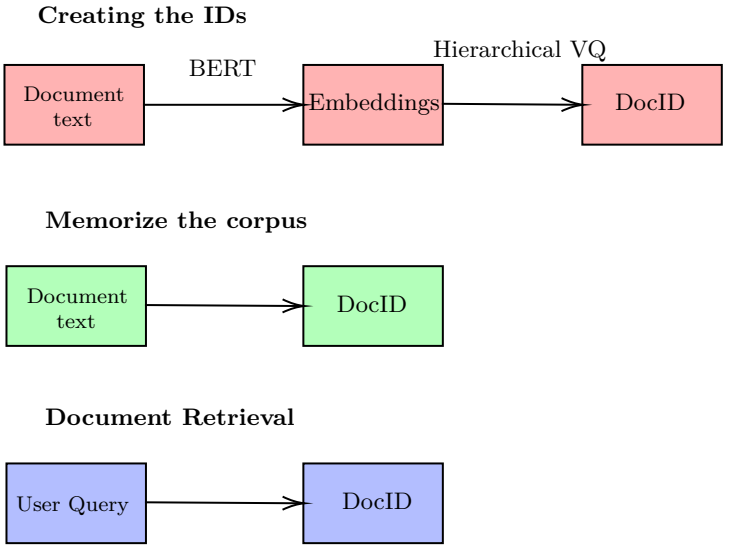


Figure 6: Training framework of DSI for document retrieval.

Figure 7: This figure provides a detailed comparison of the ID creation and retrieval processes in the TIGER and DSI models. In TIGER, item descriptions are encoded into Semantic IDs via Sentence-T5 and RQ-VAE, and then directly traing for the downstream task. Conversely, DSI introduces a memorization step and then is trained for document retrieval.

4.1.3 Generative Retrieval and Semantic ID Representation

Following the TIGER framework, our approach uses Semantic IDs, which are tuples of discrete semantic tokens derived from item features. These IDs are generated using a residual quantization namely RQ-VAE (Zeghidour et al., 2021) of embedding that are created using a pretrained Sentence-T5 model (Ni et al., 2021), ensuring that similar items have closely related Semantic IDs. This representation aids in knowledge sharing across similar items.

The T5 model's versatility allows it to handle multiple tasks in a unified manner. By training the model on a shared task set, we leverage transfer learning, where knowledge gained from one task can enhance performance in others. This multitask training setup not only improves the model's efficiency but also its adaptability to various scenarios, from sequential recommendations to dynamic product search.

4.1.4 Training and Implementation

The model is trained end-to-end using the T5 architecture, an encoder-decoder framework designed for sequence-to-sequence (seq2seq) tasksDuring training, the model learns to predict Semantic IDs for the next item in a sequence, in an autoregressive manner, predicting the next token, given a user's history, and to map between item descriptions and their corresponding Semantic IDs. The product search task is integrated into this framework, allowing the model to directly retrieve items based on user queries.

Our implementation ensures that the Semantic ID space is both compact and semantically meaningful, facilitating efficient retrieval and ranking.

4.2 Semantic ID Generation

The Semantic ID Generation process in our approach is a critical component that provides a compact and meaningful representation of items. This representation, called the Semantic ID, is derived using a Residual Quantized Variational AutoEncoder (RQ-VAE). The RQ-VAE helps in converting high-dimensional item features into a low-dimensional tuple of discrete codewords, facilitating efficient retrieval and generalization.

