ROGER: Ranking-oriented Generative Retrieval

YUJIA ZHOU, School of Information, Renmin University of China, China
JING YAO, Microsoft Research Asia, China
ZHICHENG DOU*, Gaoling School of Artificial Intelligence, Renmin University of China, China
YITENG TU, School of Information, Renmin University of China, China
LEDELL WU, Beijing Academy of Artificial Intelligence, China
TAT-SENG CHUA, National University of Singapore, Singapore
JI-RONG WEN, Gaoling School of Artificial Intelligence, Renmin University of China, China

In recent years, various dense retrieval methods have been developed to improve the performance of search engines with a vectorized index. However, these approaches require a large pre-computed index and have limited capacity to memorize all semantics in a document within a single vector. To address these issues, researchers have explored end-to-end generative retrieval models that use a seq-to-seq generative model to directly return identifiers of relevant documents. Although these models have been effective, they are often trained with the maximum likelihood estimation method. It only encourages the model to assign a high probability to the relevant document identifier, ignoring the relevance comparisons of other documents. This may lead to performance degradation in ranking tasks, where the core is to compare the relevance between documents. To address this issue, we propose a ranking-oriented generative retrieval model that incorporates relevance signals in order to better estimate the relative relevance of different documents in ranking tasks. Based upon the analysis of the optimization objectives of dense retrieval and generative retrieval, we propose utilizing dense retrieval to provide relevance feedback for generative retrieval. Under an alternate training framework, the generative retrieval model gradually acquires higher-quality ranking signals to optimize the model. Experimental results show that our approach increasing Recall@1 by 12.9% with respect to the baselines on MS MARCO dataset.

CCS Concepts: • **Information systems → Retrieval models and ranking**.

Additional Key Words and Phrases: Model-based IR, Generative Model, Document Retrieval, Knowledge Distillation, Docid Representation

1 INTRODUCTION

A search engine is a valuable tool for meeting people's everyday information needs. The process of searching begins with a query, which prompts the engine to retrieve a list of candidate documents from a large collection. These candidates are then re-ranked to create a final list of results. The performance of the initial retrieval

*Corresponding author: Zhicheng Dou

Authors' addresses: Yujia Zhou, zhouyujia@ruc.edu.cn, School of Information, Renmin University of China, Beijing, China; Jing Yao, jingyao@microsoft.com, Microsoft Research Asia, Beijing, China; Zhicheng Dou, dou@ruc.edu.cn, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; Yiteng Tu, yitengtu16@gmail.com, School of Information, Renmin University of China, Beijing, China; Ledell Wu, wuyu@baai.ac.cn, Beijing Academy of Artificial Intelligence, Beijing, China; Tat-Seng Chua, dcscts@nus.edu.sg, National University of Singapore, Singapore; Ji-Rong Wen, jrwen@ruc.edu.cn, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China.

ACM 1558-2868/2024/6-ART
https://doi.org/10.1145/3603167

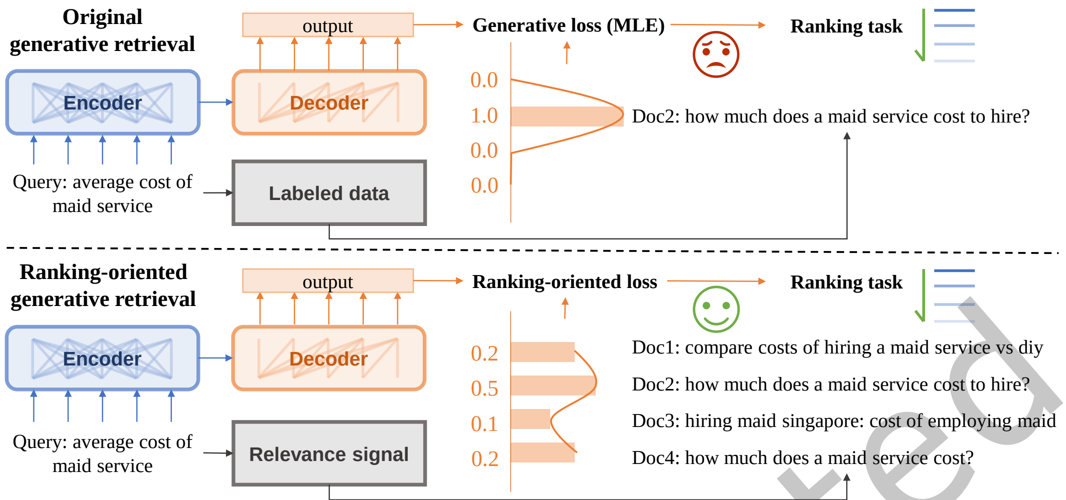

Fig. 1. Comparison of original and ranking-oriented generative retrieval. Original method trained with MLE assumes a one-point deterministic distribution, while our method additionally considers the graded semantic relevance of irrelevant documents to better adapt the model to ranking tasks.

stage is crucial to the overall quality of the search. The traditional method [1, 12, 13, 17, 42] for document retrieval involves the use of an inverted index, which has been the foundation of term-based parsing for decades. However, this method can run into issues with vocabulary mismatches [18], where different terms are used to describe the same concept, leading to incomplete or inaccurate search results. In recent years, new dense retrieval methods [22, 23, 25, 50, 52] have been developed to address this problem. These methods convert the semantic information in both queries and documents into dense vectors, and then use these vectors to retrieve relevant documents.

Although dense retrieval has been shown to be effective in practical applications, the index-based framework requires a large index to be pre-computed on the entire corpus to support document retrieval. In addition, a single vector has limited capacity to retain all semantics within a document. To make full use of the capabilities of deep neural networks, several studies have explored end-to-end generative retrieval models [2, 46, 47] that directly generate the relevant document identifier (docid). These models substitute the traditional explicit index with a dynamic neural search index [46], thereby enabling end-to-end document retrieval through the utilization of a seq-to-seq generative model.

Despite the significant advancements in generative retrieval models, they are commonly trained with maximum likelihood estimation (MLE), which maximizes the predictive probability of the ground-truth docid comprised of pre-defined tokens. However, as shown in Figure 1, such an optimization objective only encourages the model to assign high probability to the ground-truth docid (one-point distribution), and is agnostic about the relevance comparisons among non-golden docids [30, 44]. We argue that this objective is biased against the goal of ranking tasks, where the estimation of graded document relevance in relation to a specific query is of paramount importance. In light of this, we suggest building a ranking-oriented generative retrieval model through the incorporation of relevance signals into the generation process.

The relative comparisons of semantic relevance are performed well in dense retrieval models due to their utilization of ranking-based or contrastive optimization objectives [7, 22, 50]. Upon conducting an analysis of both generative retrieval and dense retrieval (as detailed in Sec. 4), it has been observed that the optimization objectives employed in these models are inconsistent, leading to variations in the distribution of ranking results. Specifically, generative retrieval models tend to perform more favorably at the first rank position, whereas dense retrieval models tend to exhibit a more balanced performance across all rank positions. This observation inspires the notion that these two retrieval paradigms can be effectively utilized in a complementary manner, with the dense retrieval model providing the necessary relevance feedback that is missing in generative retrieval models.

Due to the heterogeneity of the two retrieval paradigms, generative retrieval models face the difficulty in leveraging the relevance signals from dense retrieval models directly, reflected in two aspects. (1) Data mismatch. They are trained on different data (query-docid pairs vs. query-document pairs), which leads to different document knowledge learned by the two models. (2) Model mismatch. Different model structures (encoder-decoder structure vs. dual encoder structure) cause different abilities to fit the training data. In order to mitigate the noise introduced by these inconsistencies, an intermediate model, named dense docid retriever, is introduced to link the two retrieval paradigms. As a bridge, it could distill knowledge from the dense document retrieval model through a consistent model structure, and provide relevance feedback to the generative retrieval model within the same document representation space.

In this paper, we introduce a **R**anking-**O**riented **GE**nerative **R**etrieval model (ROGER), which aims to enhance the ability of generative retrieval to model relative semantic relevance for ranking tasks. On the basis of the original generative retriever, two dense retrievers (i.e. a dense docid retriever and a dense document retriever) are incorporated during the training stage. Concretely, the generative retriever receives the relevance feedback from the dense docid retriever to train the generator to complement the MLE objective. Additionally, to enhance the quality of relevance feedback, the dense docid retriever is further optimized through the utilization of knowledge distillation from the dense document retriever. During the inference phase, when presented with a query, the optimized generative retriever is utilized to generate a ranking list through constrained beam search.

We conduct experiments on three widely adopted document retrieval datasets: MS MARCO, Natural Questions, and ClueWeb. Experimental results indicate that our ranking-oriented training strategy is applicable to a variety of generative retrieval methods and achieves significant improvements over them. Moreover, auxiliary experiments show that our model can combine the respective advantages of generative retrieval and dense retrieval to improve the MRR and Recall metrics.

The paper makes several key contributions, which can be summarized as follows:

- We thoroughly examine the similarities and distinctions between generative retrieval and dense retrieval methods. By analyzing the strengths and weaknesses of each approach, we propose utilizing dense retrieval as a means of providing relevance feedback for generative retrieval. This integration of the two paradigms aims to leverage the benefits of both methods and enhance the generative retrieval process.
- We introduce a novel approach for generative retrieval that incorporates relevance signals to perform relative comparisons of different documents in ranking tasks. This ranking-oriented generative retrieval model aims to improve the model's ability to handle ranking tasks and enhance its overall performance.
- We present an alternate training framework that establishes a connection between the generative retrieval model and the dense retrieval model. This framework continuously sends ranking signals from the dense retrieval model to the generative retrieval model, enabling the generative model to learn from the ranking feedback and improve its ranking capabilities.
- We provide experimental results that demonstrate the effectiveness of the proposed model. The model outperforms dense and generative retrieval baselines significantly, achieving a 12.9% higher Recall@1 on the MS MARCO dataset. This improvement demonstrates the model's ability to effectively combine the advantages of generative and dense retrieval methods, thereby enhancing overall retrieval performance.

