Planning Ahead in Generative Retrieval: Guiding Autoregressive Generation through Simultaneous Decoding

Hansi Zeng
University of Massachusetts Amherst
United States
hzeng@cs.umass.edu

Chen Luo
Amazon
United States
cheluo@amazon.com

Hamed Zamani
University of Massachusetts Amherst
United States
zamani@cs.umass.edu

ABSTRACT

This paper introduces PAG–a novel optimization and decoding approach that guides autoregressive generation of document identifiers in generative retrieval models through simultaneous decoding. To this aim, PAG constructs a set-based and sequential identifier for each document. Motivated by the bag-of-words assumption in information retrieval, the set-based identifier is built on lexical tokens. The sequential identifier, on the other hand, is obtained via quantizing relevance-based representations of documents. Extensive experiments on MSMARCO and TREC Deep Learning Track data reveal that PAG outperforms the state-of-the-art generative retrieval model by a large margin (e.g., 15.6% MRR improvements on MS MARCO), while achieving 22× speed up in terms of query latency.

KEYWORDS

Generative retrieval, neural ranking models, ranking optimization

ACM Reference Format:
Hansi Zeng, Chen Luo, and Hamed Zamani. 2024. Planning Ahead in Generative Retrieval: Guiding Autoregressive Generation through Simultaneous Decoding. In *Proceedings of the 47th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Generative Retrieval (GR) [3, 37, 64, 67, 73, 80] , also referred to as differentiable search index, provides a novel paradigm for information retrieval, diverging from the traditional "index-then-retrieve" approach employed in sparse and dense retrieval models [19, 29, 30, 57, 71]. In GR, each document is first assigned a unique document identifier (DocID); then, a generative retrieval model, often based on a large language model (LLM), is trained to generate relevant DocIDs in response to a query [43, 64, 67, 73, 82]. A distinct property of GR models is their capacity to consolidate the corpus information within their parameters, which makes their integration into other generation tasks that benefit from information retrieval differentiable and seamless [73]. Important examples of such applications include knowledge-intensive text generation [22, 24, 29, 35, 58, 72] and personalized generation [59].

Each DocID in generative retrieval is often consist of a sequence of tokens. Hence, they generate DocIDs autoregressively; meaning that they generate one token at a time, conditioned on the query encoding and the previously generated tokens. Borrowed from the language modeling literature, the (constrained) beam search algorithm [43, 64, 67, 73, 82] is used for generation during inference. However, unlike language generation where multiple equally-acceptable outputs exist, each relevant document in generative retrieval is represented with only one identifier. Therefore, since beam search is a local search algorithm that tends to get stuck in local optima [61, 79], if all prefixes of this identifier do not survive the pruning process of beam search, there is no way to recover and the GR model would fail at retrieving the corresponding relevant document. Even though RIPOR [73]—the current state-of-the-art generative retrieval model—achieves substantial improvements by emphasizing on accurate generation of DocID prefixes during training, our experiments show that many relevant DocIDs still exist that cannot survive beam search pruning in RIPOR. According to results presented in Figure 1, we observe that increasing the beam size would significantly affect the retrieval effectiveness, and even using a large beam size, e.g., 1000, still cannot meet the brute force decoding performance where every document in the corpus is scored (see Section 3.1.3 for more details).

Motivated by these findings, we propose PAG–a novel optimization and decoding approach that guides autoregressive generation through an efficient simultaneous DocID decoding for approximating document-level scores. In other words, each DocID consists of a set-based and a sequential identifier. PAG first decodes the set-based identifier, in which token ordering does not matter thus can be done in a single decoding step for approximating document-level scores. PAG then continues decoding the sequential identifier conditioned on the previous generations. Our hypothesis is that conditioning autoregressive decoding on document-level scores produced by simultaneous (i.e., set-based) decoding reduces the likelihood of a relevant prefix to be pruned by (constrained) beam search. Therefore, we revisit both optimization and decoding of generative retrieval models according to this hypothesis.

Inspired by the effectiveness of bag-of-words assumption in many existing retrieval models [12, 48, 56, 57, 60, 75], we construct our set-based document identifiers based on lexical tokens. Following Zeng et al. [73], we also use residual quantization (RQ) over the relevance-based representations produced for each query and document by our GR model to form the sequential identifiers. We suggest a three-stage optimization pipeline, one for set-based DocID generation, one for sequential DocID generation, and one end-to-end training for joint set-based and sequential generation.

We conduct our evaluation on standard large-scale passage retrieval benchmarks including MSMARCO [4] and TREC Deep Learning Track Data from 2019 and 2020 [15, 16], in which the corpus consists of 8.8 million passages. Compared to the current state-of-the-art generative retrieval model, i.e., RIPOR [73], PAG demonstrates 15.6% relative improvement in terms of MRR@10 on MS-MARCO Dev set and 12.3% and 10.9% improvements in terms of NDCG@10 on TREC-DL 2019 and 2020, respectively. This is while PAG uses a 10× smaller beam size, resulting in 22× improvement in terms of query latency when using a single A100 GPU for inference. Extensive ablation studies and analysis demonstrate the impact of the decisions we made in designing the PAG framework. Even though the goal is not to compare with non-generative retrieval models, our experiments demonstrate improvements over several effective dense retrieval models. For instance, compared to TAS-B [25], RocketQA [51], and TCT-ColBERT [39], PAG achieves 11.9%, 4.1%, and 14.9% MRR@10 improvements on the MSMARCO Dev set, respectively. Another significant advantage of PAG over dense retrieval models is its memory efficiency. For example, it requires 7.7× less memory to index the entire corpus (8.8 million passages) compared to single-vector dense retrieval models.

To improve reproducibility and foster research in generative retrieval, we open-source our codebase and release trained model parameters: https://anonymous.4open.science/r/PAG-CAF0.

2 RELATED WORK

Classic Neural IR Models: With the emergence of large language models (LLMs) [14, 17, 40, 45, 52] and large-scale information retrieval datasets [4, 33], neural-based IR models have demonstrated superior results over the traditional lexical-matching models, such as BM25 [57]. In general, these IR models can fall into three categories: (1) cross-encoder models [44, 50, 81], (2) dense retrieval models [20, 23, 25, 29, 31, 39, 41, 51, 69, 70, 74, 76], and (3) sparse retrieval models [12, 13, 18, 19]. The cross-encoder model is often parameterized with LLMs, such as BERT [17] or T5 [52], and takes the concatenation of query and document pair as input to predict their relevant score. This model is effective but slow and is usually used for re-ranking. As for retrieval, the dense retrieval model often uses the bi-encoder architecture to encode the query and document separately into the low-dimensional hidden space and apply the approximate nearest neighborhood (ANN) [42, 70] search for fast retrieval. Sparse retrieval is an alternative method for retrieval, in which it encodes the query and document into the high-dimensional vector space, and usually, each element in the vector represents the importance score of a certain token. To filter out those useful tokens, the L1 [71] or FLOPs [18, 19, 46] regularizer will be incorporated into the objective function to sparsify the high-dimension vectors. For retrieval, the inverted index will be employed similar to BM25.

Generative Retrieval Models: Generate Retrieval (GR), diverges from the traditional "index-then-retrieve" paradigm used in the sparse and dense retrieval models, offering a novel approach for document retrieval. In GR, each document is represented as a unique document identifier (DocID), and a sequence-to-sequence model is trained to generate relevant DocIDs given a query.

DocIDs are usually fixed in the fine-tuning stage and hence serving as bottleneck for affecting the effectiveness of GR models. Usually, DocIDs fall into two categories: (1) semantic-based DocIDs, and (2) word-based DocIDs. Semantic-based DocIDs are usually created using quantization [6, 53, 73, 80] or hierarchical clustering algorithms [43, 62, 64, 67] on document representations to capture semantic relationships among documents. In contrast, word-based DocIDs are directly constructed from the document content, including titles [5, 8, 9, 34], n-grams [3, 7, 36, 37, 68], URLs [54, 80], and significant words [78].