4.2.1 Residual Quantized Variational AutoEncoder (RQ-VAE)

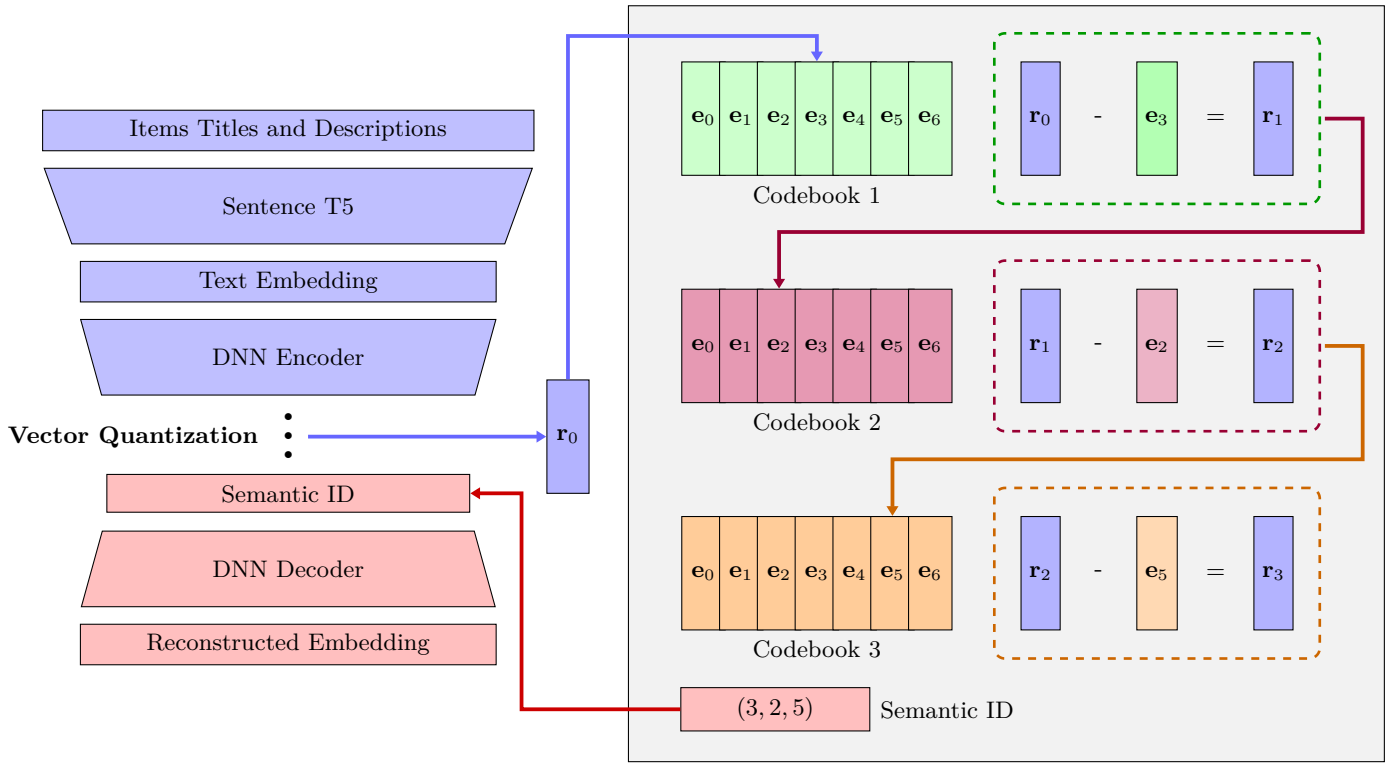


Figure 8: In this figure, the item titles and descriptions are provided as input to the Sentence-T5 model. The model embeds this input into a vector, which is then quantized using residual quantization. We employ three codebooks, each of size 7. The initial embedding $\mathbf{r}_0$ is closest to vector $\mathbf{e}_3$ in the first codebook, and this is used to compute the residual for subsequent quantization stages. This process continues across all codebooks, resulting in the final Semantic ID of $(3, 2, 5)$. We train our RQ-VAE to then reconstruct from the Semantic ID the original embedding.

The RQ-VAE model consists of an encoder, a residual quantizer, and a decoder Figure 5. The process can be summarized in the following steps:

1. **Embedding with a Text Encoder**: Each item's textual description or other content features are first processed using Sentence-T5. We choosed a a pre-trained Sentence-T5 model, which converts these inputs into a semantic embedding vector $\mathbf{x} \in \mathbb{R}^{\mathbf{d}}$, where $\mathbf{d}$ represents the dimensionality of the embedding space. This embedding captures the essential semantic information of the item.

2. **Encoding to Latent Representation**: The embedding vector $\mathbf{x}$ is then passed through the encoder component of the RQ-VAE, which transforms it into a latent representation $\mathbf{z} = E(\mathbf{x})$. This latent representation serves as a compressed and abstract form of the item's features, which will be further processed by the quantizer.

3. **Residual Quantization**: The latent representation $\mathbf{z}$ is then quantized into a tuple of codewords:

   (a) At the zero-th level, the initial residual $\mathbf{r}_0$ is set to the latent representation $\mathbf{z}$. The nearest vector $\mathbf{e}_{\mathbf{c}_0}$ from the first level codebook $\mathbf{C}_0$ is chosen such that $\mathbf{c}_0 = \arg\min_{\mathbf{i}} \|\mathbf{r}_0 - \mathbf{e}_i\|$, where $\mathbf{e}_i$ are vectors in the codebook.

