How Does Generative Retrieval Scale to Millions of Passages?

Ronak Pradeep*†§, Kai Hui*, Jai Gupta, Adam D. Lelkes, Honglei Zhuang
Jimmy Lin§, Donald Metzler, Vinh Q. Tran*
rpradeep@uwaterloo.ca,{kaihuibj,vqtran}@google.com
Google Research, §University of Waterloo
USA, §Canada

ABSTRACT

Popularized by the Differentiable Search Index, the emerging paradigm of generative retrieval re-frames the classic information retrieval problem into a sequence-to-sequence modeling task, forgoing external indices and encoding an entire document corpus within a single Transformer. Although many different approaches have been proposed to improve the effectiveness of generative retrieval, they have only been evaluated on document corpora on the order of 100k in size. We conduct the first empirical study of generative retrieval techniques across various corpus scales, ultimately scaling up to the entire MS MARCO passage ranking task with a corpus of 8.8M passages and evaluating model sizes up to 11B parameters. We uncover several findings about scaling generative retrieval to millions of passages; notably, the central importance of using synthetic queries as document representations during indexing, the ineffectiveness of existing proposed architecture modifications when accounting for compute cost, and the limits of naively scaling model parameters with respect to retrieval performance. While we find that generative retrieval is competitive with state-of-the-art dual encoders on small corpora, scaling to millions of passages remains an important and unsolved challenge. We believe these findings will be valuable for the community to clarify the current state of generative retrieval, highlight the unique challenges, and inspire new research directions.

1 INTRODUCTION

For the last several years, dual encoders [4, 10, 16, 23] have dominated the landscape for first-stage information retrieval. They model relevance by mapping queries and documents into the same embedding space, optimized via contrastive learning [9, 12]. Dense embeddings are pre-computed for all documents in a corpus and stored in an external index. This allows for fast approximate nearest neighbor search [14, 39] to retrieve relevant documents. Cross-encoders based on large Transformer models [24, 26, 30] often function on top of these retrieved documents to further refine the top results.

Recently, the emerging paradigm of generative retrieval [6, 38] sought to replace this entire process with a single sequence-to-sequence Transformer model [37, 40], showing promising results against dual encoders given a sufficiently small corpus size. Since then, various techniques, such as [2, 3, 41, 43, 45], have aimed to improve the effectiveness of generative retrieval models, either with alternative document identifier formulations, architecture changes, or training objectives. Such work, however, has only evaluated generative retrieval over relatively small corpora on the order of 100k documents, such as Natural Questions [17], TriviaQA [15], or small subsets of the MS MARCO document ranking task [21]. Despite these research contributions, a number of open questions remain unanswered, including *how well* current generative retrieval techniques work on larger corpora and *which aspects* of generative retrieval models proposed so far are vital at scale.

In this paper, we conduct the first empirical study of generative retrieval techniques over the entire MS MARCO passage-level corpus, evaluating its effectiveness over 8.8M passages. We select popular approaches in recent works and evaluate them first on Natural Questions and TriviaQA to establish a definitive ablation of techniques in a controlled setup. Our experiments mainly focus on evaluating techniques proposed by Tay et al. [38], Zhuang et al. [45], and Wang et al. [41]. Namely, we ablate document identifier design: atomic, naive, semantic; document representation design: document tokens, ground truth queries, synthetic queries [27]; and model design: prefix-aware weight-adaptive decoding, constrained decoding, and consistency loss during training. At this small scale, we demonstrate state-of-the-art results for retrieval, generative and non-generative, over the NQ variant from [41], without the need for many proposed methods.

We then scale up the corpus size leveraging the MS MARCO passage ranking task, beginning with a subset of 100k passages before increasing the count to 1M and 8.8M passages (the entire set). Incrementally doing so allows us to establish which techniques remain effective as corpus size and difficulty scale. Finally, to explore the effect of model scaling on retrieval effectiveness on large corpora, we select a set of techniques with promising results at T5.1.1-Base scale [31] and modify the parameterization to consider up to 11B parameters. As the parameter distributions vary between methods, e.g., Atomic IDs cost embedding dimension times corpus size parameters, while Naive IDs do not cost anything beyond the core Transformer model, we aim to provide some insight into the trade-off of different parameter allocations on a large corpus.

While our experimental findings are nuanced, we summarize the main findings as follows:

(1) Of the methods considered, we find synthetic query generation to be the single most critical component as corpus

∗ Equal Contribution.
† Work completed while a student researcher at Google.

Gen-IR@SIGIR2023, July 27, 2023, Taipei, Taiwan

size grows. Defining the task of generative retrieval as solely mapping from synthetic queries to document identifiers is the most effective modeling strategy, with all other modeling strategies largely unnecessary.

(2) As corpus size increases, discussion of compute cost is crucial. Methods that implicitly increase model parameters perform better using the same T5 initialization. However, the quality improvements vanish as we scale up the naive approach to similar parameter sizes. Following [7], we note that the parameter count is not the entire story and provide more discussion regarding model comparisons and trade-offs in Section 6.2.

(3) Increasing the model size is necessary for improved generative retrieval effectiveness. However, somewhat surprisingly, for the best sequential IDs, effectiveness does not improve past a certain point – peaking at XL (3B) with a slightly worse score using XXL (11B) under fixed experimental settings. We find this counter-intuitive to the common conception of generative retrieval being limited by model capacity.