During inference, search algorithms like constrained beam search [43, 64, 67, 73] or FM-index [3, 37] are used to generate valid DocIDs given a query. As for fine-tuning, the early works [3, 43, 64, 67] directly optimize the model using the (sequence-to-sequence) cross-entropy loss. Recently, [36, 73] demonstrates that utilizing the learning-to-rank loss can further enhance the model performance. Data augmentation approaches, such as using pseudo queries [67, 80, 82] are also proven to be useful as they can mitigate the distribution mismatches between the index and retrieval phases. While GR models have shown promising results on the small-scaled datasets, such as NQ-320K [64] and MSMARCO-100K [49], their effectiveness in large-scale benchmarks remains a subject of debate [49]. Addressing this, Zeng et al. [73], recently introduced RIPOR, a framework that enhances the GR model with relevance-based DocID initialization and prefix-oriented ranking optimization. RIPOR has demonstrated competitive performance to the state-of-the-art dense retrieval baselines on the standard MSMARCO-8.8M benchmark.

3 METHODOLOGY

3.1 Preliminaries and Motivations

3.1.1 ***Generative Retrieval.*** In generative retrieval, each document is symbolized with a unique identifier, which is commonly termed as *DocID*. Generative retrieval models are often developed based on large language models to take a query string and generate a ranked list of DocIDs, with respect to their generation probability in descending order. Following the probability ranking principle [55], these generation probabilities are expected to model the probability of relevance for the corresponding documents. A constrained beam search algorithm [64] is used for DocID decoding during inference. The decoded DocIDs are then mapped back to their corresponding documents, which form a final document ranking for the given query.

Formally, let M denote a generative model with an encoder-decoder architecture. The DocID for each document d in the corpus C is represented as $c^d = [c_1^d, \ldots, c_L^d]$, where L is the length of DocIDs. The model M is often trained to generate the DocIDs autoregressively for any given query q. To generate the i^{th} DocID token c_i^d, the model is conditioned on the previously generated tokens, denoted as $c_{<i}^d = [c_1^d, \ldots c_{i-1}^d]$ as well as the query encoding. Therefore, the model generates the hidden representation for the DocID token c_i^d as follows:

$$\mathbf{h}_i^d = \text{Decoder}(\text{Encoder}(q), c_{<i}^d) \in \mathbb{R}^D \tag{1}$$

Each DocID token is associated with a D-dimensional embedding. Let us assume $\mathbf{E}_i \in \mathbb{R}^{V \times D}$ represents the token embedding table

at position i, where V is the DocID vocabulary size.[1] Hence, the corresponding embedding for DocID token c_i^d is represented as $\mathbf{E}_i[c_i^d] \in \mathbb{R}^D$. Note that, the embedding table at each position can be distinct, that is to say, $\mathbf{E}_i \neq \mathbf{E}_j : \forall i \neq j$.

We follow the scoring function introduced by RIPOR [73]—the current state-of-the-art generative retrieval model—to compute the query-document relevance scores (i.e., the DocID generation score in response to the query) as follows:

$$s(c^d; q) = \sum_{i=1}^{L} \mathbf{E}_i[c_i^d] \cdot \mathbf{h}_i^d \tag{2}$$

3.1.2 ***Constrained Beam Search.*** The generative model M often generates each DocID autoregressively using constrained beam search [43, 64, 67, 73, 82]. At each decoding step i, the beam search algorithm maintains the top k prefixes with the highest probabilities (denoted by P_i^{topk}, where $|P_i^{\text{topk}}| = k$) and expands each prefix by one token. Therefore, at each decoding step, many scored prefixes are pruned due to their low probability. The constrained beam search algorithm additionally uses a prefix tree [64] to keep track of valid next tokens for each prefix. The prefix tree is built based on all DocIDs $\{c^d : \forall d \in C\}$, where C is the corpus. Therefore, constrained beam search ensures that every newly generated prefix belongs to at least one valid DocID. This can be accomplished using the following masking function g, defined for any sequence length $1 \leq i \leq L$, based on the prefix tree:

$$g([c_1, c_2, \cdots, c_i]) = \begin{cases} 0 & \text{if } [c_1, c_2, \cdots, c_i] \text{ is a valid prefix.} \\ -\infty & \text{if } [c_1, c_2, \cdots, c_i] \text{ is not a valid prefix.} \end{cases}$$

Therefore, at the i^{th} decoding step, the constrained beam search algorithm assigns the following score to expand each prefix $c_{<i}^p = [c_1, c_2, \cdots, c_{i-1}] \in P_{i-1}^{\text{topk}}$ by one token:

$$\begin{aligned} s(c_{\leq i}^p; q) &= s(c_{<i}^p; q) + s(c_i^p; q, c_{<i}^p) + g(c_{\leq i}^p) \\ &= s(c_{<i}^p; q) + \mathbf{E}_i[c_i^p] \cdot \mathbf{h}_i^p + g(c_{\leq i}^p) \end{aligned} \tag{3}$$

Based on the scoring function, the top k expanded prefixes with highest probabilities will be maintained for the next step.

3.1.3 ***Pitfalls of (Constrained) Beam Search.*** Beam search is a greedy local search algorithm that tends to get stuck into local optima instead of the global optimum [61, 79]. *Even though beam search has been successfully used in natural language generation [21, 63], we hypothesize that it is not sufficient for developing effective generative retrieval models.* In language generation, there are many alternatives that can be equally acceptable outputs, e.g., grammatically correct sentences with same semantics. Hence, if a word token is dropped through beam search, it is likely that other equally good word tokens be kept for the next decoding step. However, in generative retrieval, each relevant document is represented with a unique DocID and if a prefix of this DocID is pruned by the beam search decoding algorithm, there is no way to recover and that relevant document will not appear in the retrieval result list.

To empirically validate this hypothesis, we focused on the current state-of-the-art generative retrieval model, called RIPOR [73].

[1]Note that DocID vocabulary is different from the input vocabulary and may only contain some unique numbers.

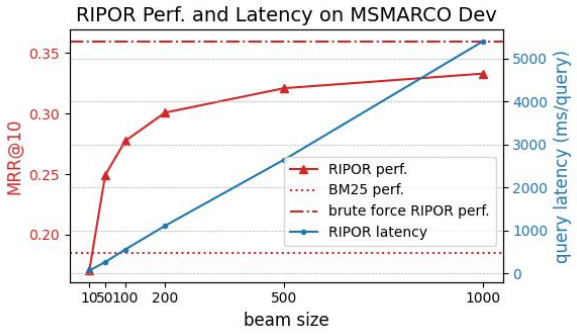

Figure 1: Retrieval effectiveness (MRR@10) and efficiency (query latency) of RIPOR [73] w.r.t different beam sizes on the MS MARCO Dev Set – a standard passage retrieval benchmark with 8.8 million passages. The experiment is conducted on a single A100 GPU with 80GB memory. Best to be viewed in color.

RIPOR uses constrained beam search for DocID decoding. The efficiency and effectiveness of this model for various beam sizes on the MS MARCO Dev set [4] are plotted in Figure 1. We observe that increasing the beam size significantly impacts the effectiveness of RIPOR, even for a short list of 10 documents (MRR@10). Even with a beam size of 1000, RIPOR with constrained beam search cannot meet the brute force decoding performance. Here, brute force decoding means that every document in the corpus is scored by RIPOR without any prefix pruning. On the other hand, large beam size values lead to substantially higher query latency, limiting the practicality of the models. These results validate our hypothesis that the prefix of many relevant documents get harshly pruned by constrained beam search even with relatively large beam size values. This has motivated us to develop alternative decoding methods for generative retrieval models.

For this, we utilize the characteristic of the prefix tree and propose a novel approach that guides the autoregressive generation through simultaneous decoding, which the details will be elaborated in 3.2. We create the set-based and sequential DocIDs to support the simultaneous and sequential decoding respectively, introduced in Section 3.3. Additionally, we propose a three-stage training pipeline for gradual adaption of the model to joint decoding, introduced in Section 3.4. The high-level overview of our framework PAG is illustrated in Figure 2.