   (b) The residual $\mathbf{r}_1$ is then computed as $\mathbf{r}_1 = \mathbf{r}_0 - \mathbf{e}_{\mathbf{c}_0}$. This process is repeated for subsequent levels, where each level refines the quantization using the residual from the previous level. The quantization process continues up to $\mathbf{m}$ levels, resulting in a sequence of codewords $(\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{m-1})$ that collectively form the Semantic ID of the item.

   This hierarchical quantization approach allows for a more granular representation, with each level providing additional detail. The use of multiple levels helps in capturing fine-grained distinctions between items, which is crucial for effective retrieval.

4. **Decoding**: The final quantized representation $\hat{\mathbf{z}}$ is obtained by combining the codewords from all levels. The decoder then attempts to reconstruct the original semantic embedding from $\hat{\mathbf{z}}$, ensuring that the quantization process preserves the critical information about the item. The reconstruction loss, along with the quantization loss, is minimized during training, helping to fine-tune the encoder and the codebook vectors.

The training of the RQ-VAE involves minimizing a composite loss function that ensures both accurate reconstruction of the input and effective quantization. The total loss, $\mathcal{L}_{\text{total}}$, consists of two main components:

- **Reconstruction Loss** ($\mathcal{L}_{\text{recon}}$): This term measures the difference between the original input embedding $\mathbf{x}$ and the reconstructed embedding $\hat{\mathbf{x}}$, obtained from decoding the quantized latent representation $\mathbf{z}$. It is typically defined as the squared Euclidean distance:

$$\mathcal{L}_{\text{recon}} := \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \tag{11}$$

- **Quantization Loss** ($\mathcal{L}_{\text{quant}}$): This component penalizes the discrepancy between the continuous latent vectors and their quantized versions. It includes both the commitment loss, which encourages the encoder output to stay close to the quantized embeddings, and the codebook loss, which encourages the codebook vectors to be updated towards the latent representations. The quantization loss can be expressed as:

$$\mathcal{L}_{\text{quant}} := \sum_{d=0}^{m-1} \|\text{sg}[\mathbf{r}_d] - \mathbf{e}_{\mathbf{c}_d}\|^2 + \beta \|\mathbf{r}_d - \text{sg}[\mathbf{e}_{\mathbf{c}_d}]\|^2 \tag{12}$$

  where $\text{sg}[\cdot]$ denotes the stop-gradient operator (van den Oord et al., 2018) that prevents gradients from flowing through certain pathways during backpropagation, and β is a hyperparameter balancing the two terms.

The total loss function $\mathcal{L}_{\text{total}}$ combines these components as follows:

$$\mathcal{L}_{\text{total}} := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{quant}} \tag{13}$$

By jointly minimizing this loss, the RQ-VAE optimizes the encoder, decoder, and the codebook vectors, ensuring that the model learns a compact and accurate representation of the input data, suitable for generating Semantic IDs.

4.2.2 Code Prediction for Sequential Recommendation

In the task of sequential recommendation, the model is trained to predict the next item's Semantic ID based on the sequence of previously interacted items. The process involves the following:

- **Input Representation**: The interaction history of a user is represented as a sequence of Semantic IDs. Each ID in the sequence corresponds to an item that the user has interacted with, providing a rich context for the next item prediction. Given a fixed number of codewords $m = 4$ and a codebook size of 256, each Semantic ID initially consists of three codewords. To handle collisions, a fourth codeword is added to differentiate duplicates.

- **Tokenization and Vocabulary Expansion**: To apply autoregressive generation for Semantic ID prediction, we extend the existing tokenizer vocabulary to include the new tokens representing the codewords. Each Semantic ID is formatted as a string such as `<a_i><b_j><c_k><d_l><e_m>` , where `a`, `b`, `c`, `d`, and `e` denote the order in the codebook. During training, the tokenizer converts the sequence of Semantic IDs into these tokens. For example, the model is provided with a context in a T5 manner, such as `"Predict Next Item:{history}"`, where `{history}` represents the sequence of previously interacted items.

