Generative Retrieval via Term Set Generation

Peitian Zhang*
namespace.pt@gmail.com
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Zheng Liu*†
zhengliu1026@gmail.com
Beijing Academy of Artificial Intelligence
Beijing, China

Yujia Zhou
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Zhicheng Dou†
dou@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Fangchao Liu
Zhao Cao
Huawei Poisson Lab
Beijing, China

ABSTRACT

Recently, generative retrieval has emerged as a promising alternative to the traditional retrieval paradigms. It assigns each document a unique identifier, known as the DocID, and employs a generative model to directly generate the relevant DocID for the input query. A common choice for the DocID is one or several natural language sequences, e.g. the title, synthetic queries, or n-grams, so that the pre-trained knowledge of the generative model can be effectively utilized. However, a sequence is generated token by token, where only the most likely candidates are kept and the rest are pruned at each decoding step, thus, retrieval fails if any token within the relevant DocID is falsely pruned. What's worse, during decoding, the model can only perceive preceding tokens in the DocID while being blind to subsequent ones, hence is prone to make such errors. To address this problem, we present a novel framework for generative retrieval, dubbed **T**erm-**S**et **Gen**eration (TSGen). Instead of sequences, we use a set of terms as the DocID. The terms are selected based on learned weights from relevance signals, so that they concisely summarize the document's semantics and distinguish it from others. On top of the term-set DocID, we propose a permutation-invariant decoding algorithm, with which the term set can be generated in any permutation yet will always lead to the corresponding document. Remarkably, TSGen perceives all valid terms rather than only the preceding ones at each decoding step. Given the constant decoding space, it can make more reliable decisions due to the broader perspective. TSGen is also resilient to errors: the relevant DocID will not be falsely pruned as long as the decoded term belongs to it. Moreover, TSGen can explore the optimal decoding permutation of the term set on its own, which further improves the likelihood of generating the relevant DocID. Lastly, we design an iterative optimization procedure to incentivize the model to generate the relevant term set in its favorable permutation. We conduct extensive experiments on popular benchmarks of generative retrieval, which validate the effectiveness, the generalizability, the scalability, and the efficiency of TSGen.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**.

KEYWORDS

Generative Retrieval, Document Identification, Term Set Generation

ACM Reference Format:
Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative Retrieval via Term Set Generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657797

1 INTRODUCTION

Document retrieval, standing as the most representative form of information retrieval, is fundamentally important to real-world applications like web search, question answering, advertising, and recommendation [17, 56]. Nowadays, they are also regarded as a critical tool for the augmentation of large language models (LLMs), where external information can be introduced to enhance their knowledge and capability [29, 41, 52]. Typical document retrieval methodologies call for two basic modules: representation and indexing. For example, a sparse retrieval system uses lexical representations and an inverted index [23, 36], while a dense retrieval system is based on dense embeddings and an ANN index [25, 51].

Recently, generative retrieval [2, 39, 43, 59] emerges as a promising alternative to the traditional sparse or dense retrieval [28]. Specifically, it assigns each document a unique identifier, known as the DocID, and employs a generative model to directly generate the DocID of the relevant document for the input query. Compared with traditional retrieval methods, generative retrieval is end-to-end differentiable: the entire retrieval pipeline including representation

*Co-first author.
†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3657797

and indexing is encapsulated into a generative model and hence can be optimized by the Seq2Seq learning [28, 39]. Defining appropriate DocID is fundamental to effective generative retrieval. A large body of research utilizes one or several natural language sequences as the DocID, such as the title [2], n-grams [1, 20, 21], or synthetic queries [38, 60]. This is because modern generative language models (e.g., T5 [34], Llama [40]) are pre-trained on natural language, thus, employing a natural language based DocID may seamlessly inherit the pre-trained knowledge of these models [1]. Besides, they are also more interpretable and generalizable than other alternatives like naive ID [32] and clustering based ID [35, 43].

Unlike natural language generation, which allows flexible paraphrasing for the same meaning, generative retrieval poses a unique challenge: it must exactly generate the relevant DocID sequence. The sequence is generated token by token. At each decoding step, the top-B (B: the beam size) likely candidates are kept and the rest are pruned; consequently, the relevant DocID cannot be generated if any token within it is falsely pruned. Intuitively, this **false pruning problem** is common and challenging to mitigate: the generative model can only perceive preceding tokens in the DocID (tokens next to the current decoding step), without access to the information in subsequent ones, hence is prone to make mistakes. This limits the overall retrieval quality.

In Figure 1(A), we introduce an example to better illustrate the false pruning problem with sequence DocID. Without loss of generality, we use one query (*"what is the sodium level in white bread"*) from the dev set of MSMARCO Dataset [30] and consider three candidate documents where Doc1 is the relevant one. Following previous studies [38], we use one synthetic query as the DocID, and SE-DSI as the generative model. We can observe that the relevant DocID is falsely pruned in the 4-th decoding step. Specifically, the likelihood of its prefix is lower than that of the second DocID, i.e. $p(\text{what,is,the,calorie} \mid Q) < p(\text{what,is,the,sodium} \mid Q)$, even though the suffix of the second DocID (*"egg white"*) is not relevant to the query at all. As stated, this problem stems from the requirement for the exact generation of the relevant DocID sequence. While existing approaches, such as increasing the beam size or maintaining multiple DocID sequences per document [1, 21], provide some relief, they do not systematically solve the issue.

In this work, we present a novel framework, dubbed **T**erm **S**et **Gen**eration (TSGen), to address the above problem. Instead of one or several sequences, TSGen uses a set of terms as the DocID. These terms are selected based on the learned weights from relevance signals. As a result, they not only concisely summarize the document's semantics, but distinguish the document from others.

On top of the term-set DocID, we propose a permutation-invariant decoding algorithm, with which the terms can be generated in any permutation yet will always lead to the corresponding document. Akin to the widely used constrained beam search [2, 59], the proposed algorithm guarantees the validity of the generated DocID. In other words, the generated term set always corresponds to a valid document in the corpus. In addition, it introduces three key advantages. (1) At each decoding step, TSGen perceives all valid terms rather than only the preceding ones, thereby acquiring full information about the DocID. Each term itself usually comprises only one token, thus the decoding space remains unchanged, and TSGen can make more reliable decisions given its broader perspective. (2) TSGen is resilient to decoding errors. In contrast to sequence-based generation, the relevant DocID will not be falsely pruned as long as the decoded terms belong to it. (3) TSGen can explore the optimal permutation on its own to generate the term set. It may permute the terms in one way for certain queries while in another way for others to maximize the likelihood. This flexibility makes it easier to generate the relevant DocID and improves the retrieval accuracy. The algorithm is implemented with an inverted index and is competitive in efficiency as elaborated in Section 3.2.