2 RELATED WORK

2.1 Sparse Retrieval

Sparse retrieval methods are widely employed in practical applications due to their efficiency and effectiveness in information retrieval systems. These methods leverage inverted indexes and incorporate various techniques to evaluate term importance and compute matching scores between queries and documents. Notable examples of such techniques include the frequency-based BM25 model [42] and graph-based approaches [3, 43] that utilize algorithms like PageRank. In recent years, representation learning [34, 40] has gained prominence and has been extensively applied to automatically derive term weights from word embeddings [14, 20, 55]. Additionally, contextualized text representation techniques have emerged, enabling the prediction of term importance in methods like DeepCT [12] and HDCT [13]. These approaches leverage advanced models that can capture the contextual meaning of terms within documents and queries. Furthermore, specific sparse representation learning methods focused on terms have been introduced, such as SparTerm [1] and SPLADE [16, 17], aiming to enhance text retrieval speed. These methods utilize term-based approaches to construct compact representations for efficient retrieval while preserving retrieval effectiveness. However, a persistent challenge in sparse retrieval methods is the issue of word mismatch, where the vocabulary used in queries may not precisely match the vocabulary found in the document collection. To address this challenge, researchers have proposed techniques to expand the set of possible terms for queries or candidate documents, thereby improving retrieval performance. For instance, some methods employ query expansion [39] to augment the original query with additional terms that are likely to improve retrieval accuracy. Similarly, document expansion [38] techniques can be employed, where potential terms are added to the candidate documents to increase the chances of retrieving relevant information. By incorporating word expansion techniques into sparse retrieval approaches, the retrieval system becomes more flexible in dealing with vocabulary discrepancies and can provide more accurate results. These expansion methods effectively increase the coverage of queries and documents, enabling the retrieval system to better capture the underlying semantics and bridge the lexical gaps between the query and the document collection.

2.2 Dense Retrieval

Deep learning methods have revolutionized the field of information retrieval by enabling the identification of semantic similarity between queries and documents. Unlike traditional methods that rely solely on word matching, these deep learning techniques take advantage of neural networks to convert queries and documents into vector representations, capturing the underlying meaning and context of the text. By representing text as vectors, it becomes possible to calculate the similarity between queries and documents using various distance metrics, such as cosine similarity or Euclidean distance. This allows for more accurate retrieval of relevant documents, even in cases where the vocabulary used in the query and the document may not exactly match.

To improve the efficiency of vector-based search, several algorithms have been developed. One popular approach is Approximate Nearest Neighbor Search (ANNS), which aims to find approximate nearest neighbors instead of exact matches. ANNS algorithms, such as the HNSW index [31] and SPTAG [10], provide efficient indexing and retrieval of vectors, significantly reducing the search time. Another technique used to enhance efficiency is Product Quantization (PQ), which compresses the vector representations by dividing them into subvectors and quantizing each subvector separately. PQ-based methods, such as OPQ [19] and JPQ [51], allow for faster computation and storage of large-scale vector indexes.

A common approach for dense retrieval is the dual encoder architecture. In this setup, separate encoders are used to encode queries and documents into vector representations. This approach has shown promising results in various tasks, such as document ranking and question answering. Pre-trained language models like BERT [15] and T5 [41] have been widely adopted in dual encoder models, providing high-quality representation vectors and improving retrieval performance. To further enhance performance, methods based on hard negative sampling have been developed. Hard negative sampling involves selecting negative examples that are challenging to distinguish from positive examples, thereby forcing the model to learn more discriminative representations. Models such as DPR [22] and ANCE [50] leverage hard negative sampling to improve retrieval accuracy and ranking quality. Moreover, there have been advancements in knowledge distillation techniques aimed at enhancing model training with more pertinent signals. For instance, models such as those described by [21] and [49] advocate for the training of dense retrieval models under the guidance of a teacher model. This method is designed to yield a retrieval system that is both effective and efficient.

2.3 Generative Retrieval

Generative models have gained significant popularity in the field of information retrieval (IR) due to their ability to generate relevant answers and retrieve information from large document collections. Initially, generative models were trained to directly produce answers or entity names for specific IR tasks [6, 37]. Recent advancements have focused on incorporating document identifiers into the generative models to enhance their retrieval capabilities. One approach, proposed in the work of Metzler et al. [33], introduced a model-based IR framework that embeds document identifiers into the model. This approach represents document identifiers using structured semantic clusters and treats the transformer memory as a neural search index. By incorporating document identifiers, the model can effectively retrieve relevant documents based on their unique identifiers.

Following this line of research, several other models have been developed, such as DSI [46, 60], NCI [47], and DynamicRetriever [57]. These models further enhance the incorporation of document identifiers into the generative retrieval process. They represent the document identifiers with the structured semantic clusters and regard the transformer memory as a neural search index. Zhou et al. [58] tried keyword-based and PQ-based identifiers and applied a three-stage training pipeline for model training. Additionally, researchers have extended the model-based IR framework to knowledge-intensive language tasks, resulting in improved performance. Chen et al. [8] and Bevilacqua et al. [2] explored the integration of generative models with knowledge to enhance question-answering and entity retrieval tasks. Similarly, Chen et al. [9] proposed a model that incorporates a knowledge corpus as a memory bank to facilitate knowledge-intensive language tasks. To cope with the challenge of updating generative models with new documents, Mehta et al. [32] proposed a model updating mechanism specifically designed for incorporating new documents into the generative retrieval model. This allows the model to adapt and incorporate the latest information while maintaining its retrieval capabilities. Additionally, a series of studies [27–29, 45, 48, 54, 56, 59] explored various methods for representing docids, aiming to enrich docid semantics with learnable parameters.

Despite these advancements, the existing models that associate queries and document identifiers with generative language models do not explicitly model the relative relevance between documents. To address this limitation, the focus of this paper is on developing a rank-oriented generative retrieval model. This model incorporates relevance feedback signals from dense retrieval models, which consider the relative relevance between documents. By leveraging the feedback signals, the rank-oriented generative retrieval model aims to improve the ranking and retrieval performance, providing more accurate and relevant results to users.

3 PRELIMINARIES

In this section, we formulate the document retrieval problem and introduce the technique of dense retrieval and generative retrieval. Document retrieval is a fundamental task in information retrieval, aiming to retrieve relevant documents from a large collection given a user's query. Dense retrieval is a technique employed in document retrieval that focuses on dense vector representations of documents and queries. Unlike traditional sparse vector representations, dense representations encode rich semantic information and capture the contextual meaning of the text. Dense retrieval models mainly depend on dual-encoder architectures to learn these dense

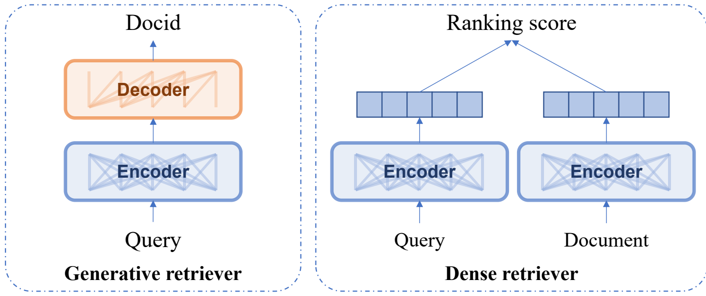

Fig. 2. The basic framework of generative retrieval and dense retrieval.

representations. Generative retrieval is another technique used in document retrieval that leverages generative models, such as T5 or BART, to perform retrieval. The detailed introduction is as follows.

3.1 Problem Statement

Given a query q and a corpus C, the target of document retrieval is to return top-k ranked documents from the corpus. To support ranking, the model should compute the relevance score between the query q and each document d in the corpus C, denoted as:

$$\text{score}(d, q) = f(d|q), \quad d \in C \tag{1}$$

In this equation, the function $f(d|q)$ represents the model used to compute the ranking score of document d given the query q. Here, d is an element of the set C, which contains all the documents to be ranked.

3.2 Dense Retrieval

Dense retrieval encodes the query and documents separately into embeddings, and uses the similarity function, such as inner product or cosine similarity, to compute the ranking scores. In this way, Eq. (1) can be written as:

$$\text{score}(d, q) = D_\phi(d|q) = \langle \vec{X}_{q;\phi}, \vec{X}_{d;\phi} \rangle, \tag{2}$$

where D_ϕ denotes the dense retriever with parameters ϕ, $\vec{X}$ represents embeddings encoded by the dense retriever (e.g. BERT Encoder), and $\langle , \rangle$ is the similarity function. Dense retrieval models are typically trained with ranking-based or contrastive losses. Each training sample consists of a query q, a positive document d^+, and a set of negative documents C^-. The dense loss function, such as the contrastive loss, can be formalized as:

$$\mathcal{L}_{\text{dense}} = -\log \frac{\exp\left(D_\phi(d^+|q)\right)}{\exp\left(D_\phi(d^+|q)\right) + \sum_{d^- \in C^-} \exp\left(D_\phi(d^-|q)\right)}. \tag{3}$$

3.3 Generative Retrieval

Generative retrieval aims to generate the relevant docids for a given query directly with a seq-to-seq model. The final ranking scores rely on the generation probability of docids. To represent docids, previous studies tried various strategies such as keyword-based docids [2, 9] or semantic-based docids [46, 47]. Each docid uniquely identifies a document in the corpus and can reflect its semantics.