- **Model Prediction**: Using a seq2seq architecture, the model autoregressively predicts the tokens of the Semantic ID for the next item. Autoregressive generation in language models involves generating each token step-by-step, conditioned on the previously generated tokens. In our setup, the model predicts the next token based on the provided instruction and the given context. Formally, given a sequence of previously interacted items represented by their Semantic IDs, $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t$, the probability of the next Semantic ID $\mathbf{s}_{t+1}$ is modeled as:

$$P(\mathbf{s}_{t+1}|\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t) = \prod_{i=1}^{m} P(c_{t+1,i}|c_{t+1,1}, c_{t+1,2}, \ldots, c_{t+1,i-1}, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t)$$

  where $c_{t+1,i}$ denotes the i-th codeword of the Semantic ID $\mathbf{s}_{t+1}$. The Semantic ID is broken down into tokens by the tokenizer, and the prediction starts with the most significant codeword (at the highest level of quantization) and progresses to the least significant codeword, capturing the hierarchy in item representation and ensuring that each prediction is informed by the entire context of previous interactions.

- **Generative Capability**: The generative nature of the model allows it to produce Semantic IDs even for items that may not be explicitly present in the training data. This capability is particularly useful for cold-start scenarios, where the model must recommend items with limited interaction history. The ability to generate novel Semantic IDs enables the system to adapt to new items and evolving user preferences effectively.

This detailed process of Semantic ID generation and prediction using RQ-VAE, combined with the autoregressive capabilities of seq2seq models, enables our system to efficiently and accurately handle various recommendation and retrieval tasks.

Table 1: Some item descriptions for products with the same first three semantic codes a_236, b_109, and c_124

Semantic ID	Item description
<a_236><b_109><c_124><d_8><e_0>	Ms. Pac-Man: Maze Madness Exciting 2D action as you navigate more than 100 game mazes, Chomp the pellets and avoid the ghosts, while fighting new...
<a_236><b_109><c_124><d_99><e_0>	Ms. Pac-Man Maze Madness Guide the rotund heroine through more than 180 different mazes. Video Games, Legacy Systems, Sega Systems, Sega Dreamcast...
<a_236><b_109><c_124><d_8><e_1>	Ms. Pac-Man Video Games, Legacy Systems, Nintendo Systems, Super Nintendo, Games. Ms. Pac-Man is here to drive you totally wild and crazy with more...
<a_236><b_109><c_124><d_153><e_0>	Ms. Pac-Man Video Games, Legacy Systems, Atari Systems, Atari 2600, Games. By now you've probably spent some time playing the master of mazes – PAC- ...

4.3 Training and Implementation details

4.3.1 Models details

RQ-VAE Implementation Details. We use item's content features such as title, price, brand, and category, which is then passed to the pre-trained Sentence-T5 model to obtain the item's semantic embedding of 768 dimension. The embedding is then quantized using RQ-VAE. The encoder and the decoder of the RQ-VAE have five intermediate layers of size 2048, 1024, 512, 256, 128 and 64 with ReLU activation, with a final latent representation dimension of 32. To quantize this representation, four levels of residual quantization is done. For each level, a codebook of cardinality 256 is maintained, where each vector in the codebook has a dimension of 32. When computing the total loss, we use $\beta = 0.25$ for Equation 12. The RQ-VAE model is trained for 5000 epochs. We use the AdamW optimizer with a learning rate of 1e-3, incorporating L2 regularization with a weight decay of 1e-4, and a batch size of 1024. After training, the learned encoder and the quantization component are used to generate a 4-tuple Semantic ID for each item. To prevent multiple items from being mapped to the same Semantic ID, a unique fourth code is added for items that share the same first three codewords. For example, two items associated with the tuple (12, 98, 4, 123) are assigned (12, 98, 4, 123, 0) and (12, 98, 4, 123, 1) respectively. If there are no collisions, 0 is assigned as the fourth codeword by default. This process results in a unique Semantic ID of length 5 for each item in the recommendation corpus. In table 1, we present examples of item descriptions that share the same first three semantic codes.

T5 Implementation Details. We use a pre-trained T5-small model implemented with the Transformer library to build our transformer-based encoder-decoder architecture. To enable the model to process input for the sequential recommendation task, we extend the vocabulary of the pre-trained sequence-to-sequence model by adding tokens for each semantic codeword. Specifically, the vocabulary is expanded to include 1280 (256×5) tokens representing items in the corpus. Additionally, we incorporate user-specific tokens into the pre-trained tokenizer's vocabulary.

We trained this model for 250k steps for the "Video Games" dataset. We use a batch size of 32. The learning rate is 1e-3 with a cosine scheduler with 1% of warmup. To conduct our experiments, we used either 2 V100 16GB GPUs. The estimated duration of training is 2 days.