Back to our example in Figure 1(B), the three documents are paired with their term-set DocID. At the i-th decoding step, the model decodes the next term from the valid decoding space (V_i). Initially, this space encompasses the union of all terms, and it gradually narrows down during the decoding process to ensure validity. We can observe that the model sequentially decodes *"sodium, white, bread, calorie"* from V_i, successfully generating the relevant DocID. At each decoding step, the model is exposed to all valid terms in all documents, so it can thoroughly examine all information in the DocID before making a decision. Besides, the model is allowed to use a different permutation than the highlighted one, for example, it may decode *"bread"* or *"calorie"* other than *"white"* in the second generation step, which is just a different permutation of the term set so the relevant DocID can still be generated. This error resilience applies to the pruning of negative documents as well. In the second and third steps, Doc3 and Doc2 are pruned when all the terms of their DocID are considered irrelevant. Moreover, the model can adjust the permutation of the term set for different queries. For instance, it may generate *"calorie white bread sodium"* in response to the query *"how much calories is in the white bread"*.

Finally, we devise an iterative optimization procedure to guide TSGen to generate the relevant term set in its favorable permutation. We employ the standard Seq2Seq loss while periodically update the permutation of the term set based on the latest model snapshot. The model converges in a state where it can successfully generate the term set in the permutation that it reckons the most probable.

The contributions of our work are summarized as follows:
(1) We propose TSGen, a novel generative retrieval framework where a set of terms is used as the DocID, and any permutations of these terms lead to the corresponding document. It systematically addresses the false pruning problem of the widely used sequence-based DocID designs in existing generative retrieval methods.
(2) We devise tailored techniques for TSGen, including the learned term selection, the permutation-invariant decoding algorithm, and the iterative optimization procedure.
(3) We conduct extensive experiments on popular generative retrieval benchmarks. The results validate the effectiveness, generalizability, scalability, and efficiency of TSGen.

2 RELATED WORK

2.1 Traditional Document Retrieval

Document retrieval has been extensively studied for a long time. There are mainly two threads of research towards traditional document retrieval. One of them is *sparse retrieval*, which stores the bag-of-words (BOW) representation of a document and estimates relevance based on the weights of overlapping terms, exemplified

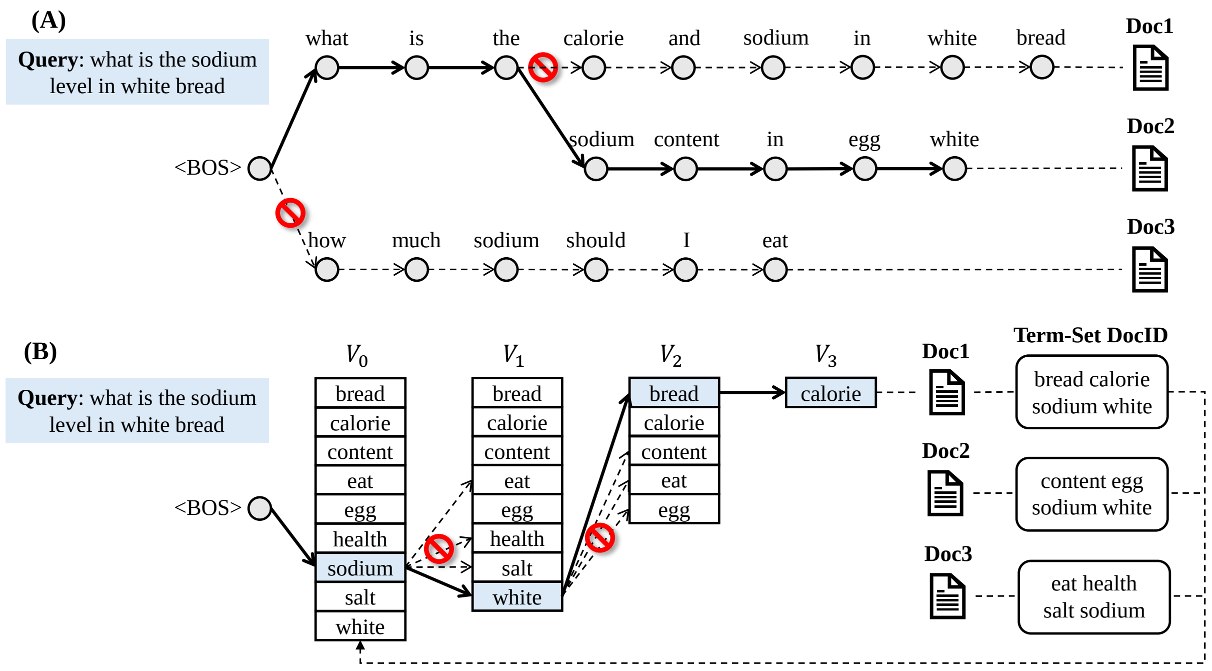

Figure 1: (1) An example of the false pruning problem stemming from the requirement of exactly generating the sequence DocID. The relevant document (Doc1) is falsely pruned during decoding due to the limited perspective of the model and the low resilience to errors. (2) TSGen successfully retrieves the relevant document in the same context. Based on the term-set DocID and the permutation-invariant decoding, TSGen enjoys a broader perspective at each decoding step to make more reliable decisions and is resilient to errors.

by BM25 algorithm [36]. Another one is *dense retrieval*, which fine-tunes a pretrained language model to encode the document and the query in a low-dimensional semantic space, and model the retrieval problem as the maximum inner product search (MIPS) [3, 17, 45, 53]. When applied with tailored training techniques, including mining hard negatives [33, 46, 49], distillation [24, 50], and retrieval-oriented pretraining [42, 44], dense retrieval has achieved state-of-the-art effectiveness. In the meantime, continuous efforts have been made to enhance the sparse retrieval with PLMs to learn contextualized term weights [8, 10, 11, 23, 26]. All the aforementioned retrieval methods rely on standalone indexes to work, such as the inverted index [61] and the ANN index [16]. These separate indexes cannot be optimized together with the encoding model.

2.2 Generative Retrieval

Recently, generative retrieval is proposed as an alternative to the traditional index-then-retrieve pipeline [39], where a generative model (usually a Seq2Seq language model like T5 [34]) is fine-tuned to directly map the input query to its relevant document. In this formulation, the document is represented by its identifier, namely the DocID, a much shorter signature than the document itself and hence easier to generate. Meanwhile, the index is replaced with the parametric memory of the language model and hence can be optimized end-to-end. As a result, the DocID becomes one of the most decisive factors for generative retrieval: the model must generate the exact same DocID for the targeted document, and the ranking of the document is determined by the generation likelihood of its DocID. Based on the choices of DocIDs, current works can be roughly partitioned into four groups:

(1) The atomic ID based methods [39, 58–60] use an integer or a tokenized integer. They lack semantics since there's no relation between the document's content and its DocID. Consequently, it requires the model to remember all DocIDs during training.

(2) The clustering based methods [4, 27, 39, 43] use a series of cluster center indices. Prevalent approaches include hierarchical KMeans [39, 43], product quantization [4, 59], and residual quantization [35, 48, 55]. Despite their improvements, these DocIDs are not directly compatible with pre-trained generative model because their embeddings in the decoding table are newly introduced.