Our findings conclude that on the entire MS MARCO passage ranking task, simply scaling a model trained solely on synthetic queries to Naive ID generation demonstrates the best effectiveness of all techniques considered. On a small subset of 100k passages, a T5-Base model trained with this strategy achieves 82.4 MRR@10 (Section 6.1), competitive with GTR-Base [23] at 83.2 MRR@10. While on the 8.8M passages, a T5-XL model trained with this approach achieves only 26.7 MRR@10.

While the field of generative retrieval continues to evolve rapidly, it is clear that achieving competitive effectiveness against state-of-the-art dense retrieval models at scale remains an important and unsolved challenge. Our results suggest the need for continued research into generative retrieval and more fundamental advances to the paradigm before we are able to fully leverage the power of scaling up model parameters. We believe that our findings will help the research community better understand the current challenges faced when applying generative retrieval models to larger corpora and inspire new research in this direction.

2 RELATED WORK

Traditional retrieval models like BM25 [35] that rely on the lexical overlap, term frequency heuristics, and inverse document frequency, while reasonably strong on their own, tend to fail at matching documents that have minor word overlap but are semantically related. A popular solution is dual encoders [4, 10, 16], where a pretrained language model such as BERT [8] is used to compute low-dimensional dense representations instead of the high-dimensional sparse representations found in BM25. These dual encoder models are further trained on the target task to achieve improved effectiveness. Based on the success of T5 in various natural language understanding tasks, Ni et al. [22] proposes scaling up dual encoders by training T5-style pretrained language models with a two-stage contrastive learning approach on the Semantic Text Similarity (STS) tasks. The Generalizable T5 Retriever (GTR) [23] extends this idea to information retrieval. The most successful GTR models were pretrained on a large-scale question-answering dataset curated from the internet and fine-tuned on the MS MARCO Passage Ranking task [21].

Existing approaches often apply synthetic query generation to improve retrieval effectiveness. Nogueira et al. [27] first leveraged a vanilla sequence-to-sequence Transformer to train a model that can map passages to queries that it might be able to answer. Nogueira et al. [25], doc2query-T5 further improved the effectiveness of the traditional Transformer by leveraging a T5 model. Ma et al. [18] experimented with similar ideas and showed that query generation is effective across a wide range of corpora and task setups.

Prior to generative retrieval, sequence-to-sequence language models, like T5 [32], were shown to be effective for reranking tasks. In this setup, models assign scores to the top-k results from a first-stage retrieval method. One can then use these scores to rerank the documents. For example, monoT5 [24] was the first to leverage T5 as a pointwise reranker by training a model that takes the concatenation of the query and document as input and generates a relevance label. Hui et al. [13], Pradeep et al. [30], Zhuang et al. [44] have since improved the performance and efficiency of generation-based reranking. These approaches continue to demonstrate strong effectiveness [5, 28, 29].

Generative retrieval seeks to replace the entire information retrieval process with a single sequence-to-sequence model capable of mapping queries directly to relevant document identifiers [20]. Differentiable Search Indexes (DSI) [38] first demonstrated the potential of this paradigm, where T5 is used to parameterize an end-to-end search system, with the model parameters encoding all information about the corpus. See Section 3 for more information. DSI was shown to outperform a dual encoder baseline on Natural Questions dataset [17]. Zhuang et al. [45] explores the effectiveness of DSI and synthetic queries on a 100k passage subset of the MS MARCO passage ranking corpus and XOR QA [1]. Neural Corpus Indexer [41] builds on the success of DSI and introduces a combination of more input variants and architectural additions, some of which we describe and explore in this work. Many works have explored various document identifier designs, including document substring [2], metadata-based approaches [43, 46], and learned quantization [33, 36]. More recently, [3] proposes a distillation approach on top of DSI, learning from the rankings generated by dense retrieval using a multi-task training loss. However, none of these works have explored training or evaluating generative retrieval systems on corpora larger than O(100k) documents. Given that the generative retrieval paradigm has extended beyond traditional information retrieval into areas such as recommender systems [33] and vision [42], we believe our study on scaling will be crucial for an evergrowing community.

3 METHODS

In this section, we revisit the design details of the generative information retrieval method, using the Differentiable Search Index (DSI) [38] as the baseline. Then, we describe multiple techniques introduced in subsequent works that we aim to ablate and study in this work [41, 45].

3.1 Background

DSI [38] reformulates the retrieval task as a sequence-to-sequence (seq2seq) task, with queries as inputs and document identifiers (docids) relevant to the query as generation targets. The corpus,

namely the mapping between the document's content and its identifier, is encoded using the parameters of the LLM. DSI achieves this by leveraging two seq2seq tasks: indexing and retrieval. During training, the model learns to generate the docid given the document content (indexing task) or a relevant query (retrieval task). At inference time, the model processes a query and generates a ranked list of identifiers as retrieval results.

3.2 Inputs and Targets

In the framework discussed, DSI learns to encode the mapping between the long-form textual representation of a document and its identifier in its parameters while also learning to fetch the same identifier when it receives a relevant query as input.

Two crucial design choices are how documents are represented (i.e., the inputs in the indexing task) and how document identifiers (docids) are represented (i.e., the targets in both indexing and retrieval tasks). Two primary considerations are: (1) For document representations, it is prohibitive to encode long textual sequences with a Transformer [40]-based LLM, making it difficult to index full documents and (2) The naive identifiers taken from an existing dataset could be sub-optimal, for instance, due to their lack of semantic meaning. In this work, we consider different design choices for both these components.

3.2.1 Document Representations. One straightforward idea is to pick a text span from the document as a representation. DSI considers the first 64 tokens (FirstP) in each document, whereas Wang et al. [41] leverages ten randomly-selected chunks of 64 consecutive tokens, a technique they call Document As Query (DaQ). When working with Natural Questions and TriviaQA, which contain lengthy documents, we examine each variant separately and in combination. In the case of MS MARCO, which has short passages, FirstP and DaQ are essentially the same, assuming sufficient context length.