4.3.2 Datasets details

For our experiments, we focus on a single domain, Video Games (Games), as shown in Table 2. We use each user's most recent 50 interactions as their historical interaction sequence. The dataset is split into training, validation, and test sets based on absolute timestamps. Specifically, instead of predicting the latest interactions for each user, we split the reviews chronologically using two timestamps, t_1 and t_2, to divide the data in an 8:1:1 ratio. The training set contains reviews within the time range $[0, t_1]$, the validation set is in $]t_1, t_2]$, and the test set covers the period $]t_2, \infty[$. This split is applied consistently for both pretraining and all downstream evaluation tasks. The models are evaluated on the test set using the version that achieves the lowest training loss on the validation set.

Table 2: : Statistics of the processed datasets in the sequential recommendation task for the domain "Video Games" from Amazon Reviews 2023. "Avg. Hist. Length" denotes the average length of interaction sequences. "Avg. Metadata Length" denotes the average number of character in the item metadata.

Dataset	# Users	# Items	# Interactions	Avg. Hist. Length	Avg. Metadata Length
Games	743,796	115,813	2,532,640	4.47	1163.64

For our experiments, we focus on items within the Video Games (Games) domain. Our model is trained using the full Amazon C4 dataset and the ESCI training split.

Table 3: Statistics of the datasets for product search tasks after preprocessing. "Avg. $|q|$" denotes the average number of characters in the queries. "Avg. $|t_i|$" denotes the average number of characters in the item metadata.

Name	#Queries	#Items	Avg. $\lvert q\rvert$	Avg. $\lvert t_i\rvert$
ESCI	27,643	1,367,729	22.46	544.28
Amazon-C4	21,223	1,058,417	229.89	538.97

4.4 Results

4.4.1 Baselines

Baselines for Sequential Recommendation. For the model architecture, we focus on two primary categories: (1) ID-based methods, where each unique item ID is assigned a learnable embedding vector to represent the item. Examples of these methods include GRU4Rec (Hidasi et al., 2016) and SASRec (Kang and McAuley, 2018); and (2) Text-based methods, which generate item representations by encoding item metadata into text-based representations. Notable examples in this category include SASRec (Text) (Hou et al., 2022) and UniSRec (Hou et al., 2022). For text-based approaches, such as UniSRec, we can fix the model architecture and examine how different text encoders impact the final performance. We specifically evaluate the performance using RoBERTa (Liu et al., 2019) and supervised SimCSE (Gao et al., 2022) as the pre-trained language model (PLM) baselines. In our experiments, we employ the architecture of UniSRec but do not initialize the behavior encoder with pre-trained checkpoints.

4.4.2 Overall Performance

Results on sequential recommendation. The results show that our implementation achieves the best performance across all evaluation metrics, surpassing the TIGER model in Recall@10, NDCG@10, Recall@50, and NDCG@50. Although GenIR methods show promise and appear to outperform both

Text-based and ID-based approaches in this specific evaluation, there is a need for additional experiments on diverse datasets to confirm the true superiority of these models.

Model	Games			
	R@10	N@10	R@50	N@50
ID-based methods				
GRU4Rec	2.19	1.19	5.21	1.84
SASRec	2.23	1.24	4.92	1.83
Text-based methods				
SASRec (Text)				
RoBERTa$_{\text{BASE}}$ (123M)	1.67	0.92	4.09	1.45
UniSRec (Text)				
RoBERTa$_{\text{BASE}}$ (123M)	2.22	1.22	5.47	1.92
BLAIR-MLM$_{\text{base}}$ (123M)	2.49	1.40	5.90	2.14
BLAIR$_{\text{base}}$ (123M)	2.72	1.46	6.43	2.27
SASRec (Text)				
RoBERTa$_{\text{LARGE}}$ (354M)	1.56	0.86	3.88	1.37
UniSRec (Text)				
RoBERTa$_{\text{LARGE}}$ (354M)	1.95	1.07	5.17	1.77
SimCSE$_{\text{LARGE}}$ (354M)	2.73	1.48	6.40	2.27
BLAIR-MLM$_{\text{large}}$ (354M)	2.51	1.39	6.18	2.18
BLAIR$_{\text{large}}$ (354M)	2.82	1.52	6.54	2.32
GenIR methods				
TIGER	3.77	2.02	9.13	3.19
Ours	**3.83**	**2.04**	**9.47**	**3.27**
Improvement	+1.6%	+1%	+3.7%	+2.5%

Table 4: Performance comparison of different methods on the sequential recommendation task. The best performance score is denoted in **bold**. The numbers reported are Recall@10, NDCG@10, Recall@50 and NDCG@50.