(3) The learned DocID based methods [37, 47] progressively learn the DocID based on document reconstruction or relevance signals. Similar to the clustering based ones, these DocIDs must be optimized apart from the pre-trained weight of the generative model.

(4) The natural language based methods [1, 2, 5, 6, 9, 19, 38, 59] use one or several natural language sequences, e.g., the URL, the title, n-grams, or synthetic queries. These methods may directly inherit the enhanced capability from the pre-trained generative model. Besides, they are more interpretable than other alternatives. However, we argue that the natural language sequences naturally suffer from the false pruning problem. As a remedy, our work adopts a set of terms as DocID, systematically mitigating the false pruning problem.

2.3 Set Generation

The concept of set generation originates from the Set Prediction Networks [54], which optimizes permutation-invariant losses to learn to decode a set of vectors based on the input vector. There have also been successful implementations to generate a set of texts using transformers [14]. However, instead of generating independent elements one by one, we seek to generate a term set in its optimal permutation. This is somehow relevant to the autoregressive graph generation, which generates a graph by sequentially adding nodes and edges [12, 13, 15, 22]. Multiple autoregressive node-edge sequences (intractable) may lead to the same graph, yet only a few of them are believed to be easier for model learning since they may follow some straightforward patterns. By comparison, our task is even more challenging as there are no given nodes and edges. In other words, we have to construct the term set from scratch and then determine its generation order. In this paper, we propose the learned term selection to extract discriminative terms, and the iterative optimization to guide the model to generate the relevant term set in its favorable permutation.

3 METHODOLOGY

In generative retrieval, each document is associated with its identifier, namely the DocID. A generative model is employed to generate the DocID of the relevant document for the query. The relevance between the query Q and the document D can be expressed as:

$$\mathrm{Rel}(Q, D) = \Psi\left(\left\{\prod\nolimits_{i=1}^{|I|} p(I_i \mid I_{<i}, Q; \Theta) : I \in \mathcal{I}(D)\right\}\right), \quad (1)$$

where $\mathcal{I}(D)$ denotes all the DocIDs assigned to D and Ψ is a family of functions to aggregate multiple scalars into one. Usually, there is only one DocID for each document, so $\Psi(\cdot)$ is the identity function $\mathbb{1}(\cdot)$. SEAL [1] and its followers [5, 21] assign multiple sequences to each document as the DocID. In that case, $\Psi(\cdot)$ is implemented with an intersective scoring function based on heuristics.

Computing $\mathrm{Rel}(Q, D)$ with all documents in the corpus is computationally prohibitive, thus, existing methods resort to constrained beam search for efficient generation. In brief, at each decoding step, only B candidates are selected for a further generation while others are all pruned out based on the likelihood of their prefixes (i.e. $\sum_{i=1}^{*} \log p(I_i \mid I_{<i}, Q; \Theta)$). However, the generative model is likely to falsely prune the relevant docid by mistake since it can only perceive preceding tokens without access to subsequent ones.

In this work, we propose the Term Set Generation (TSGen) framework to address this problem. Specifically, instead of one or several sequences, TSGen uses a set of terms as the DocID, denoted as $\mathcal{T}(D) = \{\tau_1, \ldots, \tau_N\}$. These terms are automatically selected to concisely summarize the information in the document and distinguish it from others. On top of the term-set DocID, we revolutionize the constrained beam search with a permutation-invariant decoding algorithm, with which any permutation of the term set always leads to the corresponding document[1]. Meanwhile, the model can explore the optimal permutation of the term set on its own. This translates to an aggregation function of $\Psi(\cdot) \leftarrow \max(\cdot)$. Finally, the model is incentivized to generate the term set in its favorable permutation through an iterative optimization procedure.

In the remaining part of this section, we will elaborate on the learned term selection (Section 3.1), the permutation equivariant decoding (Section 3.2), and the iterative optimization (Section 3.3).

[1]We explicitly guarantees the uniqueness of the term sets as described in § 3.1, §4.1.

3.1 Learned Term Selection

The quality of the terms is critical to the retrieval quality of TSGen. They should be informative because otherwise they do not bring about information and may interfere with the decoding process. They should also be discriminative so that the model can better learn the relation between the query and the term sets. In this place, inspired by existing sparse retrieval methods [10, 23], we employ a term selection module to learn from the query-document relevance and perform accurate term selection.

Specifically, each document D is first partitioned into a list of terms: $[t_1^D, \ldots, t_{|D|}^D]$. Then, the terms are encoded into hidden states by a BERT. Finally, the weight of each term is obtained by pooling its hidden state into a scalar through an MLP layer (two linear transformations plus one dropout). Formally,

$$\boldsymbol{e}_i^D = \mathrm{BERT}([t_1^D, \ldots, t_{|D|}^D])[i], \quad w_i^D = \mathrm{MLP}(\boldsymbol{e}_i^D). \quad (2)$$

The same operation is performed on the query Q as well, resulting in the term weights $\{w_j^Q\}_{j=1}^{|Q|}$. Both BERT and MLP are learned with contrastive learning based on the relevance signals $\mathcal{A} = \{\langle Q, D^+, \{D_i^-\}_{m=1}^M \rangle\}$ where D^+ is the relevant document to Q, and $\{D_i^-\}_{m=1}^M$ are M hard negatives mined from BM25:

$$\mathcal{L} = \min\left(-\log \frac{\exp(s(Q, D^+))}{\exp(s(Q, D^+)) + \sum_{m=1}^{M} \exp(s(Q, D_m^-))}\right), \quad (3)$$

$$s(Q, D) = \sum\nolimits_{t_j^Q = t_i^D} w_j^Q w_i^D \quad (4)$$

where $t_j^Q = t_i^D$ indicates t_j^Q and t_i^D are the same term.

Therefore, the module learns to assign high weights to those terms that not only represent the whole documents, but also distinguish the relevant document from the irrelevant ones. Based on these well-learned weights, we select the top-N terms for each document to formulate the term-set DocID:

$$\mathcal{T}(D) = \left\{t_i^D : w_i^D \in \text{top-}N\left(\{w_i^D\}_{i=1}^{|D|}\right)\right\}. \quad (5)$$

In practice, we choose the smallest value of N while ensuring the uniqueness of the term set. For example, $N = 12$ is already enough for a small-scale corpus like NQ100K.

3.2 Permutation-Invariant Decoding

Constrained beam search is the most prevalent way for generative retrieval to efficiently generate valid DocIDs. It creates a prefix tree (a.k.a. trie) encompassing all DocIDs in the corpus and constrains the decoding process along a root-to-leaf trajectory on this trie. In TSGen, any permutation of the term set DocID leads to the corresponding document. Naively, this yields $N!$ possible trajectories on the trie, which is intractable. Instead, we propose a permutation-invariant decoding algorithm to replace the constrained beam search. It is characterized by three properties. Firstly, it guarantees that the generated DocID is *valid*. In other words, the generation result is precisely one permutation of an existing term set in the corpus. Secondly, it allows the model to explore the *optimal* permutation of the term set, so that the model can first generate the terms it reckons more probable, then make fine-grained decisions for others. Thirdly, it is implemented with an inverted index based structure and hence is competitive in *efficiency*. The overall algorithm is depicted in Algotithm 1. We'll go through the algorithm and give the necessary explanation in the following.