3.2.2 Synthetic Query Generation. For training the model for the retrieval task, the natural baseline uses existing labeled data, i.e., queries from the retrieval dataset as inputs and the docids labeled as relevant as targets (we will denote this as "Labeled Queries" in our tables). However, as argued in Zhuang et al. [45] and Wang et al. [41], there are two kinds of gaps between the index and retrieval tasks. First is the data distribution gap: queries for the retrieval task are short and request specific information, while the documents for the indexing task are long and convey information. Second is the coverage gap: the model is exposed to the entire corpus during the training of the indexing task, while only positive examples have associated queries in the retrieval task. The latter problem is exacerbated in the MS MARCO passage ranking task as only 550K passages have an associated query for training the retrieval task, while the indexing task has to learn to encode all 8.8M passages in the corpus. Their proposed method for mitigating this gap is by generating synthetic queries for each document using a query generation model such as docT5query [25]. The generative retrieval model is then trained to predict the docid given the corresponding synthetic queries. We can also think of these synthetic queries as alternative document representations.

3.2.3 Document Identifiers. In this work, we consider four kinds of different identifiers: the three kinds of document identifiers from the original DSI paper: unstructured atomic identifiers (Atomic IDs), naive string identifiers (Naive IDs), and semantically structured identifiers (Semantic IDs), and the 2D Semantic IDs from Wang et al. [41].

Atomic IDs. We treat each docid as a single, or "atomic" token in this setting. The decoder, then, only needs to run for a single decoding step; we then sort the logits of the docids to obtain the ranked document list. The setting requires adding a token, for each document, to the model vocabulary, increasing the model's parameter count by corpus size times embedding dimension, which can be expensive for large corpora. When considering millions of documents, we apply two optimizations to make implementation more feasible. First, the encoder's embedding table is adjusted to only consist of the standard T5 vocabulary, while the decoder's output projection only corresponds to docids. Second, we take special care to ensure the output projection is properly sharded across cores to distribute memory cost to allow scaling. In the `t5x` framework [34], this corresponds to setting appropriate partitioning rules.

Naive IDs. In this setting, the original document identifier from a corpus is directly used and treated as a textual string. For example, a five-digit number "42915" is treated as a string and passed through the SentencePiece vocabulary of T5. It is worth noting that such naive document identifiers might also capture some semantics of the corpus, as they depend on the curation pipeline that might leak some notions of relatedness.

Semantic IDs. Following Tay et al. [38], instead of relying on predefined identifiers, Semantic IDs aim to imbue document identifiers with hierarchical semantic information. Specifically, after encoding documents into dense vectors, a hierarchical k-means algorithm recursively clusters the space into k clusters until individual clusters include no more than c documents. Consequently, all document identifiers form a tree, where non-leaf nodes correspond to super-clusters, and leaf nodes are clusters with at most c documents each. Semantic IDs are formed by composing these cluster ids, each from 0 to $k-1$, tailed by a document id in the leaf nodes between 0 and $c-1$. In this work, we use the identifiers generated by Wang et al. [41] for NQ and TriviaQA for a fair comparison. These are based on a 12-layer BERT model. For MS MARCO, we use SentenceT5-Base [22], and $c = 100$. Since the passage-level corpus is large, if a cluster ends up bigger than 1M documents, we sample 100k when computing centroids. We used $k = 10$ clusters at each level, corresponding to the ten digits $(0 \ldots 9)$.

2D Semantic IDs. In the Semantic ID setting, the same tokens are used to represent different semantic meanings at different positions: we use the same set of numbers/tokens between 0 to $k-1$ for all identifiers, but they represent semantic clusters at different levels in the tree. To address this, NCI [41] extends the Semantic ID and introduces its 2D variant by adding an extra dimension to encode the positions, making the model aware of levels of clustering when decoding the identifier. To implement this modeling change,

they additionally introduce a change to the decoder described in the next section.

3.3 Model Variants

Besides alternative ways of constructing model inputs and targets, generative retrieval approaches that build on DSI have also investigated novel modeling components. Here, we review three model components introduced by Bevilacqua et al. [2] and Wang et al. [41].

Prefix-Aware Weight-Adaptive Decoder (PAWA) is proposed as a method for decoding 2D Semantic IDs. Unlike a standard Transformer decoder, which uses the same matrix to project the decoder's hidden representation to the vocabulary space for every position, PAWA uses different projection matrices at each timestep, with the weights of each projection matrix computed adaptively by a separate Transformer decoder. Specifically, in a vanilla decoder, the dense representation $h \in \mathbb{R}^{l \times d}$ from the last decoder layer is projected into the vocabulary space with $W \in \mathbb{R}^{d \times |V|}$, where l denotes the sequence length for decoding. To incorporate the position, the extra decoder in PAWA separately processes the input query and the already-decoded docid tokens to output a projection matrix $W^{pawa} \in \mathbb{R}^{d \times l \times |V|}$, replacing W. This aims to capture that the semantic meaning of a docid token depends on its position in the output sequence as well as on the docid prefix preceding it. The experiments in this paper use the open-source PAWA implementation provided by the original authors[1] as a reference and build it out on `t5x`. For more details, one could refer to [41] and their code base.

Constrained decoding can be used to avoid generating invalid document identifiers [2, 41]. A potential reason is that the space of identifiers is sparse, especially for Semantic IDs, and constrained decoding may help with memorization. While we have empirically found that roughly less than 1 in 20 DSI-based generation beams are invalid, we include this method nonetheless, as it is widespread in the literature. In this work, we adopt an exact match approach that leverages a trie to ensure only valid document identifiers are decoded.