Results on product search. Due to time constraints, while we obtained results for the product search component, we were unable to perform a fair and thorough comparison. Although the sequential recommendation task was comprehensively tested and analyzed, the product search functionality, which involves retrieving relevant items based on user queries, could not be fully evaluated against appropriate baselines. Furthermore, the original paper only assessed this component in a zero-shot setting, leaving us without a fair baseline for comparison. In future work, we plan to introduce suitable baselines and conduct a more comprehensive evaluation, including comparisons with our results, to better validate the effectiveness of the product search within the generative framework.

4.4.3 Ablation Studies

Choice of codebooks. The design of DocIDs is critical, and the selection of the number and size of codebooks significantly impacts both performance and uniqueness. We conducted ablation studies with various configurations to evaluate their influence on the overall design.

Model	Games			
	R@10	N@10	R@50	N@50
Ours	**11.60**	**6.14**	**23.97**	**8.79**

Table 5: Performance comparison of different methods on the product search task. The best performance score is denoted in **bold**. The numbers reported are Recall@10, NDCG@10, Recall@50 and NDCG@50.

Table 6: Performance Metrics Table on TIGER model

Model	R@10	N@10	% of Unique Indices	Number of Max Conflicts
(3x256)	2.08	1.13	64.4%	68
(3x512)	3.59	1.90	79.1%	44
(4x256)	3.77	2.02	90.4%	32

Impact of the indexing tasks. In our experiments, we explored two different strategies for the indexing tasks to assess their impact on the model's performance. First, we implemented the item2index task alongside the sequential recommendation task, allowing the model to simultaneously learn to map items to indices and retrieve items from indices. Second, we tested the impact of training with only the item2index task, focusing solely on the mapping process without the retrieval component. Finally, we experimented with a hybrid approach where the model was trained with item2index for a fixed number of steps before transitioning to the sequential recommendation task exclusively. These variations helped us understand the contribution of each component to the overall performance, revealing insights into how the balance between mapping and retrieval tasks affects the model's ability to generate unique and effective indices while maintaining high recommendation accuracy.

Table 7: Performance Metrics Table

Model	R@10	N@10	R@50	N@50
reco only	3.52	1.86	8.96	3.04
item2index hybrid	3.63	1.93	9.03	3.10
item2index only	3.82	2.05	9.45	3.27
all tasks	3.83	2.04	9.47	3.27

5 Conclusions

While this research advances the integration of RS and Product Search through a unified GenIR framework, it also highlights the complexities and challenges inherent in such an approach. The use of generative models like T5 to directly generate document and item identifiers offers a promising simplification of the retrieval process. However, the absence of fair baselines for product search, due to the original paper's (Hou et al., 2024a) focus on zero-shot evaluations, limits the current assessment. Future work will need to introduce appropriate baselines and thoroughly compare results to fully validate this approach.

For the recommendation task, our evaluation on a single domain is a recognized limitation, and we plan to address this by incorporating additional datasets to better assess the model's generalizability. Furthermore, the computational overhead associated with generative models remains a significant challenge, indicating the need for ongoing research to optimize these methods for practical deployment.

Future Works. A key area for future research is investigating potential errors in the SemanticID generation process. While we currently monitor the percentage of unique SemanticIDs, it would be

valuable to develop a dedicated metric to assess the quality of these identifiers. A metric similar to COMET (Rei et al., 2020) in machine translation could provide insights into partial matches and error analysis for SemanticID predictions. Additionally, although Rajput et al. (2023) paper addresses the cold-start item problem, further exploration is needed to improve the model's handling of newly introduced items. Since our approach does not directly rely on collaborative filtering (CF), it is crucial to study how the model performs in cold-start scenarios for new users, where user data is sparse or entirely absent. This will ensure the robustness and adaptability of the system when dealing with unfamiliar users, enhancing its real-world applicability.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2018. URL https://arxiv.org/abs/1611.09268.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23. ACM, September 2023. doi: 10.1145/3604915.3608857. URL http://dx.doi.org/10.1145/3604915.3608857.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. *Proceedings of the 22nd international conference on Machine learning*, 2005. URL https://api.semanticscholar.org/CorpusID:11168734.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval, 2021. URL https://arxiv.org/abs/2010.00904.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22. ACM, July 2022a. doi: 10.1145/3477495.3531827. URL http://dx.doi.org/10.1145/3477495.3531827.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22. ACM, October 2022b. doi: 10.1145/3511808.3557271. URL http://dx.doi.org/10.1145/3511808.3557271.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 306–315, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3614821. URL https://doi.org/10.1145/3583780.3614821.