Algorithm 1 Permutation-invariant decoding.

```
1: ▷Create an empty inverted index that maps each term to documents containing it.
2: $\Phi \leftarrow$ Inverted_Index()
3: for $D \in \mathcal{D}$ do
4:     for $\tau \in \mathcal{T}(D)$ do
5:         $\Phi(\tau)$.add($^*D$)
6:     end for
7: end for
8: ▷For each beam...
9: $B \leftarrow$ Beam Size
10: for $1 \le b \le B$ do
11:     ▷Initialize the generated term with "<BOS>".
12:     $X_{b,0} \leftarrow$ [<BOS>]
13:     ▷Initialize the pointers to valid documents.
14:     $Y_{b,0} \leftarrow \{^*D : D \in \mathcal{D}\}$
15:     ▷Initialize the valid decoding space.
16:     $V_{b,0} \leftarrow \cup\{\mathcal{T}(D) : D \in \mathcal{D}\}$
17: end for
18: ▷Start generation...
19: for $1 \le t \le N$ do
20:     ▷Beam search over the valid decoding space.
21:     $\forall 1 \le b \le B, \quad \mathcal{X}_{b,t} \leftarrow X_{b,t-1} \times V_{b,t}$
22:     $X_{1,t}, \dots, X_{B,t} \leftarrow \arg\max_{X_{b,t} \in \mathcal{X}_{b,t}} \sum_{b=1}^{B} \log p(X_{b,t} \mid X_{b,<t}; Q; \Theta)$
            $s.t. X_{i,t} \neq X_{j,t}$
23:     ▷Reorder valid documents to align with current beams.
24:     $\forall 1 \le b \le B, \quad Y_{b,t-1} \leftarrow \text{reorder}(Y_{b,t-1})$
25:     ▷Get documents containing the newly decoded term.
26:     $\forall 1 \le b \le B, \quad x_{b,t} \leftarrow X_{b,t}[-1], \quad y_{b,t} \leftarrow \Phi(x_{b,t})$
27:     ▷Narrow down the valid documents.
28:     $\forall 1 \le b \le B, \quad Y_{b,t} \leftarrow Y_{b,t-1} \cap y_{b,t}$
29:     ▷Update the valid decoding space.
30:     $\forall 1 \le b \le B, \quad V_{b,t+1} \leftarrow \{\mathcal{T}(D) : D \in Y_{b,t}\} \setminus X_{b,t}$
31: end for
32: ▷Return the retrieved documents.
33: return $\{Y_{b,N}\}_{b=1}^{B}$
```

The beam size is denoted as B. We keep track of the following variables: (1) A list of terms that have been generated until the t-th generation step in b-th beam, denoted as $X_{b,t}$. (2) Pointers to documents whose DocIDs contain all the generated terms until the t-th step in b-th beam, namely the *valid documents*, denoted as $Y_{b,t}$. (3) The valid decoding space for next-term generation in the t-th step and b-th beam, denoted as $V_{b,t}$.

In line 1-7, we build an inverted index to map each term to the document whose DocID contains that term. The document is represented by its pointer for efficiency. In line 10-17, we initialize the above three variables. Notably, the initial valid decoding space is all existing terms. From line 19, we start to loop over the N generation steps (since each DocID consists of N terms). We first get the valid decoding space by conducting Cartesian product between the already generated terms and all valid terms. Then we perform a standard beam search over the valid term space, selecting the top B hypotheses based on likelihood. Therefore, the generative model is enabled to select any terms in the valid decoding space, which translates to exploring the optimal permutation of the term set on its own. In line 23, we reorder $Y_{b,t-1}$ so that it aligns with $X_{b,t}$ in terms of the beam index. Then we look up the inverted index to obtain the documents containing the newly decoded term $X_{b,t}[-1]$, and update the valid documents by intersection in line 25. Next, we can get the valid decoding space of the next generation step, which is the union of terms in the valid DocIDs minus the already decoded terms. At the end of the loop, the valid B documents are returned, which correspond to the generated term sets.

The permutation-equivariant decoding draws the same outcome as the intractable solution based on trie: guaranteeing the validity of the generated term-set DocID while allowing the model to explore the optimal permutation of the term set. Besides, it is competitive in efficiency as verified in Table 5.

3.3 Iterative Optimization

The generative model to remember the DocIDs in its parameters, meanwhile, it must learn the relevance between the query and the DocIDs. This calls for fine-tuning the model with annotated or synthetic data. Existing works usually employ one or several sequences as DocID, which are defined in advance of the optimization of the model. Therefore, the Seq2Seq learning can be directly applied, which maps the input query to the relevant DocID.

In TSGen, a set of terms is used as the DocID, and all permutations of the term set lead to the corresponding document. This implies $N!$ possible sequences for a single document, while enumerating them is intractable. A straightforward solution is to randomly sample one or several sequences from these candidates and perform the standard Seq2Seq learning. However, as mentioned in Section 1, different permutations of the term set may significantly differ in the generation likelihood. Thus, a randomly sampled permutation may not be favorable to the model, which means its generation likelihood is low. Optimizing the model towards generating in such a permutation may adversely influence its memorization and generalization capability. Instead, we incentivize the model to follow the permutation that it deems the most probable, which is in line with the goal of our permutation-equivariant decoding algorithm. This is done with our iterative optimization procedure.

In T-th iteration (T starts from 1), we sample the favorable permutation of the term-set DocID from the generative model itself to serve as the learning objective. Specifically, given the model's latest snapshot Θ^{T-1} and the query, we let the model generate the permutation it deems the most probable in a similar way to Algorithm 1. Formally, denote the beam size as B', terms generated until the t-th step in the b-th beam as $Z_{b,t}$, we perform the following operation on all beams $1 \le b \le B'$ and loop over N steps:

$$V'_{b,t} \leftarrow \mathcal{T}(D) \setminus Z_{b,t-1}, \quad \mathcal{Z}_{b,t} \leftarrow Z_{b,t-1} \times V'_{b,t},$$
$$Z_{1,t}, \dots, Z_{B',t} \leftarrow \arg\max_{Z_{b,t} \in \mathcal{Z}_{b,t}} \sum_{b=1}^{B'} \log p(Z_{b,t} \mid Z_{b,<t}; Q; \Theta^{T-1}; u),$$
$$s.t. Z_{i,t} \neq Z_{j,t}. \tag{6}$$

Note that we add a temperature u to the probability so that slightly diversified permutations can be generated. As a result, we obtain the permutation with high likelihood $\{Z_{i,N}\}_{i=1}^{B'}$ according to model Θ^{T-1}. Finally, we use query Q as the source, and the result of the first beam ($Z_{1,N}$) as the target of the Seq2Seq learning in this iteration. After convergence, the model is updated as Θ^{T}.