Consistency loss can be used to alleviate over-fitting by introducing a regularization term. The basic idea is that the representations generated by two forward passes with different dropout masks should be similar. Wang et al. [41] incorporate this insight into a regularization term that augments the generation loss. We investigate the softmax version as described in the NCI paper (Eq. 5 in [41]) and a KL-divergence version from an early version of NCI (Eq. 1). They compute the Kullback-Leibler (KL) divergence between the output probabilities of two independent forward passes per position, where $p_{i,1}$ and $p_{i,2}$ are the probability distributions over the vocabulary space from the two forward passes at position i, respectively.

$$\mathcal{L}_{reg} = \frac{1}{2}[D_{KL}(p_{i,1} \,||\, p_{i,2}) + D_{KL}(p_{i,2} \,||\, p_{i,1})] \quad (1)$$

While we closely follow the implementation of the Neural Corpus Indexer code base, we find that these regularization terms lead to training instability and that the model effectiveness often diverges into a *NaN* loss. As a result, we do not include consistency regularization in our final experimental setup.

[1] https://github.com/solidsea98/Neural-Corpus-Indexer-NCI

Dataset	#Docs	% Covered by train query set
NQ100k [41]	110k	98.4%
TriviaQA [41]	74k	57.7%
MSMarco100k	100k	92.9%
MSMarco1M	1M	51.6%
MSMarcoFULL	8.8M	5.8%

Table 1: The coverage statistics of the benchmark datasets and their training query sets.

4 EXPERIMENTAL SETTING

We limit ourselves to English retrieval tasks, focusing on the behavior of generative retrieval models at varying corpus scales.

4.1 Corpus and Training Data

Following small-scale generative retrieval experiment setups [3, 38, 41, 45], we start with experiments on the Natural Questions [17] and TriviaQA [15] datasets. To better understand how different model configurations perform at scale and in more practical settings, we also experiment with variants of the MS MARCO passage ranking dataset. The MS MARCO passage ranking dataset consists of a corpus of 8.8M passages and a training set of 532K queries. From this dataset, we construct three variants, namely, MSMarco100k (100k passages), MSMarco1M (1M passages), and MSMarcoFull (all 8.8M passages). It is worth noting that most documents in NQ100k and MSMarco100k have at least one relevant query in the training set. However, as we scale to MSMarcoFull, the fraction of documents with queries in the training set drastically drops to around 6%, leading to a more practical setup. We summarize the statistics of these datasets in Table 1.

NQ100k and TriviaQA. To enable comparisons, we reuse the documents, the segmented documents, the training/testing splits, and generated query sets from Wang et al. [41]. The Natural Questions and TriviaQA datasets have corpora of sizes 109K and 74K, respectively. Note that Wang et al. [41] refers to NQ100k as NQ320k; we refer to the number of *unique* documents instead of the labeled training data size. Most documents in the NQ100k dataset have at least one relevant question in the training data, while 58% of the TriviaQA dataset has this property.

MSMarco100k. In the same vein as NQ100k and TriviaQA, we curate a dataset with 100k passages sampled from the full MS MARCO passage ranking dataset. Most passages have at least one positive query for training. We also include passages relevant to the queries in the development dataset (for evaluation).

MSMarco1M. This dataset is 10× larger than MSMarco100k. As with MSMarco100k, we augment the corpus with passages relevant to development queries. We first include all passages relevant to the 533K and 7K queries from the training dataset and development

sets, respectively. This results in 516K and 7K unique passages from each set. We randomly sample passages without a query in either set to total a million passages.

MSMarcoFULL. In this setting, we note another order of magnitude scale-up in corpus size. As a result, only 5.8% of the passages have a corresponding query in the training set. We aren't aware of any previous work that has attempted to apply generative retrieval models to a dataset of this size and complexity.

4.2 Synthetic Query Generation

For NQ100k and TriviaQA, we reuse the generated questions from [41] with 20 and 15 generated questions for each document, respectively. For the MSMarco variants, we use docT5query [25] to generate questions, with 40 generated questions per passage. We also train a question-generation model using T5-base using training data from DPR [16], a retrieval dataset derived from NQ [17]. We use this model to generate 40 questions per passage, following the configuration of docT5query. We refer to this variant as "in-domain D2Q" for NQ and TriviaQA.

4.3 Evaluation Dataset and Metrics

We report evaluation results on the development sets of each dataset. For NQ100k and TriviaQA, the evaluation dataset includes 7830 and 7993 questions each. For the three MSMarco variants, we use the validation split from the MS Marco passage ranking dataset, with 6,980 examples. For each query in the development sets, we use the models to generate ranked lists of documents. We report Recall@1 as the primary metric for Natural Questions and Recall@5 for TriviaQA. For the MS MARCO passage ranking variants, we use Mean Reciprocal Rank at 10 (MRR@10) as our primary metric.

4.4 Model Variants

We evaluate all methods using a T5.1.1 backbone [31]. We test variants of labeled vs. synthetic queries, FirstP vs. DaQ document representations, as well as combinations of multiple representations. For each model variant, we ablate all versions of document identifiers when applicable. Model architecture additions are performed, in a stacking fashion, starting with the base model and then adding on PAWA, constrained decoding, and consistency loss in this order. Note, we only evaluate PAWA with 2D Semantic IDs, as it is built specifically for that setting.