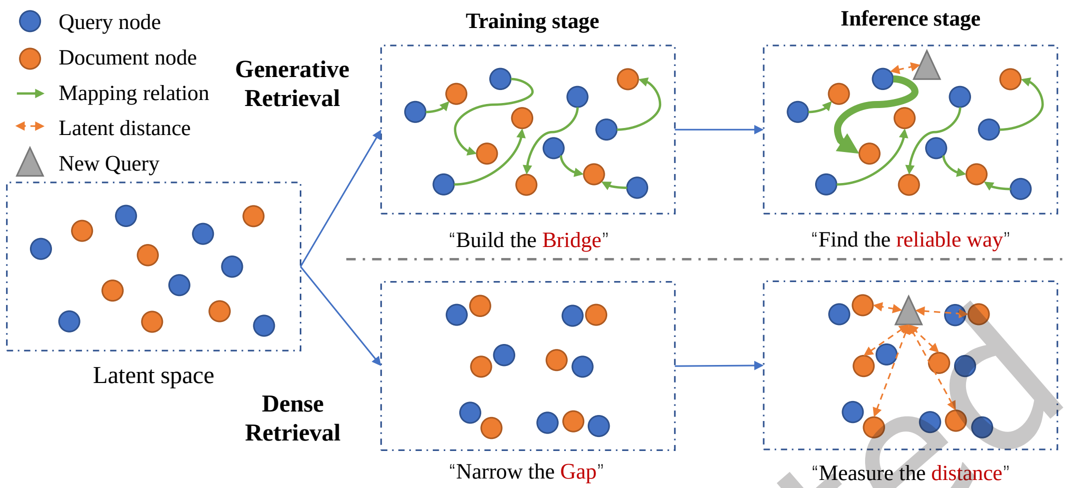

Fig. 3. Generative Retrieval vs. Dense Retrieval. Generative retrieval tends to build the bridges from the query space to document space, while dense retrieval prefers to encode the query and document into the same space and narrows their gaps. Given a query, generative retrieval tries to find a document along the existing bridge, and dense retrieval directly measures the query-document relevance within the semantic space.

Assuming that d' is the identifier of the document d, the generative model tries to predict relevant docids with the highest auto-regressive scores. Under this method, Eq. (1) can be written as:

$$\text{score}(d, q) = G_\theta(d'|q) = \prod_{i=1} G_\theta(d'_i|d'_{<i}, q), \tag{4}$$

where G_θ is the generative retriever, and d'_i is the i_{th} token of the docid, $d'_{<i}$ represent all tokens generated before the i_{th} token. The parameters θ are optimized with the standard seq-to-seq objective, i.e., maximizing the target sequence likelihood. The generative loss function, which is the cross-entropy loss, can be defined as:

$$\mathcal{L}_{\text{gen}} = -\sum_{i=1} \log G_\theta(d'_i|d'_{<i}, q), \tag{5}$$

where $G_\theta(d'_i|d'_{<i}, q)$ is the generation probability of token d'_i based on the given input. Such a generative loss only focuses on fitting the tokens of the relevant docid at each position of the decoder, without explicitly encoding the query and candidate documents to the same space for comparing representations.

4 GENERATIVE VS. DENSE RETRIEVAL

To clarify the distinction and connection between generative retrieval and dense retrieval, we initially explore their optimization goals before delving into an empirical comparison of their ranking results. Figure 3 visualizes the differences between these approaches. Imagine a scenario where various query and document nodes are dispersed throughout a space. In the case of generative retrieval, the process is akin to constructing bridges from queries directly to their corresponding documents, enhancing the model's ability to consistently identify and follow dependable routes to locate the desired document when querying. Conversely, dense retrieval operates by encoding both queries and documents into a unified space, thereby minimizing the distance between matching (positive) query-document pairs. This methodology simplifies the task of comparing the proximity of each document to a given query, facilitating more straightforward measurement of their relative distances.

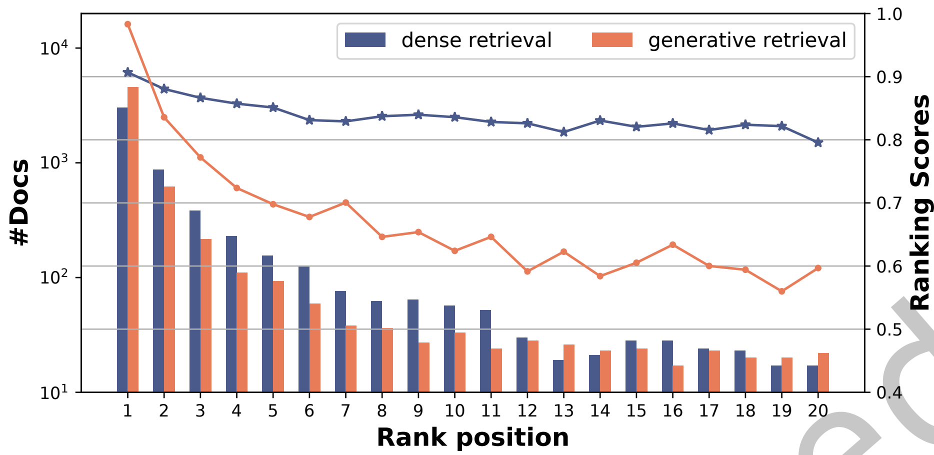

Fig. 4. Empirical study on ranking results of generative retrieval and dense retrieval.

4.1 Analysis of Optimization Objective

According to the loss functions of generative retrieval and dense retrieval in Eq. (3) and Eq. (5), their learning objectives can be formalized as updating parameters θ and ϕ to minimize the loss:

$$\theta^* = \arg\max_{\theta} \sum_i \log \underbrace{G_\theta(d'_i | d'_{<i}, q)}_{\text{q-d' relevance \& sequence dependencies}}, \tag{6}$$

$$\phi^* = \arg\max_{\phi} \left(D_\phi(d^+|q) - \log \sum_{d^- \in \widetilde{C}} \exp\left(D_\phi(d^-|q)\right) \right), \tag{7}$$

where $\widetilde{C}$ is the union of C^- and d^+. Then, we convert the model D_ϕ into the form of similarity function $\langle , \rangle$:

$$\phi^* = \arg\max_{\phi} (\underbrace{\langle \vec{X}_{q;\phi}, \vec{X}_{d^+;\phi} \rangle}_{\text{q-d relevance}} - \log \sum_{d^- \in \widetilde{C}} \exp(\underbrace{\langle \vec{X}_{q;\phi}, \vec{X}_{d^-;\phi} \rangle}_{\text{comparisons}})). \tag{8}$$

Finally, we can summarize two differences from the optimization objectives of generative retrieval and dense retrieval. (1) Data mismatch. In terms of relevance modeling, generative retrieval puts more emphasis on the relationship between query and docid, while dense retrieval concentrates on the query-document matching. (2) Model mismatch. Generative retrieval models the sequence dependencies between docid tokens, but dense retrieval focuses on capturing relative comparisons as ranking signals. Concretely, the generative retrieval approach concentrates on creating negative samples at the level of individual tokens, considering every possible combination of tokens that isn't the correct match as a negative example. On the other hand, dense retrieval methods prioritize the use of document-level negative samples, particularly emphasizing the importance of selecting high-quality negative instances that are closely related to the query. These two differences give each approach its own set of benefits and drawbacks. To further explore the impact of different optimization objectives on the ranking results, we perform an empirical study as follows.

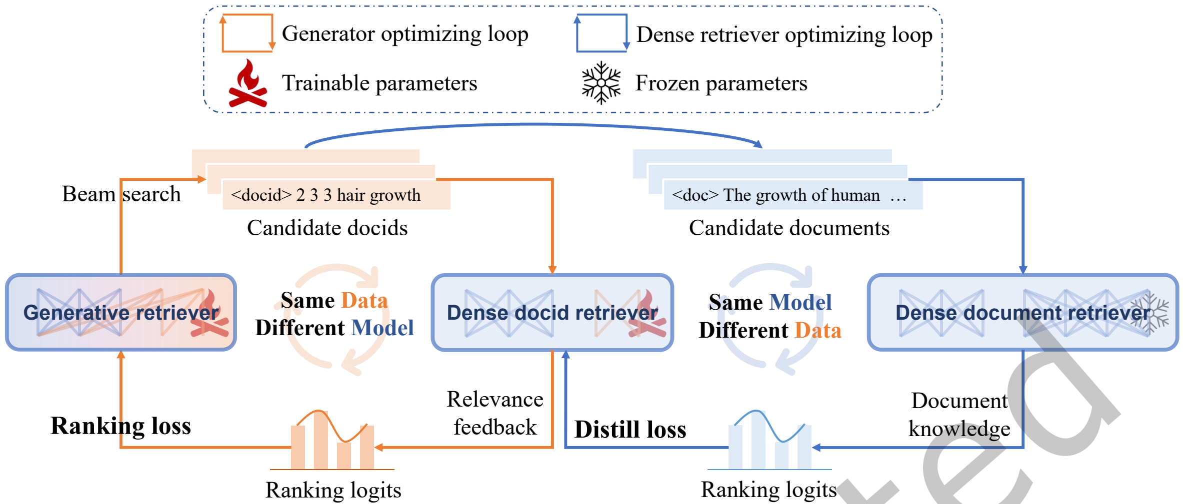

Fig. 5. The training framework of ranking-oriented generative retrieval. A dense docid retriever is built to bridge the gap between the generative retriever and dense document retriever. We perform two optimizing loops to further train the generator and the dense docid retriever, so as to inject ranking signals into the generative retriever.

4.2 Empirical Study on Ranking Results

Based on the above analysis, we realize that the generative and dense retrieval models have certain differences in learning objectives. Here, we empirically study whether these differences would be reflected in their ranking results.

Experimental Setup. We choose DPR [22] and Ultron [58] as the dense and generative retrieval model respectively, and investigate their ranking results on the development set of the Natural Question dataset [24]. Concretely, for each query, we retrieve the top 20 documents and count the number of relevant documents appearing in different rank positions. Additionally, we calculate the average ranking score of relevant documents at each position. To better show the tendency, we normalize generation probabilities and relevance matching scores to the same scale (0-1) with sigmoid function for comparison.