3.2 Planning-Ahead Constrained Beam Search

The prefix tree used in the constrained beam search ensures that every expanded prefix $c_{\leq i}^p$ at step i is a valid sequence, thus leading to a set of valid DocIDs once decoding finishes. However, according to Equation (3), which is used in state-of-the-art generative retrieval models, the document ID prefixes are expanded solely based on the contribution by the next token score. Our motivative experiments in Section 3.1.3, on the other hand, demonstrates that this is not sufficient and the prefix of many relevant documents get pruned in constrained beam search, even with large beam size values. This section introduces a new approach for generative retrieval by planning sequential decoding using an efficient simultaneous decoding

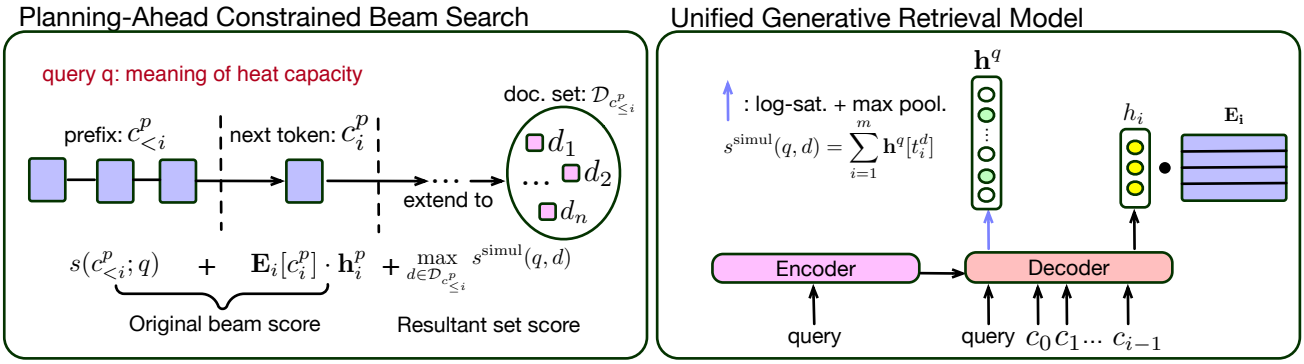

Figure 2: A visualization of the PAG framework. Left: Illustration of simultaneous decoding guiding autoregressive generation with approximate document-level scores. Right: illustration of the model M employing joint decoding of set-based and sequential DocIDs.

that approximates document-level scores. These scores are considered as priors for sequential decoding and let the decoding phase keep the tokens that are likely to lead to high relevance scores once decoding is finished.

3.2.1 ***Simultaneous Decoding.*** Here we introduce an approach called *simultaneous decoding* that produces a score for each document in one decoding step, using $s^{\text{simul}}(q, d)$. The next subsection incorporates this simultaneous document-level decoding into sequential decoding of autoregressive models. To this aim, for each document $d \in C$, we construct a new type of set-based DocIDs t^d, consisting of a *set* of tokens $\{t^d_1, t^d_2, \cdots, t^d_m\}$, where m is the set size for each document d. Note that m is a constant for all documents and is a hyper-parameter. Unlike c^d, t^d is a set and there is no particular order for tokens in t^d, hence they can be decoded simultaneously.

To compute the simultaneous decoding scores for a given query q, we feed the query text to the generative model M to obtain the contextualized representations: $\mathbf{Q} = \text{Decoder}(\text{Encoder}(q), q) \in \mathbb{R}^{|q| \times D}$, where $|q|$ and D respectively represent the query length and the output embedding dimensionality for each token. Let V_T denote the vocabulary size for DocIDs in simultaneous decoding, and $\mathbf{E}_{\text{simul}} \in \mathbb{R}^{V_T \times D}$ denote the associated embedding matrix. Inspired by Formal et al. [18, 19], we apply log-saturation and max pooling operations to compute a weight per DocID token.

$$\mathbf{h}^q = \text{MaxPool}\big(\log(1 + \text{Relu}(\mathbf{E}_{\text{simul}} \cdot \mathbf{Q}^T))\big) \in \mathbb{R}^{V_T} \tag{4}$$

Then the document-level simultaneous relevance score for every (q, d) pair is then computed as:

$$s^{\text{simul}}(q, d) = \sum_{i=1}^{m} \mathbf{h}^q[t^d_i] \tag{5}$$

$s^{\text{simul}}(q, d)$ can also be written as $s^{\text{simul}}(q, t^d)$.

3.2.2 ***Guiding Autoregressive Generation through Simultaneous Decoding.*** PAG uses simultaneous document scoring as priors for computing prefix scores in autoregressive generation. In other words, for decoding any prefix, we consider the maximum approximate document score that can be associated with that prefix as prior. There exist other aggregation functions, such as mean$(\cdot)$ or min$(\cdot)$, to consume here, however, we empirically found max$(\cdot)$ superior. Formally, let $\mathcal{D}$ be a set of top n documents with the highest scores, according to Equation (5). We rewrite Equation (3) as:

$$\begin{aligned} s'(c^p_{\le i}; q) &= \max_{d \in \mathcal{D}_{c^p_{\le i}}} s^{\text{simul}}(q, d) + s(c^p_{\le i}; q) \\ &= \max_{d \in \mathcal{D}_{c^p_{\le i}}} s^{\text{simul}}(q, d) + s(c^p_{<i}; q) + \mathbf{E}_i[c^p_i] \cdot \mathbf{h}^p_i + g(c^p_{\le i}) \end{aligned} \tag{6}$$

where $\mathcal{D}_{c^p_{\le i}} = \{d \in \mathcal{D} | c^p_{\le i} = c^d_{\le i}\}$ is a subset of all documents from $\mathcal{D}$ with the prefix of $c^p_{\le i}$. This modified scoring function conditions next token decoding on an approximate of (future) resultant document score through simultaneous scoring. This can minimize the impact of aggressive document ID pruning in the original beam search algorithm. Based on the modified prefix scoring function, we propose a novel decoding method for generative retrieval and term it as *planning-ahead constrained beam search*. This decoding process is illustrated in Algorithm 1.

3.2.3 ***Computational Cost of Decoding.*** To assess the computational costs of sequential and simultaneous decoding, we utilize Floating Point Operations (FLOPs). The sequential decoding mainly incurs costs from multiple forward calls of the generative model M. Assuming a beam size of k, a sequential DocID length L, and P_m FLOPs per forward call of M, the total FLOPs for sequential decoding are approximately $P_{\text{seq}} = L \cdot k \cdot P_m$. In contrast, simultaneous decoding involves a single forward call of M and additional computations for generating simultaneous relevance scores across the corpus using Equation (5). If the corpus size is $|C|$, the total FLOPs for simultaneous decoding is $P_{\text{simul}} = P_m + |C| \cdot m$. The FLOPs difference is $\Delta P = P_{\text{seq}} - P_{\text{simul}} = P_m \cdot (L \cdot k - 1) - C \cdot m$.

Let us assume M is T5-base, a language model that is often studied for generative retrieval [43, 64, 67, 73] and is considered relatively small compared to today's landscape of LLMs. It requires approximately 7.5×10^9 FLOPs per forward call and given the size of retrieval collections like MSMARCO's 8.8 million passages (used in this study), we infer that simultaneous decoding is notably faster than sequential decoding. This is empirically supported by the query latency comparison in Table 3 in our experiments. While our focus in this paper is on the million-scale dataset, it is important to

note that for billion-scale collections, the simultaneous decoding, if done brute-force, might be slower. Approximation techniques like [1, 2, 26, 27] could be used for maintaining simultaneous decoding's efficiency at billion-scale. We acknowledge that further exploration needed in the future.

Algorithm 1 Planning-Ahead Constrained Beam Search

Require: Generative retrieval model M, query q, beam size k, retrieval corpus C, set-based DocIDs $\{t^d\}_{d\in C}$, sequential DocIDs $\{c^d\}_{d\in C}$, vocabulary size for each token embedding V, sequential DocID length L.

1: Pre-compute document-level scores for every t^d: $s^{\text{simul}}(q, t^d)$ using Eq. (5), and select the top n documents to construct the set $\mathcal{D}$.
2: Find every possible prefix $c^*_{\leq i}$ and the resultant set $\mathcal{D}_{c^*_{\leq i}}$ from $\mathcal{D}$. Based on that, construct a dictionary T, where the key is $c^*_{\leq i}$ and the value is $\max_{d\in\mathcal{D}_{c^*_{\leq i}}} s^{\text{simul}}(q, t^d)$.
3: $B_1 \leftarrow \{< 0, 0, \text{BOS} >\}$
4: **for** $i = 1$ to L **do**
5: $B \leftarrow \emptyset$
6: **for** $(s, s', c^p_{<i}) \in B_i$ **do**
7: **for** $c^p_i \in V$ **do**
8: $c^p_{\leq i} \leftarrow [c^p_{<i}, c^p_i]$, $\max_{d\in\mathcal{D}_{c^p_{\leq i}}} s^{\text{simul}}(q, t^d) \leftarrow T[c^p_{\leq i}]$
9: Apply Eq. (6) and B.add$(< s(c^p_{\leq i}; q), s'(c^p_{\leq i}; q), c^p_{\leq i} >)$
10: **end for**
11: **end for**
12: $B_{i+1} \leftarrow B$.top(k) based on the $s'(\cdot)$. (the second element)
13: **end for**
14: **return** B_{L+1} (the third element is DocID, and the second element is corresponding relevant score)

3.3 DocID Construction

We can envision multiple approaches for constructing the sequence- and the set-based document identifiers. Without loss of generality, in the following, we describe the approach we used in this paper.