For model scaling experiments, we mainly investigate whether Atomic IDs are an effective way to scale to millions of passages, given the parameter cost. As such, we consider larger models with Naive IDs and Semantic IDs comparable to T5-Base with Atomic IDs, which total 7B parameters when scaling to 8.8M docids.

For baselines we provide BM25 [35] and BM25 with doc2query-T5 [25]. For Natural Questions and TriviaQA, we also include the previous results reported for the NCI-variant of NQ (i.e., NQ100k). This includes state-of-the-art generative retrieval results like NCI and GenRet [36], as well as GTR-Base, a state-of-the-art dual encoder [23]. For the new MS MARCO variants, we provide our own GTR-Base [23] results.

4.5 Implementation Details

We use T5.1.1 as implemented by `t5x` [34]. We implement the different setups described in Section 3 in the form of `seqio` tasks. For the MS MARCO variants, we set the maximum input sequence length to 128 for all experiments, and 64 for the NQ100k and TriviaQA, following the NCI setup. We initialize our models with the pre-trained T5-base model. For the PAWA decoder, we randomly initialize the PAWA model parameters. Following [38] for sequential IDs, beam search, with 40 beams, is used during inference.

We revise hyperparameter settings from [38] to ones we have found to empirically perform better, especially for indexing larger corpora like MSMarcoFULL. We set the batch size in all our experiments to 512. We train our models with a learning rate of 10^{-3} and a dropout rate of 0.1. We use 10k learning rate warm-up steps for all runs, except for Atomic IDs which use 100k steps. We train our small-scale datasets, NQ100k, TriviaQA, and MSMarco100k, for 1M steps. For MSMarco1M and MSMarcoFULL, we train our model to convergence or, at most, 9M steps. We use 8 TPUv4 chips for training models at the T5-Base scale. T5-Large, T5-XL, and T5-Base with Atomic IDs over MSMarcoFULL use 64 TPUv4 chips. For T5-XXL, we use 128 chips. Our most expensive runs took roughly 10-14 days to train to convergence on MSMarcoFULL.

5 EXPERIMENTAL RESULTS

We report our results in three parts. First, we ablate all the methods from Section 3 using T5-base on small-scale datasets: NQ100k and TriviaQA. We observe which techniques work best on this small scale with widely studied datasets. Then we transfer the same set of techniques and scale up to the entire MS MARCO passage ranking dataset to observe whether the same methods hold their ground at larger scales and discuss our findings. Finally, to understand whether the effectiveness benefit from Atomic IDs can be attributed to additional model parameters on large corpora, we select the best approach and scale the model size up to 11B (T5-XXL equivalent) for sequential ID approaches.

5.1 Ablations over Small Corpora

We report our ablations over NQ100k and TriviaQA in Table 2. The strongest combination of our techniques (row 7) sets a new state-of-the-art result on NCI's variant of NQ, without using any sophisticated modeling techniques. The choice of document representation by far dominates the overall performance of the retriever. Using just the training queries provided by the dataset performs the worst due to the low coverage of the documents. FirstP is a major improvement over this and DaQ is better than FirstP. However, the usage of D2Q is essential to strong generative retrieval performance, resulting in a 7pt+ gain. This by far trumps all other proposed techniques.

As for other design choices, we see that at this small scale, naive and Semantic IDs perform about on par (varying between task configurations), with Atomic IDs consistently the best. We note though that on NQ100k, Atomic IDs add 80M parameters to a T5-Base model that would otherwise be 220M parameters (a 36% increase). Given the comparable performance in the best configuration (row 7), these extra parameters may or may not be worth it, but we refer to Section 6.2 for more discussion. Modeling techniques from [41],

	Model	NQ100k At.	NQ100k Nv.	NQ100k Sm.	TriviaQA At.	TriviaQA Nv.	TriviaQA Sm.
Baselines							
	BM25 (via Wang et al. [41])	-	15.1	-	-	56.9	-
	BM25 w/ doc2query–T5 (via Wang et al. [41]	-	35.4	-	-	59.7	-
	GTR-Base (via Sun et al. [36])	-	56.0	-	-	-	-
	NCI [41]	-	62.8	65.9	-	88.8	90.5
	GenRet [36]	-	-	68.1	-	-	-
Ours							
(1a)	Labeled Queries (No Indexing)	50.7	49.2	49.0	60.9	56.7	61.4
(2a)	FirstP + Labeled Queries (DSI)	60.0	58.4	58.7	71.6	75.2	78.9
(2b)	DaQ + Labeled Queries	61.4	60.4	60.0	81.0	80.4	77.6
(3a)	DaQ + D2Q + Labeled Queries	69.6	67.9	67.9	88.2	85.7	86.3
(3b)	FirstP + DaQ + D2Q + Labeled Queries	69.0	68.2	67.2	88.9	86.9	87.4
(4a)	3b + PAWA (w/ 2D Semantic IDs)	-	-	66.3	-	-	86.5
(4b)	3b + Constrained Decoding	-	-	67.3	-	-	87.3
(5)	4b + Consistency Loss (NCI)	-	-	66.3	-	-	86.6
(6a)	DaQ Only	17.1	18.4	15.6	41.0	31.3	20.6
(6b)	D2Q Only	43.6	42.3	42.9	61.9	57.8	57.1
(6c)	6b + PAWA (w/ 2D Semantic IDs) + Constrained Decoding	-	-	43.1	-	-	57.7
(7)	3b + in-domain D2Q	**70.7**	**69.7**	**69.5**	**90.0**	**88.0**	**89.2**

Table 2: Results on small scale Natural Questions and TriviaQA datasets, reported in Recall@1 and Recall@5 respectively. First block presents baseline results in existing literature. Second block presents ablation results in a stacking fashion. Third block demonstrates the importance of document representation, in particular D2Q. Last row is the best method revised with in-domain D2Q.

i.e. 2D Semantic IDs, PAWA, constrained decoding, and consistency loss, do not reliably improve the model over the use of synthetic queries alone.