Observations. From Figure 4, we can observe that: (1) The generative retrieval model has a higher likelihood of placing the correct document at the top spot, while the dense retrieval model is relatively more balanced in every rank position. (2) The relevant documents at the first position are given very high scores by the generative model, and the scores for the following positions decrease rapidly. By contrast, the ranking score curve of dense retrieval is smoother. These two findings are consistent with our analysis of the optimization objectives of the two retrieval paradigms. As demonstrated in Equation (8), the optimization goal of the generative retrieval model is solely to give high probability to the correct document identifier. This approach presumes a distribution focused on a single point, resulting in both high accuracy and confidence at the first position. In contrast, the optimization objective of dense retrieval model (shown in Equation (9)) captures the nuances between relevant documents and hard negative documents, allowing it to measure the relative relevance comparisons between different documents. As a result, its curve is smoother and it performs fairly well at each position.

Overall, generative retrieval and dense retrieval learn associations between queries and documents through different optimization objectives, and show different distributions over the ranking results. This motivates us to add the learning of relevance comparisons to the optimization objective of the generative retrieval model, so as to enhance its performance on the ranking tasks.

5 RANKING-ORIENTED TRAINING METHOD

Based on the findings above, the generative retrieval model, which is trained with the standard MLE method, does not fully consider relative comparisons between irrelevant documents. On the basis of this, we propose a ranking-oriented training approach that leverages the relevance feedback from dense retrieval to improve the performance of original generative retrieval model.

The overall training framework is depicted in Figure 5. The first step is to link the generative retriever and the dense retriever with a dense docid retriever. Next, the generative retriever is optimized with relevance feedback from the dense docid retriever. Then, the dense docid retriever is updated with knowledge distillation from the dense document retriever. The details are given below.

5.1 Basic Settings

To start, we establish the foundational parameters for the generative retrieval model. Our ROGER framework, in theory, is adaptable to any generative retrieval models, regardless of the type of docids they utilize. Within the scope of this study, we examine two distinct models: NCI [47], which employs semantic cluster-based docids, and Ultron [58], which uses string-based docids. The T5-base [41] serves as the primary architecture for these models, but it's noteworthy that it can be substituted with other encoder-decoder models, such as BART [26], offering flexibility in the underlying technology.

5.2 Linking the Two Retrieval Paradigms

As we discussed in Sec. 4, there are two main differences in the optimization objectives of generative retrieval and dense retrieval: data mismatch and model mismatch. These inconsistencies can lead to noisy relevance feedback and hinder the training of the generative retrieval model. To cope with this problem, we propose to add an intermediate model to link the two paradigms. This model should have the same architecture as the dense retrieval model, but is trained on the same data as the generative retrieval model, namely the dense docid retriever. As a bridge, it can not only learn document knowledge from the dense document retriever, but also provide unbiased relevance feedback to the generative retriever.

Given the training data $\mathbb{D}$, which consists of a series of relevant query-document pairs, we represent documents as identifiers to construct a set of query-docid pairs $\mathbb{D}'$. Following previous methods [2, 46], identifiers can be represented as a sequence of semantic cluster ids or several keywords. Based on $\mathbb{D}$ and $\mathbb{D}'$, we first warm up the generative retriever and two dense retrievers with Eq. (5) and Eq. (3). Next, we attempt to inject the ranking signals hidden in the dense docid retriever into the generative retriever by alternately updating their parameters.

5.3 Optimizing the Generative Retriever with Relevance Feedback

To focus on the ranking task, we leverage the dense docid retriever to provide the relative relevance scores over docids. Since it is intractable to enumerate all the candidate docids, we only require the generative retriever to accurately measure the relevance of the most probable candidate docids. Concretely, as shown in Algorithm 1, for each query q, we generate a set of candidate docids C_q, which is its own beam search results.

Over the candidate docids, there are two types of relevance feedback that may be useful for ranking. (1) **Relevance to the query**: We compute the matching scores between the query q and docids in C_q as the feedback; (2) **Relevance to the ground-truth** : The matching scores between the ground-truth docid $\widehat{d'}$ and docids in C_q are fed to the generative retriever. The ranking logits of these two types of relevance feedback are denoted as:

$$\Phi_{q,d'} = \frac{\exp(D_\alpha(d'|q))}{\sum_{d'\in C_q}\exp(D_\alpha(d'|q))}, \Phi_{\widehat{d'},d'} = \frac{\exp(D_\alpha(d'|\widehat{d'}))}{\sum_{d'\in C_q}\exp(D_\alpha(d'|\widehat{d'}))},$$

Algorithm 1 Generative Language Model with Constrained Beam Search

Input: Language model LM, starting token S, maximum sequence length L, beam width B, constraints C, input query q, number of sequences n
Output: Generated sequences C_q
$Q \leftarrow \{[S], 1.0\}$ {Initialize beams with starting token and initial score}
$C_q \leftarrow []$ {Initialize empty generated sequences}
while C_q is not complete and sequence length $< L$ **do**
  $Q' \leftarrow []$ {Initialize empty set for next iteration}
  for all (S_i, P_i) in Q **do**
    $C_i \leftarrow$ Apply constraints to S_i
    $W_i \leftarrow$ Generate next words from LM given input query q and context C_i
    for all (w, p) in W_i **do**
      $S_{\text{new}} \leftarrow S_i + [w]$
      $P_{\text{new}} \leftarrow P_i \times p$
      $Q' \leftarrow Q' \cup \{(S_{\text{new}}, P_{\text{new}})\}$
    end for
  end for
  $Q \leftarrow$ Select top-B beams from Q' based on scores
  $C_q \leftarrow$ Extract top-n sequences from Q
end while
return C_q

where D_α denotes the dense docid retriever. After receiving the relevance feedback from D_α, the generative model is encouraged to assign higher estimated probabilities to more relevant docids through a ranking-oriented loss. To achieve this goal, following [49], we choose four types of ranking losses as the ranking-oriented learning objective for ranking order preservation. Formally, assuming that the predicted ranking scores of generative retriever G_θ over candidate documents are defined as:

$$\Psi_{q,d'} = \frac{\exp(G_\theta(d'|q))}{\sum_{d' \in C_q} \exp(G_\theta(d'|q))},$$

we regard the relevance $\Phi_{q,d'}$ or $\Phi_{\widehat{d'},d'}$ as the teacher score to optimize the generative retriever. We investigate several potential ranking losses as follows:

- **Margin Ranking Loss** is a loss function commonly used in ranking tasks, where the goal is to learn a ranking model that can order items or instances based on their relevance or preference. The margin ranking loss encourages the model to correctly rank relevant items higher than irrelevant items by a certain margin. Given the teacher score, we construct a set of docid pairs (d'_+, d'_-) as the target to optimize the generative retriever G_θ. The ranking loss is defined as:

$$\mathcal{L}_{\text{rank-margin}} = \arg\max_\theta \max(0, \text{margin} - \Phi^{G_\theta}_{q,d'_+} + \Phi^{G_\theta}_{q,d'_-}) \tag{9}$$

where the docid pair (d'_+, d'_-) means the docid d'_+ is more relevant to the query q than the docid d'_-. The hyper-parameter margin is set to 0 in experiments.

- **Mean Square Error** (MSE) is a widely used metric in statistics and machine learning for evaluating the quality of a regression model. Given the query q, it measures the average squared difference between the scores predicted by the dense model and the generative model:

$$\mathcal{L}^{q}_{\text{rank-mse}} = \arg\min_{\theta} \sum_{d' \in C_q} ||\Psi_{q,d'} - \Phi_{q,d'}||, \mathcal{L}^{d}_{\text{rank-mse}} = \arg\min_{\theta} \sum_{d' \in C_q} ||\Psi_{q,d'} - \Phi_{\widehat{d'},d'}||, \tag{10}$$

In addition to above pointwise and pairwise ranking losses, we also select several listwise losses as ranking losses, which are shown to be effective in previous studies [4, 5].

- KL-Divergence is a measure of the difference between two probability distributions. It quantifies how one distribution differs from another, providing a measure of the information lost when one distribution is used to approximate another. Formally, as the ranking loss, it can be expressed as:

$$\mathcal{L}^{q}_{\text{rank-kldiv}} = \arg\max_{\theta} \sum_{d' \in C_q} \Phi_{q,d'} \log \frac{\Phi_{q,d'}}{\Psi_{q,d'}}, \mathcal{L}^{d}_{\text{rank-kldiv}} = \arg\max_{\theta} \sum_{d' \in C_q} \Phi_{\widehat{d'},d'} \log \frac{\Phi_{\widehat{d'},d'}}{\Psi_{q,d'}}. \tag{11}$$

- ListNet is a learning-to-rank algorithm that aims to improve the effectiveness of ranking models in information retrieval tasks. It is a widely used function for ranking order preservation [7]. The cross-entropy is minimized for matching scores of the dense docid retriever D_α and generation probabilities of the generative retriever G_θ over C_q:

$$\mathcal{L}^{q}_{\text{rank-listnet}} = \arg\max_{\theta} \sum_{d' \in C_q} \Phi_{q,d'} \log \Psi_{q,d'}, \mathcal{L}^{d}_{\text{rank-listnet}} = \arg\max_{\theta} \sum_{d' \in C_q} \Phi_{\widehat{d'},d'} \log \Psi_{q,d'}. \tag{12}$$

Both KL-divergence and ListNet methods rely on capturing the listwise similarity between the teachers and students in the context of knowledge distillation. Notably, these approaches naturally highlight the similarities between the top-ranked documents by both functions. This emphasis on the top-ranked documents ensures that knowledge distillation remains consistent with the evaluation metrics used in information retrieval tasks.