3.3.1 ***Sequential DocID Construction.*** We follow the approach of relevance-based DocID initialization introduced in RIPOR [73] to construct the sequential DocIDs (i.e., c^ds), in which we treat the generative model M as a dense encoder. We utilize the encoder-decoder architecture of M by feeding a document text to the encoder and a start token to the decoder. The document representation is then obtained by the contextualized output representations of the decoder:

$$\mathbf{d} = \text{Decoder}(s_0, \text{Encoder}(d)) \in \mathbb{R}^D \tag{7}$$

where s_0 is the start token. By using the M as the dense encoder, we can obtain the dense representation $\mathbf{d}$ for each document d. Subsequently, employing the residual quantization (RQ) algorithm [11], we compile a token embedding tables $\{\mathbf{E}_i\}_{i=1}^L$ to determine the DocID $c^d = [c^d_1, \ldots c^d_L]$ for each document $d \in C$. This optimization process would make each representation $\mathbf{d}$ be approximated as the sequence of token embeddings:

$$\mathbf{d} \approx \sum_{i=1}^{L} \mathbf{E}_i[c^d_i]$$

3.3.2 ***Set-Based DocID Construction.*** The sequential DocID c^d captures the document's semantic information by applying the RQ on derived dense representation d. In the realm of IR, it is well-acknowledged that combining the semantic and lexical information of documents would enhance the retrieval system performance [10, 38, 39, 66, 77]. With this motivation, we set V_T to the vocabulary size of our generative model M, and $\mathbf{E}_{\text{simul}}$ to the embedding table in M. Hence, the tokens $\{t^d_1, t^d_2, \ldots t^d_m\}$ for each set-based DocID t^d will be directly constructed based on the tokenized document content. There exist various methods to score and extract the representative tokens from documents [18, 19, 38, 60]. For the scoring part, the traditional methods, such as TF-IDF [60], weight each token using the corresponding term statistics, e.g., term frequency and inverse document frequency. With the recent advancement of neural lexical models [18], the term importance scores can be directly learned from the supervised training data.

Similar to Equation (4), we take the document content d as the input to the encoder and decoder of the model M to obtain the contextual representation $\mathbf{h}^d \in \mathbb{R}^{V_T}$. Then we follow the training objective used in [18] by linearly combining the MargiMSE loss (retrieval-oriented objective) with FLOPs regularizer [19, 47] to sparsify document representations. We describe the training process in Section 3.4 in detail. The non-zero weights in $\mathbf{h}^d$ represent the importance score of the corresponding tokens. Hence, we select the top m tokens to form the set-based DocID $t^d = \{t^d_1, \ldots t^d_m\}$ for each document d.

3.4 PAG Optimization

The whole optimization process consists of three stages, the first two of which can be trained in parallel. The first two stages are applied to make generative retrieval model capable of predicting set-based DocIDs and sequential DocIDs, respectively. In the final stage, we jointly train the two types of DocIDs together in a unified model, which make it suitable for the planning-ahead constrained beam search decoding introduced in Section 3.2.

3.4.1 ***Generative Retrieval Model for Set-based DocIDs.*** : This stage contains two training phases: (1) the pre-training phase is used for obtaining the set-based DocIDs and model warmup; (2) the fine-tuning phase is to used to train the generative retrieval model for set-based DocIDs prediction.

Pre-training: To obtain the set-based DocIDs t^d, we first treat the generative retrieval model M as a sparse encoder. Given any triple (q, d^+, d^-), where d^+ and d^- represent a relevant and a non-relevant document for q. We follow the MarginMSE method [23] to obtain the teacher margin $T(q, d^+, d^-) = S^T(q, d^+) - S^T(q, d^-)$ for each triplet. Usually, the teacher relevance scores $S^T(q, d)$ for each (q, d) pair is derived from the cross-encoder based teacher model [23, 25]. We obtain the sparse representations for q, d^+, d^- using Equation (4). Based on the sparse representations, We apply the training objective of Formal et al. [18] with MarginMSE loss and FLOPs regularizer [47]. After pre-training, we can apply the

"sparse encoder" to obtain the sparse representation $\mathbf{h}^d$ for every document d. Each non-zero element in the sparse vector represents the important score for the corresponding token. We select top m tokens for each document d as the set-based DocID t^d. We denote the trained model as M^{sp}.

Fine-tuning: We first use M^{sp} as the negative sampler to retrieve the top 100 documents for each query q, and denote the set as $\mathcal{D}_q^{sp}$. We construct the training triples: (q, t^{d^+}, t^{d^-}), $d^- \in \mathcal{D}_q^{sp}$ for fine-tuning the generative retrieval model. The model is initialized with M^{sp}. We use Equation (5) to compute the relevance score for each (q, d) pair. The loss function for each triplet is computed as:

$$\mathcal{L}_{\text{set}}(q, t^{d^+}, t^{d^-}) = \left(s^{\text{simul}}(q, t^{d^+}) - s^{\text{simul}}(q, t^{d^-}) - T(q, d^+, d^-)\right)^2$$

Where $T(\cdot)$ is the teacher margin the same as in the pre-training stage. Motivated by the self-negative training strategy [51, 70, 76], we use the trained model as negative sampler to sample top 100 documents, and merge the negative document set with $\mathcal{D}_q^{sp}$ for each query q, then we apply the same $\mathcal{L}_{\text{set}}$ training loss to fine-tune the model again, and we denote the trained model as M^{set}.

3.4.2 *Generative Retrieval Model for sequential DocIDs.*

Similarly, this training stage contains two phases. The first phase is to construct the sequential DocIDs and the second phase is to fine-tune the generative retrieval model.

Pre-training: The goal of this stage is to obtain the sequential DocIDs c^d. We use the same method as RIPOR [73] that treats the generative retrieval model as a dense encoder and apply the residual quantization (RQ) on the obtained dense representations. We apply the MarginMSE loss with a two-step training strategy to train the model. To construct the training triplets, the negative documents in the first step are sampled from top 100 documents of BM25 and the negatives in the second step are sampled from the top 100 of trained model itself at first step. As introduced in Section 3.3.1, we obtain the dense representation for each document d by using Equation (7), and then applying RQ to obtain the sequential DocIDs. We obtain the trained model M^{ds}, and the corresponding token embeddings $\{\mathbf{E}_i\}_{i=1}^{L}$ after the dense retrieval pre-training.

Similar to RIPOR, this stage also contains a seq2seq pre-training phase. Specifically, we use the doc2query model [12] to generate 10 pseudo-queries for each document, then we take each of the pseudo-queries as input to model and predict the corresponding sequential DocID using a seq2seq loss. The model is initialized with M^{ds} and we denoted the trained model as M^{s2s}.

Fine-tuning: Similar to the previous fine-tuning stage, we construct the training triplets by using the M^{ds} to sample top 100 negative documents and denote the negative set as $\mathcal{D}_q^{ds}$ for each query q. We apply the multi-objective training loss in RIPOR [73] for prefix-oriented fine-tuning. The full-length relevance score between (q, c^d) can be computed via Equation (2). Similarly, the relevance score by generative retrieval model produced by the first i tokens of c^d: $[c_1^d, \dots c_i^d]$ can be computed via Equation (3). Given any triplet (q, c^{d^+}, c^{d^-}), we can use the modified MarginMSE loss for each prefix with length i:

$$\mathcal{L}_{\text{seq}}^i = \left(s(c_{\le i}^{d^+}; q) - s(c_{\le i}^{d^-}; q) - \alpha_i T(q, d^+, d^-)\right)^2$$

where α_i is the monotonically increasing weight w.r.t i ranging from [0, 1], and $\alpha_L = 1$. Refer to Zeng et al. [73] for the details. Therefore, the multi-objective loss is:

$$\mathcal{L}_{\text{seq}} = \sum_{i \in S_L} \mathcal{L}_{\text{seq}}^i$$

S_L is a set containing the sampled prefix lengths and L. The optimized model is termed as M^{seq} which starts from M^{s2s}. Once M^{seq} is trained, we use it as the negative sampler for sampling top 100 documents for each query q, denoted as $\mathcal{D}^{sq}$.