At this corpus scale, our best result uses a mixture of FirstP, DaQ, labeled queries, and synthetic queries for training. However, importantly, the *quality* of the synthetic queries are quite important, with synthetic queries from a generator specifically trained for the question-answering domain significantly outperforming the query generator trained over MS MARCO which was used by previous works.

5.2 Scaling Corpus Size

We now consider the scaled version of the MS MARCO passage ranking task, scaling from 100k to 1M and 8.8M passages. Results are reported in Table 3. Perhaps the most striking observation about the transition to MS MARCO is the absolute requirement of synthetic queries for strong retrieval performance. Synthetic queries result in a 2-3x improvement over the original DSI formulation alone. In fact, using only synthetic queries to docid as the indexing task is the most effective and straightforward training strategy on MS MARCO. This is a notable difference in the transition from NQ and TriviaQA to MS MARCO, where FirstP and DaQ did provide substantial value. This may be due to NQ and TriviaQA being based on Wikipedia articles: the beginning of Wikipedia documents are informative entity descriptions, and many sentences refer to the entity–which is likely the answer to a requested query.

As corpus size grows, DSI performance rapidly drops off, with the best result (D2Q only with Atomic IDs) rapidly falling off from 80.3 to 55.8 and finally 24.2 as we scale to the full 8.8M passages. Vanilla Semantic IDs also drop off as we scale to the full corpus, under-performing naive identifiers. We conjecture that this may be due to the potentially increased length of semantic identifiers being more difficult to decode than naive identifiers coupled with a noisy partitioning of the semantic space (especially when using an off-the-shelf embedding model such as SentenceT5-Base.) However, we do observe that Semantic IDs decoded via PAWA perform better. We provide some insight into why this might be in the next section where we examine model size. Constrained decoding only provides marginal value and generally is not worth the added complexity.

5.3 Scaling Model Size

How much of Atomic ID's strong performance can be attributed to its additional model parameters? On MSMarcoFULL, decoding Atomic ID document tokens adds an additional 7B parameters to the otherwise 220M parameter T5-Base model. We take the best configuration on MSMarcoFULL from Table 3 and scale model parameters of Naive ID and Semantic ID (PAWA) to similar sizes for comparison. We report results in Table 4.

Overall, we observe a general trend that as parameter count increases, retrieval performance improves. Indeed, both Atomic IDs and PAWA Semantic IDs had the strongest performance in Table 3, which we now attribute to their increased size. Notice that the difference here only comes out when scaling to MSMarcoFULL, where these parameter differences magnify significantly over smaller corpus scales. However, not all methods are equal. PAWA and 2D Semantic IDs [41] significantly increase decoding parameters with its extra decoding stack, yet yield no gain over naively scaling the Transformer with Naive IDs, underperforming

		MSMarco100k			MSMarco1M			MSMarcoFULL		
	Model	At.	Nv.	Sm.	At.	Nv.	Sm.	At.	Nv.	Sm.
Baselines										
BM25		-	65.3	-	-	41.3	-	-	18.4	-
BM25 (w/ doc2query–T5)		-	80.4	-	-	56.6	-	-	27.2	-
GTR-Base		-	83.2	-	-	60.7	-	-	34.8	-
Ours										
(1a)	Labeled Queries (No Indexing)	0.0	1.1	0.0	0.0	0.5	0.0	0.0	0.0	0.0
(2a)	FirstP/DaQ + Labeled Queries (DSI)	0.0	23.9	19.2	2.1	12.4	7.4	0.0	7.5	3.1
(3b)	FirstP/DaQ + D2Q + Labeled Queries	79.2	77.7	76.8	53.3	48.2	47.1	14.2	**13.2**	6.4
(4a)	3b + PAWA (w/ 2D Semantic IDs)	-	-	77.1	-	-	50.2	-	-	9.0
(5)	4a + Consistency Loss (NCI)	-	-	77.1	-	-	50.2	-	-	9.1
(6b)	D2Q only	**80.3**	**78.7**	**78.5**	**55.8**	**55.4**	54.0	**24.2**	**13.3**	11.8
(4a')	6b + PAWA (w/ 2D Semantic IDs)	-	-	78.2	-	-	**54.1**	-	-	**17.3**
(4b')	6b + Constrained Decoding	-	-	**78.6**	-	-	54.0	-	-	12.0
(5′)	6b + PAWA (w/ 2D Semantic IDs) + Constrained Decoding	-	-	78.3	-	-	**54.2**	-	-	**17.4**

Table 3: Results on the development set of the scale variant MS MARCO V1 passage collections, reported in MRR@10. Best results per column and results within 0.1 of best are bolded. Note that FirstP here is equivalent to DaQ as MS MARCO input passages fit into the input window.

T5 Scale	Training	Params	Inference FLOPs	MRR@10
Base	D2Q Only + Atomic ID	7.0B	0.9×10^{12}	24.2
Base	D2Q Only + Naive ID	220M	1.4×10^{12}	13.3
Base	D2Q Only + PAWA (2D Sem.)	761M	6.8×10^{12}	17.3
Large	D2Q Only + Naive ID	783M	3.5×10^{12}	21.4
Large	D2Q Only + PAWA (2D Sem.)	2.1B	1.1×10^{13}	19.8
XL	D2Q Only + Naive ID	2.8B	9.3×10^{12}	**26.7**
XXL	D2Q Only + Naive ID	11B	4.3×10^{13}	24.3

Table 4: Scaling up model size for sequential ID approaches in comparison to Atomic IDs for MSMarcoFULL.

by 4pts at around 700M parameters. This pattern continues to hold scaling PAWA to 2.1B parameters, thus, in order to save resources, we do not scale PAWA any further.