Finally, we combine the generative loss and the ranking loss to train the model:

$$\mathcal{L}_{\text{mul}} = (1-\gamma)\mathcal{L}_{\text{gen}} + \gamma\mathcal{L}_{\text{rank}}, \tag{13}$$

where γ represents the weight assigned to the ranking loss. The term $\mathcal{L}_{\text{rank}}$ can utilize any of the ranking losses previously discussed, with the experimental results for different ranking losses detailed in Table 4.

The generative loss and the ranking loss focus on modeling the associations between queries and documents from different perspectives and they can effectively complement each other. After this optimization loop, ranking-oriented signals are transmitted to the generative retriever, so as to enhance its performance on the ranking tasks.

5.4 Optimizing the Dense Docid Retriever with Knowledge Distillation

Intuitively, a more accurate dense docid retriever has the potential to provide improved relevance feedback. However, a model that is solely trained on the query-docid data may not take full advantage of the available document information. To address this limitation, we propose an optimization approach that involves distilling document knowledge from the dense document retriever.

In our framework, we denote the dense docid retriever as D_α and the dense document retriever as D_ϕ. Given a candidate docid d' in the set C_q (which represents the candidate documents for a given query q), we compute the ranking score for the corresponding document d and the query q using D_ϕ. This ranking score reflects the relevance of the document to the query.

To optimize the dense docid retriever D_α, we apply a knowledge distillation loss that minimizes the discrepancy between the ranking score distribution of D_α and D_ϕ. This knowledge distillation loss is defined as follows:

Algorithm 2 Iterative training of ROGER

Input: Generative retriever G_θ, Dense docid retriever D_α, Dense document retriever D_ϕ, Training data $\mathbb{D}$

Initialize G_θ, D_α, D_ϕ from the pre-trained language models
$\mathbb{D}' \leftarrow$ Represent documents as identifiers on $\mathbb{D}$
Warm up G_θ, D_α on $\mathbb{D}'$, warm up D_ϕ on $\mathbb{D}$
while model has not converged **do**
  for training steps of G_θ **do**
    $C_q \leftarrow$ Generate candidates by $G_\theta(\cdot|q)$ with beam search
    $S_{D_\alpha} \leftarrow$ Get the relevance feedback by D_α on C_q
    Update parameters of G_θ with Eq. (13)
  end for
  for training steps of D_α **do**
    $C_q \leftarrow$ Generate candidates by $G_\theta(\cdot|q)$ with beam search
    $S_{D_\phi} \leftarrow$ Compute the ranking scores by D_ϕ on C_q
    Update parameters of D_α with Eq. (14)
  end for
end while

$$\mathcal{L}_{\text{kd}} = \arg\max_{\alpha} \sum_{d' \in C_q} \Phi_{q,d'} \log \frac{\Omega_{q,d}}{\Phi_{q,d'}}, \tag{14}$$

where $\Phi_{q,d'}$ represents the ranking score of the candidate docid d' computed by the dense docid retriever D_α, and $\Omega_{q,d}$ represents the ranking score of the corresponding document d and query q computed by the optimized dense document retriever D_ϕ.

Importantly, since D_α and D_ϕ share a consistent model architecture, it becomes possible to distill document knowledge into the docids as comprehensively as possible. In other words, the dense docid retriever can benefit from the document-level information extracted by the dense document retriever. It is worth noting that different docids may have varying capacities to retain and represent document-level information. Exploring the upper limit of document information that different docids can retain is an interesting direction for future research.

5.5 Summary

Given an original generative retriever G_θ, ROGER updates its parameters with the ranking-oriented loss following three steps. First, candidate docids are generated by G_θ using beam search. Second, it updates the parameters θ based on the relevance feedback from the dense docid retriever D_α. Third, for a more accurate relevance feedback, it updates the parameters α by distilling the document knowledge from the dense document retriever. We present the iterative training framework in Algorithm 2. Note that our proposed ROGER is an incremental training method and applicable to a variety of existing generative retrieval models.

6 EXPERIMENTAL SETTINGS

6.1 Datasets and Evaluation Metrics

Datasets. We perform our experiments on three commonly used benchmarks for document retrieval, Natural Questions 320k [24], MS MARCO Document Ranking [35], and ClueWeb [11].

Natural Questions (NQ) contains questions from real users and Wikipedia articles that are meant to answer them. It consists of approximately 231k articles with 307k training pairs and 7.8k test queries. We remove unnecessary HTML tags and extract the title, abstract, and content from each Wikipedia article during the dataset processing process. Different from previously done in [47], we remove duplicated documents based on the URL of each article instead of the title, which is more reasonable and in line with practical application.

MS MARCO Document Ranking is a large-scale dataset focused on the document ranking task. It consists of 3.2 million documents and a training set with 367,013 queries. As previously done in [58], we construct a set of candidate documents with approximately 320k articles based on the labeled documents. After that, the same data processing procedure as Natural Question 320K is applied.

ClueWeb is a multi-graded relevance dataset, which comprises a web archive consisting of more than 50 million documents sourced from the TREC Web Tracks 2009-2011. We construct a subset by random sampling 300K documents for experiments.

Metrics. We employ the widely used metrics Recall@N (R@N) and Mean Reciprocal Rank (MRR) for evaluating the document retrieval performance on binary relevance dataset, and employ normalized discounted cumulative gain (NDCG@N) and expected reciprocal rank (ERR) for multi-graded relevance dataset.

Table 1. Overall results of sparse retrieval, dense retrieval, and generative retrieval on binary relevance dataset. Document Rep. indicates the method to represent document semantics. The best results are shown in **bold** and the best baselines are <u>underlined</u>. "‡" or "†" indicates the result is significantly better than all baselines or the original generative model with paired t-test at $p < 0.05$ level.

Model	Document Rep.	MS MARCO				Natural Questions			
		R@1	R@5	R@10	MRR@10	R@1	R@5	R@10	MRR@10
Sparse Retrieval									
BM25	sparse terms	0.1894	0.4282	0.5507	0.2924	0.1406	0.3691	0.4793	0.2360
DocT5Query	sparse terms	0.2327	0.4938	0.6361	0.3481	0.1907	0.4388	0.5583	0.2955
Dense Retrieval									
RepBERT	dense vector	0.2525	0.5841	0.6918	0.3848	0.2257	0.5220	0.6565	0.3513
Sentence-T5	dense vector	0.2727	0.5891	0.7215	0.4069	0.2251	0.5200	0.6512	0.3495
DPR	dense vector	0.2908	0.6275	0.7313	0.4341	0.2278	0.5344	0.6858	0.3592
ANCE	dense vector	0.2965	<u>0.6343</u>	<u>0.7428</u>	<u>0.4409</u>	0.2454	0.5421	0.6908	0.3688
Generative Retrieval									
DSI	semantic cluster	0.2574	0.4358	0.5384	0.3392	0.2742	0.4726	0.5658	0.3431
DSI-QG	semantic cluster	0.2882	0.5074	0.6226	0.3845	0.3017	0.5320	0.6637	0.3885
SEAL	n-grams	0.2758	0.5247	0.6101	0.3768	0.2930	0.5412	0.6853	0.4034
NCI	semantic cluster	0.2954	0.5799	0.6728	0.4046	0.3269	<u>0.5582</u>	<u>0.6920</u>	<u>0.4284</u>
Ultron	title + url	<u>0.2982</u>	0.6039	0.6831	0.4253	<u>0.3378</u>	0.5420	0.6705	0.4251
ROGER-NCI	semantic cluster	0.3061‡	0.5902†	0.6878†	0.4202†	0.3320†	**0.5634**‡	0.6980‡	0.4345‡
ROGER-Ultron	title + url	**0.3307**‡	**0.6393**‡	**0.7513**‡	**0.4635**‡	**0.3590**‡	0.5559†	**0.6986**‡	**0.4492**‡

6.2 Baselines

For comparison, we select three classes of models as baselines.

(1) *Sparse Retrieval models* are used to evaluate the query-document relevance by considering the frequency and importance of the query terms within each document. We select two classic methods as baselines. The **BM25** [42]

Table 2. Overall results on multi-graded relevance dataset. Document Rep. indicates the method to represent document semantics. The best results are shown in **bold** and the best baselines are underlined. "‡" or "†" indicates the result is significantly better than all baselines or the original generative model with paired t-test at $p < 0.05$ level.

Model	Document Rep.	ClueWeb			
		NDCG@5	NDCG@10	NDCG@20	ERR@20
Sparse Retrieval					
BM25	sparse terms	0.3095	0.2987	0.2900	0.1634
DocT5Query	sparse terms	0.2485	0.2169	0.1957	0.0985
Dense Retrieval					
RepBERT	dense vector	0.3149	0.3041	0.2917	0.1996
Sentence-T5	dense vector	0.3031	0.3002	0.2964	0.1989
DPR	dense vector	0.3137	0.3104	0.3091	0.2084
ANCE	dense vector	0.3292	0.3263	<u>0.3235</u>	0.2171
Generative Retrieval					
DSI	semantic cluster	0.2405	0.2389	0.2372	0.1372
DSI-QG	semantic cluster	0.3267	0.3159	0.3067	0.2140
SEAL	n-grams	0.3220	0.2987	0.2752	0.1566
NCI	semantic cluster	0.3340	0.3241	0.3157	0.2275
Ultron	title + url	<u>0.3358</u>	<u>0.3287</u>	0.3156	<u>0.2288</u>
ROGER-NCI	semantic cluster	0.3397†	0.3324†	0.3268†	0.2371†
ROGER-Ultron	title + url	**0.3627**‡	**0.3587**‡	**0.3541**‡	**0.2421**‡

method incorporates the tf-idf feature to determine term weights, while **DocT5Query** [38] expands a document with queries predicted by a T5 model [41].