3.4.3 *Unified Optimization for Generative Retrieval with Set-based & Sequential DocIDs.*

In this stage, we train a single model that can predict the set-based DocIDs and sequential DocIDs jointly, which makes it capable of the proposed *planning-ahead constrained beam search*. We initialize the weights of the generative retrieval model M by averaging the weights of M^{set} and M^{seq}. We use the two types of DocIDs to construct the training triples: $\{(q, t^{d^+}, t^{d^-}), (q, c^{d^+}, c^{d^-})\}$ and the negative sample set is $\mathcal{D}_q^{sp} \cup \mathcal{D}_q^{sq}$ for each query q. We obtain the score $s^{\text{simul}}(q, d)$ as Equation (5) for each (q, d) pair. To be compatible with the document-level simultaneous relevance score computation, we apply the slight modification for model M to generate the hidden representation for the next token c_i^d. Different from Equation (1), we additionally feed the query content to the decoder:

$$\mathbf{h}_i^d = \text{Decoder}(\text{Encoder}(q), q, c_{<i}^d) \in \mathbb{R}^D$$

Based on this, $s(c^d; q)$ is computed the same as Equation (2). Therefore, we use the following objective to train the unified generative retrieval model for each triplet:

$$\mathcal{L}_{\text{set}}(q, t^{d^+}, t^{d^-}) + \mathcal{L}_{\text{seq}}(q, c^{d^+}, c^{d^-})$$

4 EXPERIMENTS

4.1 Experiment Settings

4.1.1 *Datasets.*

We assess our model on the MSMARCO passage retrieval benchmark [4] comprising 8.8M passages and 532K train queries. Each trained query contains 1.1 relevant passages on average. We evaluate our model using three evaluation datasets: (1) MSMARCO Dev with 6980 queries with incomplete relevance annotations; (2, 3) TREC-DL 2019 & 2020: the passage retrieval datasets are used in the first and second iterations of TREC Deep Learning Track [15, 16], which contains 43 and 54 queries respectively with a relatively complete relevance annotations done by TREC via pooling. For evaluation, we use the official metric for each dataset: (1) MRR@10 for MSMARCO Dev; (2) NDCG@10 for TREC-DL 19 and 20. We additionally follow Zeng et al. [73] and use Recall@10 for all the three datasets.

4.1.2 *Implementation Details.*

We apply T5-base [52] as the backbone for our generative retrieval model M. For sequential DocID initialization, we apply the residual quantization (RQ) by Faiss [28] implementation. The sequential DocID length L is set to 8 and the vocabulary size V is set to 2048. As for hyper-parameters used in set-based DocIDs, the size of selected tokens (m) is 64. In the pre-training and fine-tuning stages for set-based DocIDs, we set the learning rate to 0.0005 and training epochs to 100. The weights for the FLOPs regularization [18, 19, 47] for query and

Table 1: Experimental results on MSMARCO and TREC Deep Learning Track Data. Highest generative retrieval performances are boldfaced. Superscript $*$ denotes statistically significant improvement compared to all generative retrieval baselines. Superscripts $^{\triangle}$ and $^{\triangledown}$ denote significantly higher and lower performance compared to PAG for sparse and dense retrieval models. (t-test with Bonferroni correction, p_value < 0.01). We use brute-force search for dense retrieval models.

Model	KD	Index Mem.(GB)	MSMARCO Dev		TREC DL 2019		TREC DL 2020	
			MRR@10	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10
Generative Retrieval Methods								
DSI	✗	0.03	.045	.138	.163	.076	.150	.070
DSI-QG	✗	0.03	.105	.292	.320	.138	.328	.120
Ultron-PQ	✗	0.84	.117	.248	.331	.135	.342	.134
NCI-QG	✗	0.03	.153	.352	.403	.167	.394	.159
SEAL	✗	-	.127	-	-	-	-	-
MINDER	✗	12.16	.186	.383	.506	.201	.392	.144
LTRGR	✗	12.16	.255	.531	.598	.238	.553	.182
RIPOR	✓	1.06	.333	.562	.628	.205	.631	.191
PAG	✓	3.27	**.385***	**.670***	**.705***	**.267***	**.700***	**.236***
Some Sparse and Dense Retrieval Methods (For Reference)								
BM25	✗	4.00	.185$^{\triangledown}$	.381$^{\triangledown}$	.512$^{\triangledown}$	.178$^{\triangledown}$	.477$^{\triangledown}$	.164$^{\triangledown}$
docT5query	✗	13.00	.272$^{\triangledown}$	.536$^{\triangledown}$	.642$^{\triangledown}$	.247$^{\triangledown}$	.619$^{\triangledown}$	.224$^{\triangledown}$
ANCE	✗	25.30	.330$^{\triangledown}$	566$^{\triangledown}$	.648$^{\triangledown}$	.239$^{\triangledown}$	.646$^{\triangledown}$	.185$^{\triangledown}$
ADORE	✗	25.30	.347$^{\triangledown}$	.611$^{\triangledown}$	.683$^{\triangledown}$	.264$^{\triangledown}$	.666$^{\triangledown}$	.214$^{\triangledown}$
RocketQA	✓	25.30	.370	-	-	-	-	-
TCT-ColBERT	✓	25.30	.335$^{\triangledown}$	.596$^{\triangledown}$	.670$^{\triangledown}$	.240$^{\triangledown}$	.668$^{\triangledown}$	.218$^{\triangledown}$
MarginMSE	✓	25.30	.325$^{\triangledown}$	.581$^{\triangledown}$	.699$^{\triangledown}$	.250$^{\triangledown}$	.645$^{\triangledown}$	.203$^{\triangledown}$
TAS-B	✓	25.30	.344$^{\triangledown}$	.622$^{\triangledown}$	.717$^{\triangle}$	.255$^{\triangledown}$	.685$^{\triangledown}$	.230$^{\triangledown}$
CL-DRD	✓	25.30	.382$^{\triangledown}$	.651$^{\triangledown}$	.725$^{\triangle}$	.266	.687$^{\triangledown}$	.216$^{\triangledown}$

document representations are 0.01 and 0.008, respectively. In the dense encoder pre-training and prefix-oriented fine-tuning stages for sequential DocIDs, the learning rate is also set to 0.0005 and the training epochs are 50 and 150, respectively. The $S_L = \{4, 8\}$, and the corresponding weights α_is are $\{0.5, 1.0\}$ In the seq2seq pre-training stage, the learning rate is 0.001 and the number of training steps is 250, 000. For the final unified training stage, the learning rate is 0.0005 and the number of training epochs is set to 120. We use Adam optimizer [32] with the linear warmup scheduling, the warmup ratio is set to 4.5% of the total training steps. The beam size is 100, and the top 1000 documents are selected to form $\mathcal{D}$ for inference. The model is trained on 8 A100 GPUs each with 40GB memory. For fair comparison in terms of efficiency, we use an A100 GPU with 80GB memory for all models at inference.

4.1.3 ***Baselines.*** We compare our model with the following state-of-the-art generative retrieval models: DSI [64], DSI-QG [82], NCI-QG [67], Utron-PQ [80], SEAL [3], MINDER [37], LTRGR [36] and RIPOR [73]. To the best of our knowledge, RIPOR provides the state-of-the-art performance among all generative retrieval model baselines. We also select the following competitive sparse and dense retrieval methods as points of reference: BM25 [57], docT5query [12], ANCE [70], ADORE [76], RocketQA [51], TCT-ColBERT [39], MarginMSE [23], TAS-B [25], and CL-DRD [74].

Note that we do not claim improvements over all possible retrieval models. The goal of this paper is to introduce a novel optimization and decoding technique for generative retrieval that significantly advances the state-of-the-art in generative retrieval.

4.1.4 ***Comparison with Baselines.*** The comparison between baselines and PAG is illustrated in Table 1. First, PAG consistently outperforms other generative retrieval baselines across the three datasets. Compared with RIPOR that also uses knowledge distillation, PAG achieves 15.6% relative improvement on MRR@10 in MSMARCO Dev set, which emphasizes the importance of adding set-based DocIDs constructed by the lexical approach for computing relevance scores. Notice that, the beam size used in PAG is 100, which is 10 times smaller than the beam size of 1000 used in RIPOR. This can reduce the query latency significantly and implies the effectiveness of employing planning-ahead constrained beam search. Second, compared with the dense retrieval methods in our experiments, PAG also consistently shows better performance while using less index memory. For instance, PAG improves MRR@10 by 11.9% over TAS-B, and in the TREC-DL 20 set, it leads to 2.2% improvement on NDCG@10. It is also important to note that PAG use 8 times less index memory compared with the dense retrieval models.