There are two remaining issues. One is the initial permutation of the term set. Among different options, e.g., purely randomized permutation, or sampling from the pre-trained generative model like T5 and GPT, we empirically find that permuting the terms by their estimated importance in our term selection module brings forth the best performance. The other one is about the convergence. Although the sampled permutation is always changing, we keep track of the model's retrieval accuracy on a hold-out validation set.

4 EXPERIMENTS

Experiments are conducted to verify the effectiveness (Section 4.2), the generalizability (Section 4.3), the scalability (Section 4.4), and the efficiency (Section 4.5) of TSGen. Meanwhile, we analyse the individual contribution of our technical designs (Section 4.6).

4.1 Settings

• **Datasets.** We leverage two popular datasets that are widely used by previous works on auto-regressive search engines. One is the NQ320K dataset [39, 43], which is curated from Natural Questions [18], containing 109k documents, 320k training queries, 7830 testing queries. The other one is MS300K dataset [37, 59], which is curated from MSMARCO [30], containing 320k documents, 360k training queries, and 772 testing queries. To investigate the performance of TSGen when scaled to a larger corpus, we also leverage the MSMARCO Passage dataset, which contains 8.8M passages, 500k training queries, and 6980 testing queries.
• **Metrics.** We measure the retrieval quality at the top-K cut-off by MRR@K (M@K) and Recall@K (R@K), which focus on the perspective of ranking and recall, respectively.
• **Implementations.** We leverage T5-base [34] as our backbone. We select $N = 12$ terms on NQ320K and MS300K, and $N = 16$ terms on MSMARCO Passage. We treat each single word, separated by space, as one term. The term usually contains a single token. Sometimes it contains multiple tokens. We append a "," to the last token, which indicates the termination of the term. We can apply this mechanism to other granularities such as n-grams, i.e. a set of n-grams can be used as the DocID. We leave it to future work. Following the existing works [43, 59], we leverage fine-tuned DocT5 [7] to generate synthetic training queries. For each document, we use 10 synthetic queries on NQ320K and 3 synthetic queries on MS300K. When generation, the beam size B is set to 100 throughout the experiments, which is also the same as in previous works. As for iterative optimization, we perform 2 iterations by default, and we sample the favorable permutation with a temperature $u = 3$. We've uploaded our implementations to https://github.com/namespace-Pt/TSGen.
• **Baselines.** We introduce a diverse collection of generative retrieval baselines with different DocID settings: DSI [39]: using hierarchical KMeans IDs; NCI [43]: using layerwise distinguished KMeans IDs. GENRE [2]: using titles; Ultron [59]: using urls; SEAL [1]: using all n-grams in the document; MINDER [21]: using titles, synthetic queries, and n-grams; GenRet [37]: using learned DocIDs. We also compare several traditional retrieval baselines, including the sparse retrieval methods BM25 [36], UniCOIL [23], and SPLADEv2 [10]; the dense retrieval methods DPR [17], ANCE [46], and GTR [31]. Among them, UniCOIL and SPLADEv2 are both based on learned term weights and are closely related to TSGen.

4.2 Main Analysis

We main evaluation results of TSGen on NQ320K and MS300K are reported in Table 1. We have the following observations.

Firstly, **TSGen demonstrates a notable advantage over the strongest generative retrieval baseline on both datasets**. For example, on NQ320K, it outperforms GenRet by +2% on M@100; on MS300K, it achieves a relative improvements of +16% over Ultron on MRR@100. This verifies the high retrieval quality of TSGen.

Secondly, when compared with natural langauge sequence based DocIDs, e.g. GENRE with titles, Ultron with URLS, and SEAL/MINDER with n-grams, **TSGen achieves better Recall**. This is as expected since the sequence based DocID is more likely to cause the false pruning of the relevant DocID, while TSGen can largely mitigate the problem. TSGen also significantly outperforms them on MRR. This is because TSGen can follow the optimal permutation to generate the term-set DocID, which yields a higher likelihood for the relevant DocID so that it is ranked higher.

Thirdly, **the advantage of TSGen still holds in comparison with traditional retrieval approaches**: It achieves superior ranking performance than all traditional retrieval baselines in terms of MRR and Recall at small cutoffs. This again validates the effectiveness of TSGen. In our ablation studies, we'll show that the term-set DocID and the permutation-invariant decoding are the main contributors to such advantages.

Fourthly, despite the above advantages, we may observe that the sparse/dense retrieval baselines may outperform TSGen in terms of R@100 on MS300K. This is in line with the findings in [57]. We conjecture it's because the "false pruning" problem, though largely mitigated by TSGen, still occurs. However, now it does not orient from the DocID and the generation process, instead, it orients from the generative model. That's to say, the model believes *all terms* in the term-set DocID are irrelevant to the query and hence chooses others for decoding. How to avoid such errors and learn a better model to make accurate decisions would be left to future work.

4.3 Generalizability Analysis

The retrieval capability of the generative model can be decomposed into two dimensions. 1) The **memorization**, which memorizes the DocID and the query-doc relevance that have been seen during training. 2) The **generalization**, which generalizes the learned knowledge to new DocIDs that have not been observed in training. Many existing works have been shown to have limited generalization capability [37]. They usually rely on extensive data augmentation (e.g. generate many synthetic queries for each document in the corpus) to alleviate this problem. However, the problem remains especially when new documents are updated to the corpus [4, 27].

In this experiment, we evaluate the memorization and generalization capability of TSGen. We partition the corpus into two halves for both NQ320K and MS300K, with training queries preserved for 50% of the documents (Seen), and with training queries removed for the other 50% of the documents (Unseen). The statistics of the

Table 1: Evaluation of the retrieval effectiveness on NQ320K and MS300K. † denotes the results on NQ320K are copied from [37]. * indicates significant improvements over the best generative retrieval baseline with p-value < 0.05.

		NQ320K					MS300K				
Category	**Method**	**M@10**	**M@100**	**R@1**	**R@10**	**R@100**	**M@10**	**M@100**	**R@1**	**R@10**	**R@100**
Sparse	BM25†	–	0.211	0.151	0.325	0.505	0.313	0.325	0.196	0.591	0.861
	UniCOIL	0.710	0.713	0.619	0.862	0.926	0.425	0.435	0.284	0.766	0.951
	SPLADEv2	0.726	0.731	0.624	0.873	**0.954**	0.443	0.452	0.328	0.779	0.956
Dense	DPR†	–	0.599	0.502	0.777	0.909	0.424	0.433	0.271	0.764	0.948
	ANCE†	–	0.602	0.502	0.785	0.914	0.451	0.455	0.299	0.785	0.953
	GTR-Base†	–	0.662	0.560	0.844	0.937	0.484	0.485	0.332	**0.793**	**0.960**
Generative	DSI	0.594	0.598	0.533	0.715	0.816	0.339	0.346	0.257	0.538	0.692
	NCI†	–	0.731	0.659	0.852	0.924	0.408	0.417	0.301	0.643	0.851
	GENRE	0.653	0.656	0.591	0.756	0.814	0.361	0.368	0.266	0.579	0.751
	Ultron	0.726	0.729	0.654	0.854	0.911	0.432	0.437	0.304	0.676	0.794
	SEAL†	–	0.677	0.599	0.812	0.909	0.393	0.402	0.259	0.686	0.899
	MINDER	0.709	0.713	0.627	0.869	0.933	0.431	0.435	0.289	0.728	0.916
	GenRet†	–	0.759	0.681	0.888	0.952	–	–	–	–	–
	TSGen	**0.771***	**0.774**	**0.708**	**0.889**	0.948	**0.502***	**0.505***	**0.384***	0.781*	0.931*

Table 2: Evaluation of the memorization and generalization capability on seen and unseen documents, respectively.