(2) *Dense Retrieval models* embed the query and the document into separate vectors and calculate their similarity. **RepBERT** [53] and **Sentence-T5** [36] are two basic baselines with BERT encoder and T5 encoder respectively. **DPR** [22] and **ANCE** [50] are two stronger baselines that are trained with hard negatives.

(3) *Generative Retrieval models* have been devised for document retrieval recently. **DSI** [46] is the first generative retrieval model which clusters all documents into a decimal tree and uses the paths as document identifiers. Based on it, **SEAL** [2] uses every n-gram within a passage as a potential identifier and creates a FM-Index to retrieve documents. **DSI-QG** [60] and **NCI** [47] add a query generation module for data augmentation. **Ultron** [58] uses different types of docid for generation. We implement the Ultron-URL variant as the baseline. **ROGER** is our proposed model and we regard Ultron and NCI as the original generative retrieval models respectively to test the performance. Note that for all models, we consistently choose T5-base as the backbone and deduplicate documents with URL as the corpus to ensure fair comparison, which may cause discrepancies with the results of the original paper.

6.3 Implementation Details

Warming up. For the generative retriever, we leverage the pre-trained model T5 [41] as the backbone and update the parameters following existing models, such as NCI [47] or Ultron [58]. For the dense retriever, which is a dual encoder structure, it is trained with a contrastive loss over hard negatives following [22]. To ensure the consistency, we also use the T5 tokenizer for word segmentation and apply the T5 encoder to learn the dense vectors.

Training and inference. The complete training of the model is an iterative update process. For each iteration, we generate 8 candidate docids by the generative retriever with beam search. During the training of the generative retriever, the weight of ranking loss γ is set to 0.9. We set the learning rate as 1e-4 and the batch size as 32 (per GPU). During the training of the dense docid retriever, the batch size is set to 32 and the learning rate is 3e-5. For inference, we apply the constrained beam search to ensure the validity of the generated docids. Due to the memory and time overhead of beam search, we choose a maximum of 20 for the beam size. All experiments are conducted using 4 NVIDIA A100 GPUs. During the warming up phase, we allocate 12 hours to warm up the generative retriever and 6 hours for the dense retriever. In the ranking-oriented optimization phase, each complete loop (generator + dense retriever) takes approximately 4 hours. During the inference phase, a single GPU is used, and with the beam size set to 20, each query takes about 0.3 seconds.

7 EXPERIMENTAL RESULTS

7.1 Overall Performance

The overall results of models are listed in Table 1 and Table 2. Some findings are summarized as follows.

(1) **The comparison of original and our ranking-oriented generative retrieval**. No matter which generative retrieval model is used to initialize the model, ROGER achieves better performance in terms of all metrics across the three datasets, which shows the effectiveness of our ranking-oriented training strategy. Compared to Ultron, ROGER improves the performance by over 8.98% on MRR@10 on MS MARCO dataset. For NQ dataset, ROGER outperforms NCI by over 1.42% on MRR@10. These findings strongly suggest that the incorporation of ranking signals into traditional generative retrieval is a highly promising approach for enhancing performance in ranking tasks.

(2) **The comparison of MS MARCO and NQ datasets**. On MS MARCO dataset, the improvement brought by the ranking-oriented training strategy is more obvious, resulting in 8.98% and 3.85% improvement on MRR@10 over Ultron and NCI, respectively. However, the improvement percentage for the NQ dataset is comparatively lower, registering at 5.66% and 1.42%. A possible reason is that dense retrieval methods have comparable or even superior performance with generative retrieval on MS MARCO dataset, but they perform poorly on NQ dataset. Intuitively, a more robust dense model would be expected to provide higher quality ranking signals.

(3) **The comparison of different evaluation metrics**. Compared with dense retrieval, original generative retrieval models tend to exhibit superior performance at R@1, but exhibit inferior results at R@10. By incorporating ranking signals from dense retrieval into the optimization of the generative retriever, the improvement on R@10 is more obvious than on R@1. This highlights that the integration of ranking loss into the optimization process of the generative model effectively addresses its shortcomings in ranking tasks, while maintaining its strengths.

(4) **The comparison of ROGER-Ultron and ROGER-NCI**. The superiority of ROGER over Ultron is pronounced on both datasets. The primary distinction between these two generative retrieval models lies in their docid representations. Ultron utilizes natural language, such as title and URL, while NCI employs a string of IDs with semantic information. In general, the dense retrieval model works on query text and document text. Therefore, the natural language-style docid is more suitable for knowledge distillation, resulting in better relevance feedback to the generative retriever.

In summary, these results indicate that **our ranking-oriented training strategy, which incorporates ranking signals from dense models for model training, can enhance the performance of the original generative retrievers on ranking tasks.**

Table 3. Performance comparison with different relevance feedback and scoring methods. (doc retriever: *w/o.* dense docid retriever; docid retriever: *w/o.* dense document retriever; kd: knowledge distillation; BLEU/ROUGE: *w/o.* two dense retrievers)

Feedback	Scoring Method	MS MARCO				Natural Questions			
		R@1	R@5	R@10	MRR@10	R@1	R@5	R@10	MRR@10
No feedback (Ultron)	-	0.2982	0.6039	0.6831	0.4253	0.3378	0.5420	0.6705	0.4251
Query relevance	doc retriever	0.3152	0.6301	0.7408	0.4502	0.3514	0.5497	0.6869	0.4418
Query relevance	docid retriever	0.3216	0.6321	0.7456	0.4546	0.3558	0.5531	0.6931	0.4457
Query relevance	docid retriever (kd)	**0.3307**	0.6393	**0.7513**	**0.4635**	**0.3590**	**0.5559**	**0.6986**	**0.4492**
Ground-truth relevance	BLEU	0.3300	0.6106	0.6927	0.4461	0.3511	0.5450	0.6614	0.4346
Ground-truth relevance	ROUGE	0.3294	0.6002	0.6679	0.4401	0.3532	0.5421	0.6586	0.4322
Ground-truth relevance	doc retriever	0.3096	0.6201	0.7112	0.4419	0.3502	0.5485	0.6816	0.4383
Ground-truth relevance	docid retriever	0.3188	0.6320	0.7283	0.4517	0.3539	0.5507	0.6898	0.4423
Ground-truth relevance	docid retriever (kd)	0.3294	**0.6445**	0.7460	0.4623	0.3576	0.5541	0.6953	0.4470

Table 4. Performance comparison with different ranking losses. Margin Ranking loss takes advantage of the order signals and optimizes the model in a pairwise manner, while the other three losses fit the ranking logits in a pointwise (MSE) or a listwise (KL-divergence & ListNet) manner.

Ranking Loss	Information	MS MARCO				Natural Questions			
		R@1	R@5	R@10	MRR@10	R@1	R@5	R@10	MRR@10
No Ranking Loss (Ultron)	-	0.2982	0.6039	0.6831	0.4253	0.3378	0.5420	0.6705	0.4251
Margin Ranking Loss	ground-truth	0.2541	0.5298	0.5977	0.3657	0.2794	0.4783	0.5986	0.3503
Margin Ranking Loss	ranking order	0.2929	0.6002	0.7031	0.4219	0.3358	0.5445	0.6813	0.4250
Mean Square Error	ranking logits	0.3190	0.6197	0.7278	0.4431	0.3485	0.5440	0.6838	0.4376
KL-divergence	ranking logits	0.3255	0.6380	0.7421	0.4586	**0.3594**	0.5541	0.6970	0.4483
ListNet (ROGER-Ultron)	ranking logits	**0.3307**	**0.6393**	**0.7513**	**0.4635**	0.3590	**0.5559**	**0.6986**	**0.4492**

7.2 Study of Relevance Feedback Strategy

In the model training of ROGER, an important step is to receive relevance feedback signals from the dense docid retriever. As stated in Sec 5.3, there are two types of relevance: query relevance and ground-truth relevance. For both, we experiment with either a standalone dense document retriever or a pure dense docid retriever for scoring, aside from the dense docid retriever that's improved through knowledge distillation. In addition, for the ground-truth relevance, we try two other metrics widely used in dialog systems, BLEU and ROUGE, as the feedback.

The results are shown in Table 3. It can be observed that removing either the dense document retriever or dense docid retriever leads to a decline in ranking results. This highlights the necessity of the intermediate model and the effectiveness of knowledge distillation. Furthermore, leveraging the feedback of standalone docid retriever is better than directly using the ranking signals from the dense document retriever. It shows that the docid retriever can provide more suitable relevance feedback for model training. This is in line with our conjecture about data mismatch. When comparing the two types of relevance feedback, we find that using query relevance as ranking signals performs slightly better than using ground-truth relevance. A possible reason is that the query relevance is consistent with the goal of the final document retrieval task. Another interesting finding is from the scoring methods of BLEU and ROUGE. They bring significant improvement over no feedback scenarios on R@1, but the performance on R@10 even gets worse. This implies that if we choose a type of relevance feedback without ranking signals, the imbalance in the performance of each ranking position of the model will be exacerbated.