4.1.5 ***Ablation Studies.*** We conduct a thorough ablation study on MSMARCO dataset to investigate the impact of each component in PAG. The results are shown in Table 2.

Beginning with Row 1, we observe that eliminating $s^{\text{simul}}(\cdot)$ Equation (6) from the final calculation of relevance scores would lead to 10.3% and 17.7% MRR@10 and Recall@10 degradation in the MSMARCO Dev set, respectively. This is because the simultaneous relevance scoring function is based on set-based DocIDs

which are constructed from a lexical approach. It is capable of providing complementary relevance signals to the sequential DocIDs aiming at capturing semantic information. Row 2 also reflects the importance of combining lexical and semantic information for retrieval effectiveness. We find that solely using $s^{\text{simul}}(\cdot)$ for retrieval would result in a 27% and 9.1% decrease in terms of MRR@10 and Recall@10, respectively.

Based on Rows 3 and 4, we can infer that the more effective M^{seq} can ultimately boost the effectiveness of the unified generative retrieval model. For instance, applying the seq2seq pre-training and multi-objective loss for fine-tuning can lead to a 1.0% and 1.3% enhancement on MRR@10 respectively. Seq2seq pre-training applies the DocID prediction task over the whole corpus, which can mitigate the issue of distribution shift between training and evaluation sets. The multi-objective loss is designed for sequentially decoding algorithms, such as beam search, used in generative retrieval models, which can reduce the risk of making the relevant prefixes discarded from the beam, especially in early decoding steps.

The results in Rows 5 and 6 imply that using M^{set} or M^{seq} alone would result in retrieval performance degradation. For example, only use M^{set} and M^{seq} reduce the MRR@10 by 19.6% and 13.6%, respectively. Interestingly, when we linearly interpolate the lexical and semantic scores together for the final relevance score, we observe significant performance gain where the results are shown in Row 7. The simple post-hoc combination leads to 11.8% and 6.2% improvements over M^{set} and M^{seq} on the MSMARCO Dev set. This again emphasizes the effectiveness of combining the lexical and semantic information for retrieval. That being said, the retrieval performance shown in Row 7 still lags behind the original model, which implies the effectiveness of integrating M^{set} and M^{seq} into a unified model with joint optimization. We observe that by joint modeling, the model can achieve 6.9% and 13.0% enhancement on MSMARCO@10 and Recall@10 respectively.

Finally, as evidenced by Rows 8 and 9, PAG achieves superior results over M^{sp} and M^{ds}, improving the MRR@10 by 1.9% and 5.5% respectively. Notably, this performance enhancement is accompanied by a significant reduction in memory usage - PAG requires ×10.8 and ×7.7 less memory compared to M^{sp} and M^{ds}. The efficient memory utilization underscores the effectiveness of using set-based and sequential DocIDs in compressing the original embedding information near-losslessly, and demonstrates the benefit of joint modeling them.

4.2 Analysis and Discussion

4.2.1 ***The impact of beam size and number of selected subwords.*** The selection of beam size and number of selected subwords m would affect the effectiveness and efficiency of the model. The large beam size can reduce the risk of pruning the relevant prefixes out of the beam and the large U might improve the model expressiveness by extracting more lexical information from the document. However, it comes with the trade-off of increasing the query latency and index memory. To quantify that, we report the results of different settings of m and beam size k in Table 3. First, when k is fixed, an increase of m can show a trade-off in retrieval performance and resource utilization. For instance, at $k = 100$, the MRR@10 with $m = 16$ is 0.355, while it increase to 0.390 when $m = 128$, yielding 9.9% enhancement. However, this gain comes at the cost of ×3.58 and ×1.55 increase in the index memory and query latency respectively. Second, at a fixed value of m, employing a larger k can enhance retrieval effectiveness. For instance, at $m = 64$, increasing k from 10 to 100 can improve MRR@10 by 1.6% albeit at the expense of ×5.68 increase in query latency. It is noteworthy that performance degradation is less pronounced than that observed in RIPOR, as illustrated in Figure 1. This relative robustness can be attributed to the use of planning-ahead constrained beam search in PAG, which re-weights each prefix by a pre-stored document-level relevant score. Thus it would facilitate more efficient retrieval without significantly compromising performance. Finally, we can observe that using simultaneous decoding is much more efficient than autoregressive generation for retrieval, hence does not introduce too much overhead over the original beam search algorithm.

4.2.2 ***Comparison between RIPOR and PAG.*** We train a new RIPOR model with the same configurations of DocIDs used in PAG ($L = 8$, $V = 2048$). Other than the original document-level relevant labels, we also construct the prefix-level relevant labels where the prefix of a relevant document is also relevant to a given

Table 2: Ablation study results on MSMARCO Dev. Superscript $^{\triangledown}$ denotes significantly lower performance compared to PAG (t-test with Bonferroni correction, p_value < 0.01). itp. stands for interpolation.

	MRR@10	Recall@10	Index Mem.(GB)
PAG	.385	.670	3.27
1. w/o adding $s^{\text{simul}}(\cdot)$	.349$^{\triangledown}$	.614$^{\triangledown}$	0.50
2. Only $s^{\text{simul}}(\cdot)$ for search	.303$^{\triangledown}$	.569$^{\triangledown}$	2.77
3. w/o seq2seq pre-training	.381$^{\triangledown}$	.660$^{\triangledown}$	3.27
4. w/o multi-obj. learning	.380$^{\triangledown}$	.663$^{\triangledown}$	3.27
5. Only M^{set}	.322$^{\triangledown}$	.606$^{\triangledown}$	2.77
6. Only M^{seq}	.339$^{\triangledown}$	.566$^{\triangledown}$	0.50
7. Linear itp. of M^{set} and M^{seq}	.360$^{\triangledown}$	.593$^{\triangledown}$	3.27
8. Only M^{sp}	.378$^{\triangledown}$	.667$^{\triangledown}$	35.28
9. Only M^{ds}	.365$^{\triangledown}$	.641$^{\triangledown}$	25.30

Table 3: The effectiveness and efficiency comparison with different m and k on MSMARCO Dev. The experiment is conducted on an 80GB A100 GPU. Simul. D. and Seq. D. stands for simultaneous and sequential decoding respectively. QL represents query latency (ms / query).

m, k	MRR@10	Recall@10	Index Mem.(GB)	Simul. D. QL	Seq. D. QL
16, 10	.342	.577	1.30	20	44
32, 10	.367	.626	1.94	22	44
64, 10	.379	.641	3.27	25	44
128, 10	.386	.645	5.96	31	44
16, 100	.355	.620	1.30	20	250
32, 100	.372	.652	1.94	22	250
64, 100	.385	.670	3.27	25	250
128, 100	.390	.664	5.96	31	250

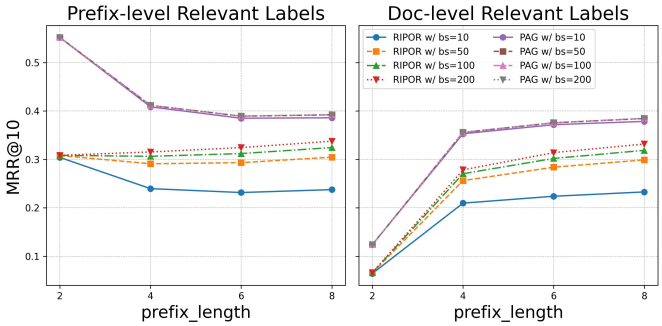

Figure 3: Results on MS MARCO Dev with different beam sizes on prefix-level and document-level labels.

query. We conduct the performance comparison between RIPOR and PAG in terms of different prefix lengths and beam sizes in the MSMARCO Dev set, which is illustrated in Figure 3. Initially, we find that PAG not only consistently outperforms in both prefix-level and document-level relevance labeling but also shows robustness against variations in beam size (bs) compared with RIPOR. Specifically, when beam size is set to 10, there is only a marginal reduction in MRR@10 compared to larger beam sizes. Moreover, the MRR@10 for those larger beam sizes (50, 100, 200) have nearly identical performance across different prefix lengths. In contrast, we can always observe notable performance improvements with increased beam sizes in RIPOR. These findings indicate the effectiveness of incorporating the document-level scores for constrained beam search.