	NQ320K						MS300K					
	Seen (50%)		Unseen (50%)		All (100%)		Seen (50%)		Unseen (50%)		All (100%)	
Method	**M@10**	**R@10**	**M@10**	**R@10**	**M@10**	**R@10**	**M@10**	**R@10**	**M@10**	**R@10**	**M@10**	**R@10**
NCI	0.771	0.882	0.050	0.143	0.410	0.549	0.408	0.643	0.034	0.082	0.260	0.412
GENRE	0.763	0.869	0.138	0.187	0.448	0.558	0.361	0.579	0.150	0.312	0.196	0.411
Ultron	0.782	0.891	0.300	0.383	0.471	0.570	0.432	0.676	0.197	0.246	0.313	0.492
MINDER	0.774	**0.907**	0.303	0.415	0.488	0.639	0.431	0.728	0.285	0.433	0.335	0.569
TSGen	**0.809**	0.900	**0.466**	**0.654**	**0.552**	**0.700**	**0.484**	**0.766**	**0.390**	**0.588**	**0.391**	**0.642**

Table 3: Dataset statistics for the evaluation of the memorization and generalization capability of TSGen.

Dataset	Shard	#Docs	#Train	#Validation	#Test
NQ320K	Seen	59,739	173,447	3,000	3,915
	Unseen	50,000	–	–	3,915
MS300K	Seen	314,461	359,000	564	772
	Unseen	289,790	–	–	439

curated datasets are reported in Table 3. Given the above setting, the generative model is prevented from memorizing any information about the unseen documents during the training stage. We pick out several strong baselines for comparison, each of which uses a distinct DocID schema. The results are reported in Table 2.

According to the results, TSGen marginally outperforms the baselines on the "seen" half; nevertheless, its advantage is significantly magnified on the "unseen" half. This indicates the superior generalizability of TSGen. The reasons are two-fold. 1) TSGen uses a set of terms as the DocID, which is naturally generalizable across documents thanks to the potential overlap of terms. In other words, the unseen documents may share the same term as seen documents. Thus the model learned on the seen half has some prior knowledge of the DocIDs in the unseen half. Though GENRE, Ultron, and MINDER may also have such prior knowledge thanks thanks to the potentially similar semantics between DocIDs, they may fail to take advantage of it due to the false pruning problem. 2) TSGen adopts the permutation-invariant decoding, which enables the model to explore its favorable permutation of the term set, and hence facilitates the generation of the relevant DocID.

Table 4: Evaluation of the scalability on MSMARCO Passage [30]. † denotes the result copied from [32]. ‡ denotes the result copied from [21].

Model	#Training Queries	M@10
BM25	–	0.187
DPR	0.5M	**0.314**
DSI+DocT5×1	8.8M	0.075
DSI+DocT5×40†	352M	0.133
MINDER‡	7.2M	0.186
TSGen	8.8M	0.195

4.4 Scalability Analysis

Scalability is a crucial challenge for generative retrieval and is of wide interest to the community [32]. In this experiment, We evaluate TSGen on the entire MSMARCO Passage dataset, which

Table 5: Evaluation of the efficiency on NQ320K.

Method	Memory (MB)	Query Latency (s) beam size = 10	Query Latency (s) beam size = 100
DSI	12	0.03	0.21
NCI	12	0.03	0.21
GENRE	27	0.05	0.47
Ultron	32	0.08	0.64
MINDER	210	0.32	3.14
TSGen	35	0.06	0.69

contains 8.8M passages. Following the setting in [32], we augment each document with M synthetic queries generated by a DocT5 [7] model, and only use the synthetic queries as the training data. Due to limited computation resources, we set $M = 1$ (the original work set $M = 40$), resulting in 8.8M training queries. We set the number of selected terms per passage (i.e. N) to 16. This results in 48,577 term-set collisions. We examine some cases and find most collisions are coming from different editions of the same page (the contents of two passages are almost identical). In this case, knowing that the computation cost grows as N increases, we do not further scale N. Instead, we keep uniqueness by applying simple heuristics given N fixed to 16. Concretely, if two documents' term sets collide, we replace one document's last selected term with another less weighted term in it to distinguish those two documents.

The results are reported in Table 4. First of all, TSGen is advantageous against DSI on such a massive corpus: when trained with the same amount of data (8.8M training queries), TSGen significantly improves the MRR of DSI. It even outperforms DSI scaled with 40 augmented queries, which consumes 40 times more training data and hence is much more expensive. Besides, TSGen improves the strong baseline MINDER by 5% when trained with a similar amount of queries. These observations validate the scalability of TSGen. Lastly, generative retrieval methods still lag far behind the performance of dense retrieval on such a large corpus. This has been a widely-known issue of generative retrieval methods. Several concurrent works point out that utilizing contrastive learning [20, 48] in training may improve the effectiveness on a large corpus, which may be combined with TSGen for further improvements.

4.5 Efficiency Analysis

The running efficiency is evaluated in Table 5. Particularly, we measure the memory consumption for hosting the DocIDs of the entire corpus; we also measure the time cost (query latency) with different beam sizes. DSI and NCI enjoy the smallest memory usage and the lowest query latency thanks to their short DocID (only 10 integers). GENRE and Ultron require more space and are a little slower than DSI, because their DocID sequences are longer. Our method, TSGen, leverages an inverted index to perform permutation-invariant decoding. It achieves similar efficiency as Ultron. MINDER employs an FM index, which consumes much more memory and is also slower than all other approaches.

4.6 Ablation Studies

The ablation studies are performed for each influential factor in TSGen based on NQ320K dataset as Table 6.

Table 6: Ablation studies on NQ320K. The default settings of TSGen are marked with *.