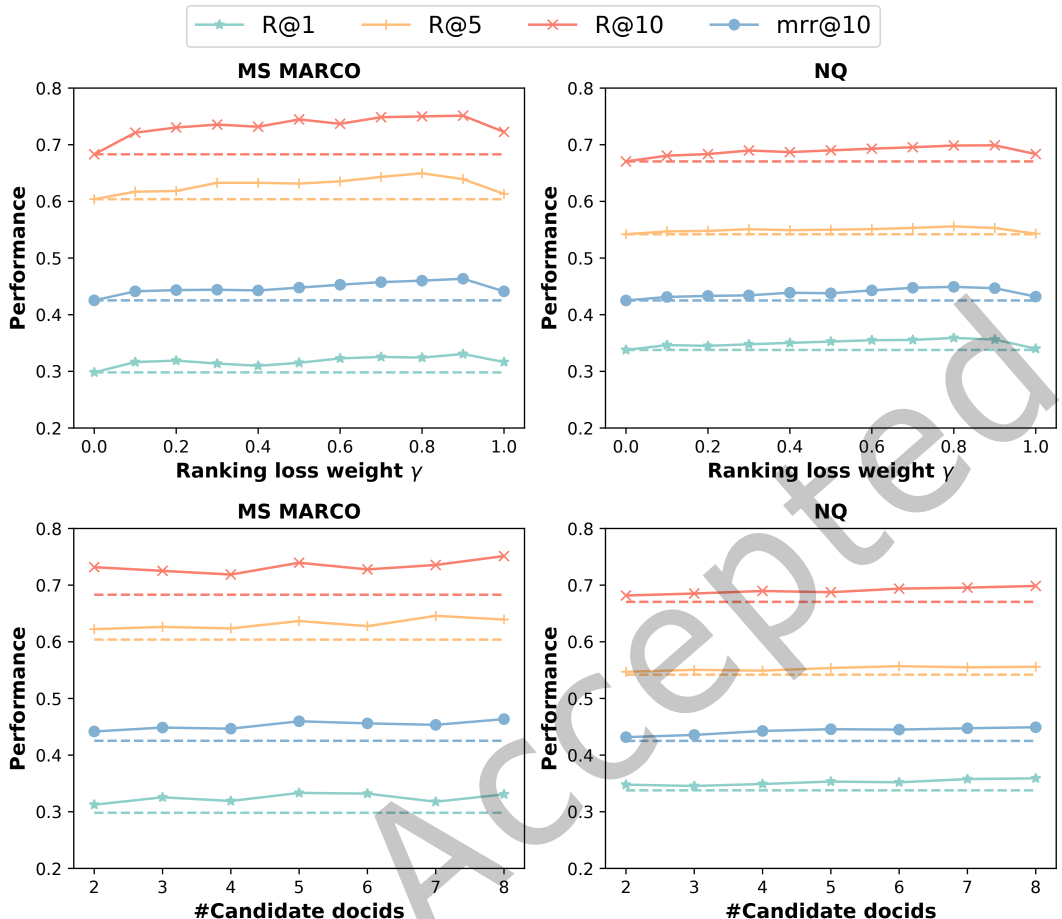

Fig. 6. Performance with different ranking loss weights and the number of candidate docids. The dashed line is the results of Ultron.

7.3 Influence of Different Ranking Losses

When receiving relevance feedback from the dense docid retriever, leveraging these signals to optimize the model remains a challenge. Based on this consideration, we further explore the impact of different ranking losses on model performance. Concretely, we choose four different loss functions, Margin Ranking Loss, Mean Square Error, KL-divergence, and ListNet. Margin Ranking Loss aims to learn the relative order between any two documents. There are two ways to construct document pairs: (a) the ground-truth docid with a random negative; (b) leveraging the ranking order from the dense docid retriever. The other three ranking losses aim to narrow the difference between two distributions, which distill the ranking logits of the dense docid retriever into the generative model.

The comparisons are shown in Table 4. It can be observed that different ranking losses do lead to distinct effects. The overall performances from emulating ranking logits feedback (KL-divergence, ListNet, MSE) are comparatively better than those from learning the pairwise order feedback (Margin Ranking Loss). This indicates that ranking logits can preserve more ranking signals required by the generative retriever. Particularly, ListNet

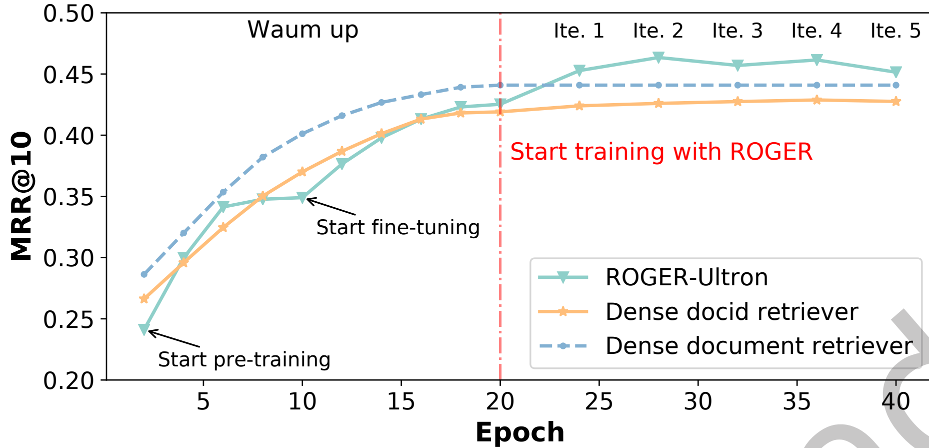

Fig. 7. Study of learning curves on MS MARCO.

and KL-divergence, which force the consistency of distribution between generation probabilities and ranking logits, are better than MSE, which focuses on consistency of ranking scores. Besides, optimizing the model with Margin Ranking Loss causes a severe drop when just using ground-truth information to construct training pairs. By contrast, the dense model provides the ranking order between any two candidates, which brings a boost on R@10.

7.4 Effect of Hyper-parameters

When optimizing the generative retrieval model, the ranking loss weight and the size of candidates are two important hyper-parameters in our framework. To investigate their impact, we train our model with different settings and test their performance. Specifically, we set the ranking loss weight γ in Eq. 13 from 0 to 1 with an interval of 0.1 to test the model performance on each evaluation metric. Limited by the memory cost, we set the maximum size of candidates to 8, and observe changes in performance as the number of candidates increasing. The results are shown in Figure 6.

The ranking loss weight determines the ratio of the generative loss and the ranking loss for each parameter update. It can be seen that a larger ranking loss weight leads to better ranking performance. However, when γ increases to 1, which means only the ranking loss plays a role, the model performance will deteriorate instead. This shows that on a adequately trained generative retrieval model, appropriately increasing γ can help it learn more ranking-oriented signals. But if it relies entirely on the ranking loss, the model's training objectives and inference will be inconsistent, thus leading to performance degradation. Considering the performance on each metric, $\gamma = 0.9$ is the best choice for our framework.

As for the size of candidates, a larger number of candidates would allow more candidates to be considered in relevance feedback, thus increasing the upper bound of the performance. It can be observed that the overall trend of the curve supports this conjecture, but not exactly a linear growth relation. Specifically, the increase in performance is more significant when it comes to R@10, which indicates that the comparison between non-golden candidate docids helps the generative model learn more signals suitable for ranking tasks. Considering the trade-off between resource consumption and model performance, we sample 8 candidates in each iteration.

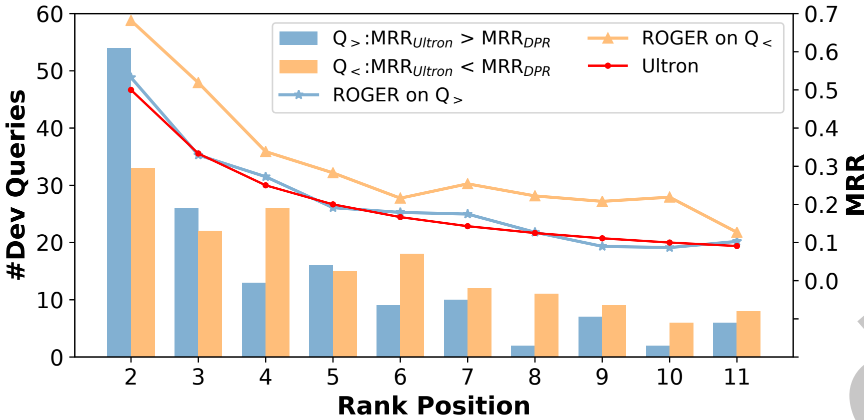

Fig. 8. Study of ranking results on MS MARCO. The ranking position is determined by Ultron's performance, with the red line representing its MRR.

7.5 Study of Learning Curves

In our approach, we aim to enhance the performance of the generative retrieval method by continuously training the model parameters that have been warmed up using the original method. To evaluate the effectiveness of our proposed method, ROGER, we conducted experiments on Ultron and measured its performance using the Mean Reciprocal Rank at 10 (MRR@10) metric on the MS MARCO dataset. We also compared the performance of the dense docid/document retriever throughout the training process.

Figure 7 illustrates the performance curves of Ultron and the two dense retrievers during the warming-up phase. As the training epochs progress, both Ultron and the dense retrievers show concurrent improvements in performance. Eventually, they achieve a comparable level of performance. This observation suggests that the initial warming-up phase effectively trains Ultron to perform competitively with the dense retrievers. However, upon implementing ROGER, which incorporates ranking signals from the two dense retrievers into Ultron, we observed a further improvement in performance. In fact, the performance of Ultron with ROGER surpassed that of the dense document model. This outcome serves as evidence that our approach successfully combines the advantages of both retrieval and dense paradigms, and enhances the model's ability to handle ranking tasks. By leveraging information from the dense retrievers, ROGER enhances Ultron's ranking capabilities and improves its overall performance. Additionally, we found that the model quickly reaches its best performance after two iterations with ROGER. This observation highlights the potential of our method in terms of convergence rate. The ability to achieve optimal performance quickly is beneficial in real-world applications where efficiency is crucial.

7.6 Study of Ranking Results

To provide a clearer comparison between our ranking-oriented training method and the original generative retriever, we conducted an analysis on the development set queries. We divided the queries into two groups based on Ultron's performance compared to the DPR model at each ranking position. One group consisted of queries where Ultron outperformed the DPR model, and the other group consisted of queries where Ultron performed worse. We then evaluated ROGER's performance on these two groups of queries, using Ultron's performance with the original generative model as the baseline. The results of this analysis are presented in Figure 8.