David R. Cheriton. From doc2query to doctttttquery, 2019. URL https://api.semanticscholar.org/CorpusID:208612557.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the trec 2019 deep learning track, 2020. URL https://arxiv.org/abs/2003.07820.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track, 2021. URL https://arxiv.org/abs/2102.07662.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018. URL https://api.semanticscholar.org/CorpusID:33169397.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings, 2022. URL https://arxiv.org/abs/2104.08821.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5), 2023. URL https://arxiv.org/abs/2203.13366.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM'16. ACM, October 2016. doi: 10.1145/2983323.2983769. URL http://dx.doi.org/10.1145/2983323.2983769.

Jiafeng Guo, Changjiang Zhou, Ruqing Zhang, Jiangui Chen, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Corpusbrain++: A continual generative pre-training framework for knowledge-intensive language tasks, 2024. URL https://arxiv.org/abs/2402.16767.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16. International World Wide Web Conferences Steering Committee, April 2016. doi: 10.1145/2872427.2883037. URL http://dx.doi.org/10.1145/2872427.2883037.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06939.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems, 2022. URL https://arxiv.org/abs/2206.05941.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024a.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems, 2024b. URL https://arxiv.org/abs/2305.08845.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data, October 2013. URL https://www.microsoft.com/en-us/research/publication/learning-deep-structured-semantic-models-for-web-search-using-clickthrough-data/.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

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

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation, 2018. URL https://arxiv.org/abs/1808.09781.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020. URL https://arxiv.org/abs/2004.04906.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020. URL https://arxiv.org/abs/2004.12832.

Yoon Kim. Convolutional neural networks for sentence classification, 2014. URL https://arxiv.org/abs/1408.5882.

Tzu-Lin Kuo, Tzu-Wei Chiu, Tzung-Sheng Lin, Sheng-Yang Wu, Chao-Wei Huang, and Yun-Nung Chen. A survey of generative information retrieval, 2024. URL https://arxiv.org/abs/2406.01197.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015. doi: 10.1609/aaai.v29i1.9513. URL https://ojs.aaai.org/index.php/AAAI/article/view/9513.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. Glen: Generative retrieval via lexical index learning, 2023. URL https://arxiv.org/abs/2311.03057.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

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, 2023a. URL https://arxiv.org/abs/2305.13731.

Xiaoxi Li, Zhicheng Dou, Yujia Zhou, and Fangchao Liu. Corpuslm: Towards a unified language model on corpus for knowledge-intensive tasks, 2024a. URL https://arxiv.org/abs/2402.01176.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Learning to rank in generative retrieval, 2023b. URL https://arxiv.org/abs/2306.15222.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Multiview identifiers enhanced generative retrieval, 2023c. URL https://arxiv.org/abs/2305.16675.

Yongqi Li, Zhen Zhang, Wenjie Wang, Liqiang Nie, Wenjie Li, and Tat-Seng Chua. Distillation enhanced generative retrieval, 2024b. URL https://arxiv.org/abs/2402.10769.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8a0e1141fd37fa5b98d5bb769ba1a7cc-Paper.pdf.

Julieta Martinez, Holger H. Hoos, and James J. Little. Stacked quantizers for compositional vector compression, 2014. URL https://arxiv.org/abs/1411.2173.

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, 2023. URL https://arxiv.org/abs/2212.09744.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10350. URL https://ojs.aaai.org/index.php/AAAI/article/view/10350.

Usama Nadeem, Noah Ziems, and Shaoen Wu. Codedsi: Differentiable code search, 2022. URL https://arxiv.org/abs/2210.00328.

Thong Nguyen and Andrew Yates. Generative retrieval as dense retrieval, 2023. URL https://arxiv.org/abs/2306.11397.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models, 2021. URL https://arxiv.org/abs/2108.08877.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020. URL https://arxiv.org/abs/1901.04085.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction, 2019. URL https://arxiv.org/abs/1904.08375.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks, 2021. URL https://arxiv.org/abs/2009.02252.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

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, 2023. URL https://arxiv.org/abs/1910.10683.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval, 2023. URL https://arxiv.org/abs/2305.05065.

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. Shopping queries dataset: A large-scale esci benchmark for improving product search, 2022. URL https://arxiv.org/abs/2206.06588.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. Comet: A neural framework for mt evaluation, 2020. URL https://arxiv.org/abs/2009.09025.

Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. Tome: A two-stage approach for model-based retrieval, 2023. URL https://arxiv.org/abs/2305.11161.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009. URL https://api.semanticscholar.org/CorpusID:207178704.

Gerard Salton and Michael McGill. Introduction to modern information retrieval. *mcgraw-hill*, 1983. URL https://api.semanticscholar.org/CorpusID:43685115.

Claude Sammut and Geoffrey I. Webb, editors. *TF–IDF*, pages 986–987. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_832. URL https://doi.org/10.1007/978-0-387-30164-8_832.

Zihua Si, Zhongxiang Sun, Jiale Chen, Guozhang Chen, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, Jun Xu, and Kun Gai. Generative retrieval with semantic tree-structured item identifiers via contrastive learning, 2024. URL https://arxiv.org/abs/2309.13375.

EuiYul Song, Sangryul Kim, Haeju Lee, Joonkee Kim, and James Thorne. Re3val: Reinforced and reranked generative retrieval, 2024. URL https://arxiv.org/abs/2401.16979.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, 2019. URL https://arxiv.org/abs/1904.06690.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. Learning to tokenize for generative retrieval, 2023. URL https://arxiv.org/abs/2304.04171.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. Idgenrec: Llm-recsys alignment with textual id learning, 2024. URL https://arxiv.org/abs/2403.19021.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding, 2018. URL https://arxiv.org/abs/1809.07426.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. Semantic-enhanced differentiable search index inspired by learning strategies, 2023. URL https://arxiv.org/abs/2305.15115.

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, 2022. URL https://arxiv.org/abs/2202.06991.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and et al. Shruti Bhosale. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Ye Wang, Rui Xie, Wenxin Hu, Wei Ye, and Shikun Zhang. Generative retrieval with large language models, 2024a. URL https://openreview.net/forum?id=ZdjKRbtrth.

Yidan Wang, Zhaochun Ren, Weiwei Sun, Jiyuan Yang, Zhixiang Liang, Xin Chen, Ruobing Xie, Su Yan, Xu Zhang, Pengjie Ren, Zhumin Chen, and Xin Xin. Enhanced generative recommendation via content and collaboration integration, 2024b. URL https://arxiv.org/abs/2403.18480.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. A neural corpus indexer for document retrieval, 2023a. URL https://arxiv.org/abs/2206.02743.

Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. Novo: Learnable and interpretable document identifiers for model-based ir. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 2656–2665, New York, NY, USA, 2023b. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3614993. URL https://doi.org/10.1145/3583780.3614993.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020. URL https://arxiv.org/abs/2007.00808.

Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. Auto search indexer for end-to-end document retrieval, 2023. URL https://arxiv.org/abs/2310.12455.

Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval, 2019. URL https://arxiv.org/abs/1903.10972.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271800. URL https://doi.org/10.1145/3269206.3271800.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec, 2021. URL https://arxiv.org/abs/2107.03312.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. Scalable and effective generative information retrieval, 2023. URL https://arxiv.org/abs/2311.09134.

Hansi Zeng, Chen Luo, and Hamed Zamani. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding, 2024. URL https://arxiv.org/abs/2404.14600.

Hailin Zhang, Yujing Wang, Qi Chen, Ruiheng Chang, Ting Zhang, Ziming Miao, Yingyan Hou, Yang Ding, Xupeng Miao, Haonan Wang, Bochen Pang, Yuefeng Zhan, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Xing Xie, Mao Yang, and Bin Cui. Model-enhanced vector index, 2023a. URL https://arxiv.org/abs/2309.13335.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, and Zhao Cao. Term-sets can be strong document identifiers for auto-regressive search engines. *ArXiv*, abs/2305.13859, 2023b. URL https://api.semanticscholar.org/CorpusID:270712186.

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, 2024. URL https://arxiv.org/abs/2311.09049.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, and Ji-Rong Wen. Dynamicretriever: A pre-training model-based ir system with neither sparse nor dense index, 2022a. URL https://arxiv.org/abs/2203.00537.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. Ultron: An ultimate retriever on corpus with a model-based indexer, 2022b. URL https://arxiv.org/abs/2208.09257.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation, 2023. URL https://arxiv.org/abs/2206.10128.

Noah Ziems, Wenhao Yu, Zhihan Zhang, and Meng Jiang. Large language models are built-in autoregressive search engines, 2023. URL https://arxiv.org/abs/2305.09612.