Scaling Transformers naively according to default T5 scales while using Naive IDs had the strongest performance on MSMarcoFULL at 26.7 MRR@10. Using only 2.8B parameters, this approach outperforms T5-Base with Atomic IDs which uses 7B parameters while achieving only 24.2 MRR@10. However, while parameter count has practical implications regarding the resources required for training and inference (especially TPU/GPU memory), there are other trade-offs to consider, which we discuss in the next section.

While Naive IDs perform well at T5-XL size, surprisingly we find that scaling further to XXL (11B) does not improve performance; in fact, it is detrimental to retrieval performance (24.3 MRR@10 vs. XL's 26.7) under the same experimental settings and hyperparameter settings, even though model training converges faster. This is counter-intuitive to most generative tasks and to the typical intuition of generative retrieval relying on model capacity to index an entire corpus of documents.

6 DISCUSSION

Our results raises multiple questions regarding the current state of generative retrieval at scale. We provide additional analysis here.

6.1 Why are synthetic queries effective?

Although the use of synthetic queries as a document representation technique has been shown to be effective in previous works [3, 41, 45], our experiments highlight its central importance to generative retrieval on a larger, more challenging corpus. We suggest that the effectiveness of synthetic queries mainly come from augmenting the input distribution during training to be closer to that observed at inference/evaluation time. Mainly, this comes in two forms: mitigating the coverage gap of ground-truth labeled queries and the document corpus, and closing the gap between the training query distribution and inference/evaluation. In addition, we find that the diversity of generated synthetic queries also can have a significant effect on retrieval performance.

Document coverage gap. In Table 1, for each dataset we report the coverage of their document corpus by the corresponding labeled query training set. When comparing MSMarco100k, 1M, and FULL the query coverage drops from 92.9% to 51.6% and 5.8% respectively. Consider rows (2a) and (3b) in Table 3 which only differ by the addition of synthetic queries. Here we observe that MSMarco100k improved by 3.3x while MSMarco1M improved by 3.9x, even though 1M is a larger corpus and may be affected by model capacity as

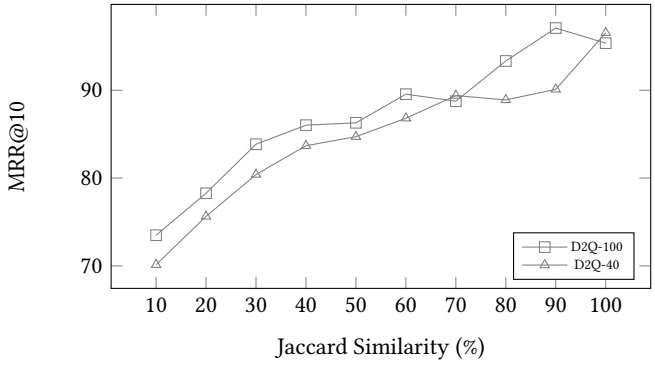

Figure 1: Jaccard similarity between synthetic queries and validation set queries vs. MRR@10 on the MSMarco100K subset.

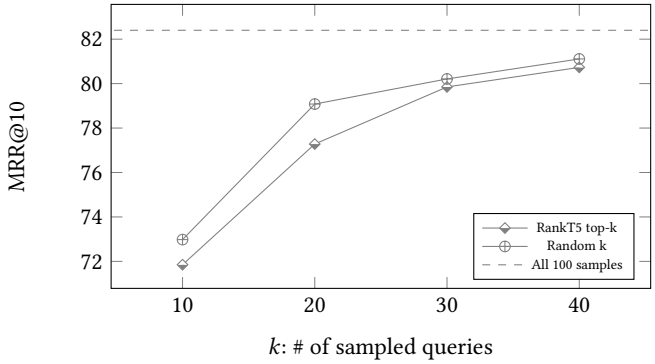

Figure 2: MSMarco100K MRR@10 as we vary the number of synthetic queries per passage. Given 100 pre-generated queries per passage, we compare random-k sampling, top-k selection via RankT5-XL, and using all 100 synthetic queries.

we see with MSMarcoFULL. Similarly, for NQ100k and TriviaQA, which have 98.4% and 57.7% coverage respectively, we observe that swapping Labeled Queries (No Indexing) (row 1a) for D2Q only (row 6b) hurts performance for NQ100k while improving performance for TriviaQA (Table 2.) Since this D2Q model is trained on MS MARCO, for NQ100k replacing its own labeled queries with synthetic queries only amounted to a 1.6% coverage gain, which is not worth the domain shift. However, for TriviaQA this amounted to a 42.3% coverage gain, which is more worth the domain shift.

Query distribution gap. Synthetic query generation effectively closes the query distribution gap between training and evaluation. Table 2 row 7, first shows the importance of the query distribution by using an in-domain query generation model to improve retrieval performance. To further understand the relationship between retrieval performance and query distribution gap, we plot the relationship between synthetic query similarity vs. validation query similarity and retrieval performance (MRR@10). For each evaluation query in the MS MARCO validation set, we measure the maximum similarity among all synthetic queries generated for the corresponding passage. Jaccard similarity is used for simplicity. For each evaluation query we then evaluate MRR@10 using the Atomic ID variant of row 6b in Table 3. Figure 1 reports the average MRR@10 within each 10pt Jaccard similarity bucket. We plot two variants using 40 and 100 sampled queries per passage for comparison.