The results indicate that the generative retrieval and dense retrieval models perform differently on different queries. At each rank position, except the first, there is a chance for the dense retrieval model to outperform the generative retrieval model. This suggests that the two retrieval paradigms have complementary strengths and weaknesses. For the group of queries where the generative retrieval model performs better, ROGER achieves similar performance to the original generative model. This outcome suggests that ROGER preserves the advantages brought by the generative paradigm and does not significantly compromise the performance on queries where the generative model is already strong. A possible reason is that the generative retriever tends to have a high confidence in the correct documents for these particular queries, which means it is less susceptible to the influence of potentially noisy feedback from the dense retriever. In contrast, for the group of queries where the dense retrieval model performs better, ROGER clearly outperforms the baseline. This result demonstrates that ROGER can effectively learn the ranking signals of the dense model while maintaining its own advantages derived from the generative paradigm. By incorporating ranking signals, ROGER enhances its ability to handle queries that are better suited for the dense retrieval approach.

7.7 Discussion

The above experiments demonstrate the effectiveness of our proposed model. In order to further understand the advantages of this method, we discuss the following three aspects:

- What additional knowledge does the dense model provide for the generative retriever to make it work better?

In original generative retrieval, the optimization objective only maximizes the generation ability of the relevant docid, which is a one-point distribution. In ROGER, it assumes a non-deterministic distribution in which other candidate docids are also assigned probability mass according to their relevance. The relative relevance over candidate docids, which is crucial in ranking tasks, can be regarded as the additional knowledge provided by the dense model.

- Why is using a docid retriever to provide relevance feedback better than using a document retriever directly?

The dense docid retriever acts as a bridge, connecting the two heterogeneous models. Since docid and document text belong to two different spaces, directly using the dense document retriever to provide relevance feedback is noisy. Therefore, we need a model that belongs to the docid space to provide the ranking signals. At the same time, in order to reduce the information loss from document text to docid, this model can learn document knowledge from the dense document retriever through knowledge distillation. Detailed experimental comparisons are presented in Sec. 7.2.

- In the era of large language models, will generative retrieval replace dense retrieval?

In the era of large language models, both generative retrieval and dense retrieval techniques continue to be valuable and have their own strengths and applications. It is unlikely that generative retrieval will completely replace dense retrieval, but rather they will coexist and complement each other in various scenarios. Generative retrieval excels at generating contextually relevant and coherent documents based on a given query or prompt, while dense retrieval excels at capturing semantic similarities between queries and documents, emphasising on efficient similarity matching.

8 LIMITATIONS

Despite the progress achieved through ROGER in the realm of generative retrieval, certain challenges persist that need to be addressed. A primary concern is the scalability of the model in the context of internet-scale data repositories. As the size of the data corpus grows, it necessitates a corresponding increase in the model's capacity. The complex interplay between the size of the corpus and the requisite capacity of the model is an area that requires further investigation. Another significant limitation is the updating mechanism of the model-based indexer. The current system lacks an efficient method for integrating new documents without undergoing a complete retraining process. This is a critical issue, as the ability to seamlessly add fresh content is crucial for maintaining an up-to-date and relevant retrieval system. Future research should prioritize the development of strategies that allow for the incremental updating of the model. Such advancements are crucial for enhancing the utility and adaptability of ROGER in the ever-evolving landscape of model-based information retrieval.

9 CONCLUSION AND FUTURE WORK

In this paper, we delve into the optimization objectives of generative retrieval and dense retrieval and identify a limitation of generative retrieval in ranking tasks stemming from a lack of adequate ranking signals. Building upon this observation, we present ROGER, a novel training strategy that integrates relevance feedback from dense retrieval to enhance the training of a ranking-oriented generative retrieval model. Through extensive experiments on three publicly available datasets, we demonstrate that ROGER effectively enhances the performance of various generative retrieval models.

The results of our experiments provide empirical evidence of the effectiveness of ROGER in improving the ranking capabilities of generative retrieval models. However, there remain opportunities for further exploration in combining these two paradigms. In the future, we plan to investigate alternative approaches, such as reinforcement learning and adversarial learning, to further enhance the integration of generative and dense retrieval. Moreover, we are also interested in the use of large language models in information retrieval. These techniques hold the potential to unlock additional performance gains and refine the synergy between generative and dense paradigms in the context of document retrieval.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China No. 62272467, the fund for building world-class universities (disciplines) of Renmin University of China, and Public Computing Cloud, Renmin University of China. The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE, and Beijing Key Laboratory of Big Data Management and Analysis Methods.

REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. *CoRR* abs/2010.00768 (2020).

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. *CoRR* abs/2204.10628 (2022).

[3] Roi Blanco and Christina Lioma. 2012. Graph-based term weighting for information retrieval. *Inf. Retr.* 15, 1 (2012), 54–92.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of ICML'2005.* ACM, 89–96.

[5] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[6] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *ICLR*. OpenReview.net.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML (ACM International Conference Proceeding Series, Vol. 227)*. ACM, 129–136.

[8] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative Evidence Retrieval for Fact Verification. In *SIGIR*. ACM, 2184–2189.

[9] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks. In *CIKM*. ACM, 191–200.

[10] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *NeurIPS*. 5199–5212.

[11] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. 2009. Overview of the TREC 2009 Web Track. In *TREC (NIST Special Publication, Vol. 500-278)*. National Institute of Standards and Technology (NIST).

[12] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *CoRR* abs/1910.10687 (2019).

[13] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 1897–1907.

[14] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 65–74.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[16] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *CoRR* abs/2109.10086 (2021).

[17] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR*. ACM, 2288–2292.

[18] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement Lexical Retrieval Model with Semantic Residual Embeddings. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12656)*. Springer, 146–160.

[19] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized Product Quantization. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 4 (2014), 744–755.

[20] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. ACM, 55–64.

[21] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 113–122.

[22] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP (1)*. Association for Computational Linguistics, 6769–6781.

[23] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Leveraging Semantic and Lexical Matching to Improve the Recall of Document Retrieval Systems: A Hybrid Approach. *CoRR* abs/2010.01195 (2020).

[24] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. 2019. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics* 7 (2019), 452–466.

[25] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 6086–6096. https://doi.org/10.18653/v1/p19-1612

[26] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*. Association for Computational Linguistics, 7871–7880.

[27] Xiaoxi Li, Zhicheng Dou, Yujia Zhou, and Fangchao Liu. 2024. Towards a Unified Language Model for Knowledge-Intensive Tasks Utilizing External Corpus. *arXiv preprint arXiv:2402.01176* (2024).

[28] Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2024. From Matching to Generation: A Survey on Generative Information Retrieval. *arXiv preprint arXiv:2404.14851* (2024).

[29] Xiaoxi Li, Yujia Zhou, and Zhicheng Dou. 2024. UniGen: A Unified Generative Framework for Retrieval and Question Answering with Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8688–8696.

[30] Yixin Liu, Pengfei Liu, Dragomir R. Radev, and Graham Neubig. 2022. BRIO: Bringing Order to Abstractive Summarization. In *ACL (1)*. Association for Computational Linguistics, 2890–2903.

[31] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[32] Sanket Vaibhav Mehta, Jai Prakash Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2022. DSI++: Updating Transformer Memory with New Documents. *CoRR* abs/2212.09744 (2022).

[33] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. *SIGIR Forum* 55, 1 (2021), 13:1–13:27.

[34] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[35] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings, Vol. 1773)*. CEUR-WS.org.

[36] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. *CoRR* abs/2108.08877 (2021).

[37] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *EMNLP (Findings) (Findings of ACL, Vol. EMNLP 2020)*. Association for Computational Linguistics, 708–718.

[38] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019).

[39] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *CoRR* abs/1904.08375 (2019).

[40] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 1532–1543.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[42] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389.

[43] François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. ACM, 59–68.

[44] Weizhou Shen, Yeyun Gong, Yelong Shen, Song Wang, Xiaojun Quan, Nan Duan, and Weizhu Chen. 2022. Joint Generator-Ranker Learning for Natural Language Generation. *CoRR* abs/2206.13974 (2022).

[45] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to Tokenize for Generative Retrieval. *CoRR* abs/2304.04171 (2023).

[46] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. *CoRR* abs/2202.06991 (2022).

[47] 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. 2022. A Neural Corpus Indexer for Document Retrieval. *CoRR* abs/2206.02743 (2022).

[48] Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. 2023. NOVO: Learnable and Interpretable Document Identifiers for Model-Based IR. In *CIKM*. ACM, 2656–2665.

[49] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, and Xing Xie. 2022. Distill-VQ: Learning Retrieval Oriented Vector Quantization By Distilling Knowledge from Dense Embeddings. In *SIGIR*. ACM, 1513–1523.

[50] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*. OpenReview.net.

[51] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. In *CIKM*. ACM, 2487–2496.

[52] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. ACM, 1503–1512.

[53] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *CoRR* abs/2006.15498 (2020).

[54] Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, and Zhao Cao. 2023. Term-Sets Can Be Strong Document Identifiers For Auto-Regressive Search Engines. *CoRR* abs/2305.13859 (2023).

[55] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. ACM, 575–584.

[56] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing Generative Retrieval with Reinforcement Learning from Relevance Feedback. In *EMNLP*. Association for Computational Linguistics, 12481–12490.

[57] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, and Ji-Rong Wen. 2022. DynamicRetriever: A Pre-training Model-based IR System with Neither Sparse nor Dense Index. *CoRR* abs/2203.00537 (2022).

[58] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer. *CoRR* abs/2208.09257 (2022).

[59] Yujia Zhou, Jing Yao, Ledell Wu, Zhicheng Dou, and Ji-Rong Wen. 2023. WebUltron: An Ultimate Retriever on Webpages Under the Model-Centric Paradigm. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[60] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation. *CoRR* abs/2206.10128 (2022).