Additionally, our analysis reveals distinct patterns in prefix-level relevance labeling for RIPOR and PAG, as illustrated in the left sub-figure. In PAG, MRR@10 values have a monotonic decrease with longer prefix lengths. In contrast, RIPOR displays the opposite trends in MRR@10 except when the beam size is 10. It is because the use of simultaneous decoding for obtaining the document-level scores in PAG ensures high Recall@10 rates for early-stage relevant prefix retrieval. In the top 10 ranking list, making the relevant prefixes with higher ranks is easier in shorter lengths since shorter prefixes tend to be more coarse-grained and dissimilar to the relevant ones. This characteristic results in the decline of MRR@10 values as prefix lengths increase. Conversely, RIPOR lacks a similar early-stage Recall@10 performance. It relies on a larger beam size (>10) and longer, more expressive prefixes to achieve higher MRR@10 values.

4.2.3 ***The impact of combining set-based and sequential DocIDs.*** We can alternatively view PAG as a retrieve-then-rerank model, in which it first retrieves the top promising documents using set-based DocIDs and then gradually refines the retrieved document scores based on the newly generated next token by sequential DocIDs. In Table 2, we already show that only using set-based DocIDs for search can achieve .303 in terms of MRR@10, while in this section, we qualitatively demonstrate the retrieval effectiveness by investigating the quality of set-based DocIDs. For this, we randomly sample 20 queries from the combining sets of TREC 19/20, and select all the relevant documents to these sampled queries. For each document d with the set-based DocID $t^d = \{t_1^d, \dots t_m^d\}$, we obtain corresponding T5-embedding (learned in PAG) for each token and

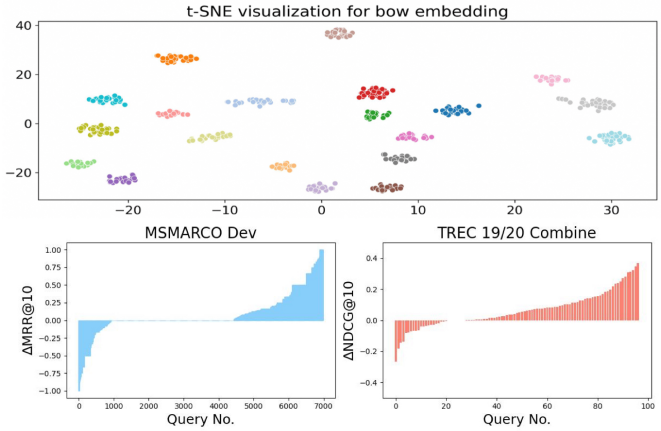

Figure 4: *Above Figure*: clusters of relevant documents to 20 queries sampled from TREC-19/20, and the color indicates the query ID. *Below Figures*: Δ MRR@10 on MSMARCO Dev and Δ NDCG@10 on TREC-19/20 between simultaneous+autoregressive decoding (PAG) and simultaneous decoding alone for each query.

denoted them as $\{\mathbf{e}_1^d, \dots \mathbf{e}_m^d\}$, then each document embedding can be computed as $\mathbf{e}_d = \frac{1}{m}\sum_{i=1}^{m} \mathbf{e}_i^d$. We apply the T-SNE [65] to the document embeddings for dimension reduction and visualize them in the above figure of Figure 4. According to the figure, we observe that almost all documents with the same label (relevant to the same query) can be clustered together. This demonstrates that the set-based DocID can extract useful tokens from each document's content for capturing relevance-based information and enhancing retrieval performance.

To better understand the re-ranking effect of combining set-based and sequential DocIDs for computing relevance scores. We compare the performance difference between the original PAG and the variant that only uses set-based DocIDs for retrieval. We report the ΔMRR@10 in the MSMARCO and the ΔNDCG@10 in the TREC-DL 19&20 sets respectively for each query, and the results are illustrated in the below sub-figures of Figure 4. For the sake of space, we merged the query sets of TREC-DL 19&20 in this experiment. We observe from the plots that the majority of queries can either benefit or at least not be harmed by joint scoring. We acknowledge that not all queries benefit from the joint scoring. Specifically, approximately 1000 out of 6980 queries in MSMARCO Dev and 20 out of 97 queries in TREC-DL 19&20 notice a declined in performance. This could be attributed to combined scoring potentially introducing biases that negatively affect queries that are better suited to lexical matching, typically captured by set-based DocIDs.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel optimization and decoding framework for generative retrieval. We introduced simultaneous decoding for efficient document-level score estimation and used it to guide autoregressive decoding. We create the set-based DocIDs under the bag-of-words assumption and sequential DocIDs based on the

relevance-based document representations to support simultaneous and autoregressive decodings, respectively. We additionally introduced a three-stage training pipeline for gradual adaptation of the model to joint decoding. Our experiments demonstrated substantial improvements compared to state-of-the-art generative retrieval baselines, in terms of both efficiency and effectiveness. Looking ahead, we aim to further optimize the model's efficiency and scale it up to billion-scale datasets. We also look forward to integrating the framework into other knowledge-intensive tasks, such as open-domain question answering.

6 ACKNOWLEDGMENT

This work was supported in part by the Center for Intelligent Information Retrieval, in part by Lowe's, and in part by an Amazon Research Award, Fall 2022 CFP. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Artem Babenko and Victor S. Lempitsky. 2012. The Inverted Multi-Index. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 1247–1260. https://api.semanticscholar.org/CorpusID:15445563

[2] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. 2018. Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors. *ArXiv* abs/1802.02422. https://api.semanticscholar.org/CorpusID:3602418

[3] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. *ArXiv* abs/2204.10628. https://api.semanticscholar.org/CorpusID:248366293

[4] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *ArXiv* abs/1611.09268. https://api.semanticscholar.org/CorpusID:1289517

[5] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. *ArXiv* abs/2010.00904. https://api.semanticscholar.org/CorpusID:222125277

[6] Jiangui Chen, Ruqing Zhang, J. Guo, M. de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual Learning for Generative Retrieval over Dynamic Corpora. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.* https://api.semanticscholar.org/CorpusID:261277063

[7] Jiangui Chen, Ruqing Zhang, J. Guo, M. de Rijke, Y. Liu, Yixing Fan, and Xueqi Cheng. 2023. A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:258418300

[8] Jiangui Chen, Ruqing Zhang, J. Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative Evidence Retrieval for Fact Verification. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:248118757

[9] Jiangui Chen, Ruqing Zhang, J. Guo, Y. Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* https://api.semanticscholar.org/CorpusID:251594672

[10] Xilun Chen, Kushal Lakhotia, Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stanislav Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen tau Yih. 2021. Salient Phrase Aware Dense Retrieval: Can a Dense Retriever Imitate a Sparse One? *ArXiv* abs/2110.06918. https://api.semanticscholar.org/CorpusID:238744204

[11] Yongjian Chen, Tao Guan, and Cheng Wang. 2010. Approximate Nearest Neighbor Search by Residual Vector Quantization. *Sensors (Basel, Switzerland)* 10, 11259 – 11273. https://api.semanticscholar.org/CorpusID:33774240

[12] David R. Cheriton. 2019. From doc2query to docTTTTTquery. https://api.semanticscholar.org/CorpusID:208612557

[13] Eun-Kyu Choi, Sunkyung Lee, Minjin Choi, Hyeseon Ko, Young-In Song, and Jongwuk Lee. 2022. SpaDE: Improving Sparse Representations using a Dual Document Encoder for First-stage Retrieval. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* https://api.semanticscholar.org/CorpusID:252212320

[14] Hyung Won Chung et al. 2022. Scaling Instruction-Finetuned Language Models. *ArXiv* abs/2210.11416. https://api.semanticscholar.org/CorpusID:253018554

[15] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the TREC 2019 Deep Learning Track. In *TREC*.

[16] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2021. Overview of the TREC 2020 Deep Learning Track. *ArXiv* abs/2102.07662. https://api.semanticscholar.org/CorpusID:212737158

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics.* https://api.semanticscholar.org/CorpusID:52967399

[18] Thibault Formal, C. Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *ArXiv* abs/2109.10086. https://api.semanticscholar.org/CorpusID:237581550

[19] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:235792467

[20] Luyu Gao and Jamie Callan. 2021. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. *ArXiv* abs/2108.05540. https://api.semanticscholar.org/CorpusID:236987190

[21] Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *ArXiv* abs/1211.3711. https://api.semanticscholar.org/CorpusID:17194112

[22] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.