Factor	Setting	M@10	R@10	R@100
DocID	Sequence	0.749	0.864	0.921
	Term-Set*	**0.771**	**0.889**	**0.948**
Term Selection	Random	0.628	0.739	0.811
	Title	0.743	0.856	0.915
	Learned*	**0.771**	**0.889**	**0.948**
Optimization	Non-Iterative	0.751	0.868	0.932
	Iterative*	**0.771**	**0.889**	**0.948**
Term Number	8	0.760	0.879	0.940
	16	**0.771**	**0.889**	0.947
	12*	**0.771**	**0.889**	**0.948**
Initial Permutation	Random	0.728	0.857	0.931
	Likelihood	0.716	0.847	0.915
	Weight*	**0.771**	**0.889**	**0.948**
Synthetic Queries (SQ)	Ultron w.o. SQ	0.670	0.779	0.845
	NCI w.o. SQ	–	0.679	0.909
	TSGen w.o. SQ	0.728	0.846	0.925
	TSGen*	**0.771**	**0.889**	**0.948**
Model Scale	DSI large	0.613	0.733	0.835
	NCI large	–	0.885	0.945
	GENRE large	0.663	0.770	0.828
	SEAL large	–	0.812	0.909
	TSGen large	**0.779**	**0.896**	**0.954**
Beam Size	10	0.770	0.876	–
	200	**0.771**	**0.892**	**0.951**
	100*	**0.771**	0.889	0.948

• **DocID.** We compare the proposed term-set DocID with the sequence based one. For the latter, the terms are ordered as a sequence by their estimated weights (empirically more competitive than other sequence orders). It can be observed that the retrieval quality of term-set DocID is notably superior to that of the sequence DocID. As discussed, the term-set DocID together with the permutation-invariant decoding effectively mitigates the false pruning problem, which often happens on the sequence based DocID.

• **Term Selection.** We compare our learned term selection with two alternatives: Random, the randomly selected terms from the document; Title: terms within the title. Firstly, there are huge differences between different term selection strategies, which verifies the importance of term selection. Secondly, although directly making use of title is a strong baseline (also a common practice in many works [2, 9, 59]), our learned selection strategy is more effective. Specifically, the term weights learned from relevance capture the relationship between queries and documents. Thus, terms selected based on these weights can distinguish the document from others, which are suitable to serve as the DocID.

• **Term Number.** We compare three settings of the number of selected terms for each document i.e. N. It can be observed that selecting less terms for each document degrades the ranking performance more than the recall. This can be attributed to the fact that the identifier collisions are more likely to happen, and hence TSGen does not know how to rank documents if they share the same DocID, yet including them all in the retrieval results can roughly maintain the the Recall@100 performance. On the other hand, selecting more terms for each document cannot further improve the retrieval quality. Therefore, our selecting protocol uses the smallest N while keeping discrimination, i.e. 12 on NQ320K.

• **Optimization.** We compare our iterative optimization with its non-iterative variant where the DocID's permutation is fixed as its initialization. Note that other settings (e.g. permutation-invariant decoding) are kept the same. It can be observed that our proposed optimization approach indeed contributes to retrieval quality. Such an advantage is easy to comprehend, considering that the training objective (the permutation of the term-set DocID) can be iteratively updated to keep consistent with the goal of our permutation-invariant decoding in the testing stage.

• **Initial Permutation.** We compare three approaches for initializing the permutation in the first iteration. 1) Random: the selected terms are randomly permuted; 2) Likelihood: the selected terms are permuted by the generation likelihood of the pre-trained T5; 3) Weight: the selected terms are permuted by their estimated weight in the term selection module. We can observe that the initialization turns out to be another critical factor for TSGen: the importance-based method is notably stronger than the other two baselines. This is probably because the importance-based initialization is easier to generate and better reflects the query-document relationship.

• **Synthetic Queries.** Augmenting synthetic queries to is a widely used strategy to improve generative retrieval [27, 43, 59]. It is also helpful for TSGen. Specifically, TSGen's retrieval quality is substantially improved on top of query generation. Besides, the relative improvement of TSGen is mainly from the proposed term-set DocID and its generation workflow, rather than the extra data augmentation. When query generation is disabled, TSGen maintains its advantage over other baselines.

• **Model Scale.** The scaling-up of the backbone generative model is another common approach to enhance generative retrieval. In our experiment, empirical improvements are also observed when we switch to a T5-large backbone. Meanwhile, it maintains the advantage when other baselines are scaled up as well.

• **Beam Size.** We default to use 100 as the beam size following previous works, and juxtapose the other two different settings. It can be observed that the Recall@10 drops when decreasing the beam size to 10. This is as expected since a smaller beam size keeps fewer candidates in each generation step and hence terms in the relevant DocID may not be included. Another interesting observation is that the MRR@10 almost stays the same when decreasing or increasing the beam size. This is because TSGen can generate the relevant DocID in its favorable permutation, thus assigning a high likelihood to it and ranking it at the top, regardless of changing the beam size.

5 CASE STUDY

In Table 7, we show an example on MS300K to qualitatively evaluate TSGen. Compared with the title or URL, the term-set DocID effectively summarize the information of the document. Besides, TSGen generates the same term set in its favorable permutation for different queries. It puts the term that matches with the query at the front so that the likelihood is maximized.

Table 7: Case study of TSGen on MS300K. The displayed document is the target of both query 1 and query 2. The term set DocID is ordered by learned term weights. Terms matching with the query are highlighted in yellow.

	Document
Title:	What Foods Not to Eat When Having High Creatinine
URL:	http://www.kidney-treatment.org/creatinine/162.html
Body:	... Creatinine is a breakdown product of creatine. Since kidneys are responsible for discharging creatinine in blood, so when kidneys are injured for some reason, creatinine level in blood increases. Therefore, high creatinine level indicates there are lots of wastes in blood. What foods not to eat when having high creatinine? 1. Adjust protein intake...
Term Set:	kidney, foods, eat, creatinine, creatine, blood, high, level, snack, salted, wastes, potassium
Query 1:	What food should not be eaten in kidney failure
Permut. 1:	kidney, foods, eat, level, high, salted, snack, creatinine, creatine, blood, potassium, wastes
Query 2:	foods to raise my creatinine level
Permut. 2:	creatinine, creatine, foods, high, level, eat, kedney, blood, mean, snack, wastes, salted, potassium

6 CONCLUSION AND FUTURE WORK

In this work, we present TSGen, a novel framework for generative retrieval to mitigate the false pruning problem stemming from the natural language sequence DocID. It employs a set of terms as the DocID instead of one or several sequences. On top of the term-set DocID, we propose the permutation-invariant decoding, with which any permutations of the term set will always lead to the corresponding document. We further devise an iterative optimization procedure to incentivize the model to generate the relevant term set in its favorable permutation. With comprehensive experiments, we empirically verify that TSGen is more effective, generalizable, and scalable than existing generative retrieval methods with competitive efficiency. In the future, we would like to explore other granularities to form the DocID, for example, the n-gram set. Besides, we may combine the recent contrastive learning techniques to further boost the performance on a large corpus.

ACKNOWLEDGEMENT

This work was supported by the Beijing Natural Science Foundation No. L233008, the 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.

REFERENCES

[1] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=Z4kZxAjg8Y

[2] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=5k8F6UU39V

[3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2309.07597 [cs.CL]

[4] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual Learning for Generative Retrieval over Dynamic Corpora. *CoRR* abs/2308.14968 (2023). https://doi.org/10.48550/arXiv.2308.14968 arXiv:2308.14968

[5] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2023. A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 1448–1457. https://doi.org/10.1145/3539618.3591631

[6] 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 *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 191–200. https://doi.org/10.1145/3511808.3557271

[7] David R. Cheriton. 2019. From doc2query to docTTTTTquery.

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

[9] Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. 2022. Multilingual Autoregressive Entity Linking. *Transactions of the Association for Computational Linguistics* 10 (2022), 274–290. https://doi.org/10.1162/tacl_a_00460

[10] 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). arXiv:2109.10086 https://arxiv.org/abs/2109.10086

[11] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2288–2292. https://doi.org/10.1145/3404835.3463098

[12] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. 2020. GraphGen: A Scalable Approach to Domain-agnostic Labeled Graph Generation. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 1253–1263. https://doi.org/10.1145/3366423.3380201

[13] Xu Han, Xiaohui Chen, Francisco J. R. Ruiz, and Li-Ping Liu. 2023. Fitting Autoregressive Graph Generative Models through Maximum Likelihood Estimation. *Journal of Machine Learning Research* 24, 97 (2023), 1–30. http://jmlr.org/papers/v24/22-0337.html

[14] Helia Hashemi, Hamed Zamani, and W. Bruce Croft. 2022. Stochastic Optimization of Text Set Generation for Learning Multiple Query Intent Representations. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 4003–4008. https://doi.org/10.1145/3511808.3557666

[15] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 2328–2337. http://proceedings.mlr.press/v80/jin18a.html

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

[17] 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[18] 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. https://doi.org/10.1162/tacl_a_00276

[19] Xiaoxi Li, Yujia Zhou, and Zhicheng Dou. 2024. UniGen: A Unified Generative Framework for Retrieval and Question Answering with Large Language Models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 8688–8696. https://doi.org/10.1609/AAAI.V38I8.28714

[20] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Learning to rank in generative retrieval. *arXiv preprint arXiv:2306.15222* (2023).

[21] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview Identifiers Enhanced Generative Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 6636–6648. https://doi.org/10.18653/V1/2023.ACL-LONG.366

[22] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 4257–4267. https://proceedings.neurips.cc/paper/2019/hash/d0921d442ee91b896ad95059d13df618-Abstract.html

[23] Jimmy Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *CoRR* abs/2106.14807 (2021). arXiv:2106.14807 https://arxiv.org/abs/2106.14807

[24] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng, Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, and Haifeng Wang. 2022. ERNIE-Search: Bridging Cross-Encoder with Dual-Encoder via Self On-the-fly Distillation for Dense Passage Retrieval. *CoRR* abs/2205.09153 (2022). https://doi.org/10.48550/arXiv.2205.09153 arXiv:2205.09153

[25] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[26] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 1723–1727. https://doi.org/10.1145/3404835.3463030

[27] 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). https://doi.org/10.48550/arXiv.2212.09744 arXiv:2212.09744

[28] 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. https://doi.org/10.1145/3476415.3476428

[29] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332 [cs.CL]

[30] 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)*, Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

[31] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Abrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large Dual Encoders Are Generalizable Retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 9844–9855. https://aclanthology.org/2022.emnlp-main.669

[32] Ronak Pradeep, Kai Hui, Jai Gupta, Adam D. Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Q. Tran. 2023. How Does Generative Retrieval Scale to Millions of Passages? arXiv:2305.11841 [cs.IR]

[33] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, 5835–5847. https://doi.org/10.18653/v1/2021.naacl-main.466

[34] 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. http://jmlr.org/papers/v21/20-074.html

[35] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. arXiv:2305.05065 [cs.IR]

[36] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[37] 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). https://doi.org/10.48550/ARXIV.2304.04171 arXiv:2304.04171

[38] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-Enhanced Differentiable Search Index Inspired by Learning Strategies. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 4904–4913. https://doi.org/10.1145/3580305.3599903

[39] 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). arXiv:2202.06991 https://arxiv.org/abs/2202.06991

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[41] Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, Anima Anandkumar, and Bryan Catanzaro. 2023. Shall We Pretrain Autoregressive Language Models with Retrieval? A Comprehensive Study. arXiv:2304.06762 [cs.CL]

[42] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. SimLM: Pre-training with Representation Bottleneck for Dense Passage Retrieval. *CoRR* abs/2207.02578 (2022). https://doi.org/10.48550/arXiv.2207.02578 arXiv:2207.02578

[43] 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). https://doi.org/10.48550/arXiv.2206.02743 arXiv:2206.02743

[44] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 538–548. https://aclanthology.org/2022.emnlp-main.35

[45] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[46] 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net. https://openreview.net/forum?id=zeFrfgyZln

[47] Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto Search Indexer for End-to-End Document Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 6955–6970. https://aclanthology.org/2023.findings-emnlp.464

[48] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2023. Scalable and Effective Generative Information Retrieval. *CoRR* abs/2311.09134 (2023). https://doi.org/10.48550/ARXIV.2311.09134 arXiv:2311.09134

[49] 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*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 1503–1512. https://doi.org/10.1145/3404835.3462880

[50] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2022. Adversarial Retriever-Ranker for Dense Text Retrieval. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.* OpenReview.net. https://openreview.net/forum?id=MR7XubKUFB

[51] Peitian Zhang, Zheng Liu, Shitao Xiao, Zhicheng Dou, and Jing Yao. 2023. Hybrid Inverted Index Is a Robust Accelerator for Dense Retrieval. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 1877–1888. https://aclanthology.org/2023.emnlp-main.116

[52] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve Anything To Augment Large Language Models. *CoRR* abs/2310.07554 (2023). https://doi.org/10.48550/ARXIV.2310.07554 arXiv:2310.07554

[53] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve Anything To Augment Large Language Models. arXiv:2310.07554 [cs.IR]

[54] Yan Zhang, Jonathon S. Hare, and Adam Prügel-Bennett. 2019. Deep Set Prediction Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 3207–3217. https://proceedings.neurips.cc/paper/2019/hash/6e79ed05baec2754e25b4eac73a332d2-Abstract.html

[55] Yidan Zhang, Ting Zhang, Dong Chen, Yujing Wang, Qi Chen, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Mao Yang, Qingmin Liao, and Baining Guo. 2023. IRGen: Generative Modeling for Image Retrieval. arXiv:2303.10126 [cs.CV]

[56] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2022. Dense Text Retrieval based on Pretrained Language Models: A Survey. *CoRR* abs/2211.14876 (2022). https://doi.org/10.48550/arXiv.2211.14876 arXiv:2211.14876

[57] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing Generative Retrieval with Reinforcement Learning from Relevance Feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 12481–12490. https://aclanthology.org/2023.emnlp-main.768

[58] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, and Ji-Rong Wen. 2023. DynamicRetriever: A Pre-trained Model-based IR System Without an Explicit Index. *Mach. Intell. Res.* 20, 2 (2023), 276–288. https://doi.org/10.1007/S11633-022-1373-9

[59] 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. arXiv:2208.09257 [cs.IR]

[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). https://doi.org/10.48550/arXiv.2206.10128 arXiv:2206.10128

[61] Justin Zobel, Alistair Moffat, and Kotagiri Ramamohanarao. 1998. Inverted Files Versus Signature Files for Text Indexing. *ACM Trans. Database Syst.* 23, 4 (1998), 453–490. https://doi.org/10.1145/296854.277632