[23] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *ArXiv* abs/2010.02666. https://api.semanticscholar.org/CorpusID:222141041

[24] Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. FiD-Light: Efficient and Effective Retrieval-Augmented Text Generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1437–1447. https://doi.org/10.1145/3539618.3591687

[25] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:233231706

[26] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: Re-rank with source coding. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 861–864. https://api.semanticscholar.org/CorpusID:10271065

[27] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 535–547. https://api.semanticscholar.org/CorpusID:926364

[28] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3, 535–547.

[29] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Conference on Empirical Methods in Natural Language Processing.* https://api.semanticscholar.org/CorpusID:215737187

[30] O. Khattab and Matei A. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:216553223

[31] O. Khattab and Matei A. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://api.semanticscholar.org/CorpusID:216553223

[32] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980. https://api.semanticscholar.org/CorpusID:6628106

[33] 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 V. Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7, 453–466. https://api.semanticscholar.org/CorpusID:86611921

[34] Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. 2022. Nonparametric Decoding for Generative Retrieval. In *Annual Meeting of the Association for Computational Linguistics.* https://api.semanticscholar.org/CorpusID:258959550

[35] Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *ArXiv* abs/2005.11401. https:

//api.semanticscholar.org/CorpusID:218869575
[36] Yongqing Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Learning to Rank in Generative Retrieval. *ArXiv* abs/2306.15222. https://api.semanticscholar.org/CorpusID:259262395
[37] Yongqing Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview Identifiers Enhanced Generative Retrieval. In *Annual Meeting of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:258947148
[38] Jimmy J. Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *ArXiv* abs/2106.14807. https://api.semanticscholar.org/CorpusID:235658149
[39] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *ArXiv* abs/2010.11386. https://api.semanticscholar.org/CorpusID:225041183
[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692. https://api.semanticscholar.org/CorpusID:198953378
[41] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. B-PROP: Bootstrapped Pre-training with Representative Words Prediction for Ad-hoc Retrieval. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[42] Yury Malkov and Dmitry A. Yashunin. 2016. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 824–836. https://api.semanticscholar.org/CorpusID:8915893
[43] Sanket Vaibhav Mehta, Jai 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. *ArXiv* abs/2212.09744. https://api.semanticscholar.org/CorpusID:254854290
[44] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *ArXiv* abs/1901.04085. https://api.semanticscholar.org/CorpusID:58004692
[45] Long Ouyang et al. 2022. Training language models to follow instructions with human feedback. *ArXiv* abs/2203.02155. https://api.semanticscholar.org/CorpusID:246426909
[46] Biswajit Paria, Chih-Kuan Yeh, Ning Xu, Barnabás Póczos, Pradeep Ravikumar, and Ian En-Hsu Yen. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. *ArXiv* abs/2004.05665. https://api.semanticscholar.org/CorpusID:211107043
[47] Biswajit Paria, Chih-Kuan Yeh, Ning Xu, Barnabás Póczos, Pradeep Ravikumar, and Ian En-Hsu Yen. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. *ArXiv* abs/2004.05665. https://api.semanticscholar.org/CorpusID:211107043
[48] Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://api.semanticscholar.org/CorpusID:14103653
[49] Ronak Pradeep, Kai Hui, Jai Gupta, Ádám Dániel Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Q. Tran. 2023. How Does Generative Retrieval Scale to Millions of Passages? *ArXiv* abs/2305.11841. https://api.semanticscholar.org/CorpusID:258822999
[50] Ronak Pradeep, Rodrigo Nogueira, and Jimmy J. Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *ArXiv* abs/2101.05667. https://api.semanticscholar.org/CorpusID:231603106
[51] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *North American Chapter of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:231815627
[52] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 140:1–140:67. https://api.semanticscholar.org/CorpusID:204838007
[53] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Hieu Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. *ArXiv* abs/2305.05065. https://api.semanticscholar.org/CorpusID:258564854
[54] Ruiyang Ren, Wayne Xin Zhao, J. Liu, Huaqin Wu, Ji rong Wen, and Haifeng Wang. 2023. TOME: A Two-stage Approach for Model-based Retrieval. *ArXiv* abs/2305.11161. https://api.semanticscholar.org/CorpusID:258762633
[55] Stephen E Robertson. 1977. The probability ranking principle in IR. *Journal of documentation* 33, 4, 294–304.
[56] Stephen E. Robertson and Steve Walker. 1997. On relevance weights with little relevance information. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://api.semanticscholar.org/CorpusID:16829071
[57] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 333–389. https://api.semanticscholar.org/CorpusID:207178704
[58] Alireza Salemi, Juan Altmayer Pizzorno, and Hamed Zamani. 2023. A Symmetric Dual Encoding Dense Retrieval Framework for Knowledge-Intensive Visual Question Answering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 110–120. https://doi.org/10.1145/3539618.3591629
[59] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When Large Language Models Meet Personalization. arXiv:2304.11406 [cs.CL]
[60] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 11–21.
[61] Felix Stahlberg and Bill Byrne. 2019. On NMT Search Errors and Model Errors: Cat Got Your Tongue? *ArXiv* abs/1908.10090. https://api.semanticscholar.org/CorpusID:201646223
[62] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, M. de Rijke, and Zhaochun Ren. 2023. Learning to Tokenize for Generative Retrieval. *ArXiv* abs/2304.04171. https://api.semanticscholar.org/CorpusID:258048596
[63] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *ArXiv* abs/1409.3215. https://api.semanticscholar.org/CorpusID:7961699
[64] 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. 2022. Transformer Memory as a Differentiable Search Index. *ArXiv* abs/2202.06991. https://api.semanticscholar.org/CorpusID:246863488
[65] Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 2579–2605. https://api.semanticscholar.org/CorpusID:5855042
[66] Shuai Wang, Shengyao Zhuang, and G. Zuccon. 2021. BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval. *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. https://api.semanticscholar.org/CorpusID:237366133
[67] Yujing Wang, Ying Hou, Hong Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A Neural Corpus Indexer for Document Retrieval. *ArXiv* abs/2206.02743. https://api.semanticscholar.org/CorpusID:249395549
[68] Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. 2023. NOVO: Learnable and Interpretable Document Identifiers for Model-Based IR. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. https://api.semanticscholar.org/CorpusID:264350310
[69] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Conference on Empirical Methods in Natural Language Processing*. https://api.semanticscholar.org/CorpusID:252917569
[70] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *ArXiv* abs/2007.00808. https://api.semanticscholar.org/CorpusID:220302524
[71] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and J. Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. https://api.semanticscholar.org/CorpusID:52229883
[72] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-Enhanced Machine Learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2875–2886. https://doi.org/10.1145/3477495.3531722
[73] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and Effective Generative Information Retrieval. In *Proceedings of the 2024 Web Conference* (Singapore, Singapore) *(WWW '24)*. (to appear).
[74] Hansi Zeng, Hamed Zamani, and Vishwa Vinay. 2022. Curriculum Learning for Dense Retrieval Distillation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://api.semanticscholar.org/CorpusID:248426770
[75] ChengXiang Zhai and John D. Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *International Conference on Information and Knowledge Management*. https://api.semanticscholar.org/CorpusID:1043470
[76] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, M. Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://api.semanticscholar.org/CorpusID:

233289894
[77] Kai Zhang, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. 2022. LED: Lexicon-Enlightened Dense Retriever for Large-Scale Retrieval. *Proceedings of the ACM Web Conference 2023*. https://api.semanticscholar.org/CorpusID:251903309
[78] Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, and Zhao Cao. 2023. Term-Sets Can Be Strong Document Identifiers For Auto-Regressive Search Engines. *ArXiv* abs/2305.13859. https://api.semanticscholar.org/CorpusID:258841428
[79] R. Zhou and Eric A. Hansen. 2005. Beam-Stack Search: Integrating Backtracking with Beam Search. In *International Conference on Automated Planning and Scheduling*. https://api.semanticscholar.org/CorpusID:11314454
[80] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Yu Wu, Peitian Zhang, and Ji rong Wen. 2022. Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer. *ArXiv* abs/2208.09257. https://api.semanticscholar.org/CorpusID:251710261
[81] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. https://api.semanticscholar.org/CorpusID:252993059
[82] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, G. Zuccon, and Daxin Jiang. 2022. Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation. *ArXiv* abs/2206.10128. https://api.semanticscholar.org/CorpusID:249890267