In general, higher Jaccard similarity correlates with higher MRR@10 performance. That is, the more similar our training queries are to the evaluation the stronger the retrieval performance. Comparing the two settings, we see that higher exposure to more synthetic queries typically promotes higher effectiveness across similarity buckets. Even though the query distribution is important, it is worth noting that even on the lowest end of similarity this setting still has strong retrieval performance. While synthetic query distribution is an important aspect of retrieval performance, it is not singular in determining the end effectiveness and the generative retrieval model goes far beyond simply detecting lexically similar queries to those seen during training.

Diversity. We provide further analysis regarding the importance of synthetic query diversity. Here we assume the same MSMarco100k setting using the Atomic ID variant of row 6b in Table 3. We vary the number of sampled synthetic queries per passage used for training and observe MRR@10. We consider using 10, 20, 30, 40 and 100 sampled queries per passage, which we construct by first sampling the full 100 then taking random subsets of the varying sizes. We use a sampling temperature of 1.0 and consider the top 10 tokens at each sampling step. Recent studies show advances in utilizing cross encoders to refine the generated query set of incoherent, unspecific queries to improve the use of D2Q [11]. Accordingly, we also experiment with ranking the 100 sampled queries and taking top-10,20,30,40 instead of randomly sampling. We do so using a state-of-the-art cross-attention re-ranker, RankT5-XL [44], to score (generated query, passage) pairs and then take the top-k.

We report results in Figure 2. We find that, consistently, sampling more synthetic queries improve performance in this setting. Surprisingly, applying RankT5-based selection over the samples hurt performance. This suggests an overall preference for more samples, and more diverse samples to improve effectiveness. Using all 100 samples performed the best, increasing MRR@10 from 80.3 (Table 3, which used 40 samples) to 82.4, closing the gap with GTR-Base (83.2 MRR@10) on MSMarco100k. Exactly why query diversity is so important still up for interpretation, but there could be a couple possiblities: more diverse samples gives higher probability of at least some of the samples being close to the target distribution and more samples could provide a type of regularization to the model.

6.2 Which model scaling approach is best?

Much of this paper has considered parameter cost as a proxy for memorization capacity, which has been conjectured in the past to be important for retrieval [38]. However, model comparisons should not stop at parameter counts as this may not correlate with other cost indicators (training speed, FLOPs, etc.) that are important to practical applications [7]. While ultimately the best method to scale generative retrieval models will be the one that unlocks the potential of the paradigm to be competitive on large scale retrieval

tasks, we can provide some first glimpses into what trade-offs are at stake as we consider larger models for larger corpora.

As a case study, we consider T5-Base with Atomic IDs compared as T5-XL with Naive IDs from Table 4. Both are trained only with synthetic queries, and represent the only two viable approaches from our experiments. PAWA severely underperforms with regards to quality as we scale model size, not to mention the FLOP expense of having an extra decoding stack during inference. We provide discussion on parameter cost, training speed, and inference FLOPs.

Parameters. As corpus size scales, generative retrieval models face a fundamental prerequisite in model size to achieve decent performance, as seen in Table 3. Between three different ways of adding parameters (naive scaling, Atomic IDs, PAWA decoder), we see quality improvements over the smaller models. As discussed, on a fixed parameter budget basis Naive IDs perform the best on MSMarcoFULL, and best in quality overall.

Training Speed. Applications that require frequent retraining value fast total training walltime. We train T5-Base Atomic IDs and T5-XL Naive IDs on the same hardware (64 TPUv4) and hyperparameter settings. To achieve the optimal performance reported in Table 4, T5-XL Naive IDs required 14 days while T5-Base Atomic ID required only 7 days. However, at 7 days T5-XL Naive IDs was quality matched with T5-Base Atomic IDs (24.5 MRR@10), making both approaches roughly equal in terms of training wall-time when accounting for quality.

Inference FLOPs. Inference FLOPs can be a proxy for serving performance, although imperfect. Here we see that while sequential identifiers can achieve more with fewer parameters, atomic identifiers are incredibly FLOP efficient during inference. T5-Base with Atomic IDs for MSMarcoFULL requires only 9.7% the inference FLOPs of T5-XL with Naive IDs for 90% of the retrieval performance (Table 4). How is this possible? Atomic IDs incur additional compute cost to compute an output projection and softmax over the enormous vocab of 8.8M docids. However, it only has to compute this once to get a complete ranking of the *entire* corpus – a potentially very special property of the approach. On the other hand, sequential identifiers require d decoding steps to decode a single docid, and k beams to find a ranking of k docids. $k = 40$ for our experiments. Thus even though Atomic IDs require an expensive output projection, sequential ids require $O(d \cdot k)$ more decoding steps. To scale Naive IDs to be competitive with Atomic IDs, also makes individual decoding steps significantly more expensive.

In the end, we cannot yet say which approach is the best as the paradigm has yet to achieve competitive results on MS Marco passage ranking. On small corpora (100k), Atomic IDs are the highest quality, efficient option without incurring too many extra parameters. From our experiments though we can see that training models to maximize memorization amplifies compute trade-offs, and the field must provide more nuanced discussions of cost trade-offs as it considers more realistic applications of generative retrieval.

7 LIMITATIONS & FUTURE DIRECTIONS

As with all empirical studies, ours has its own set of limitations which we urge the reader to consider. Multiple works have come after the experiments in this work, e.g., [3], and thus we do not present an exhaustive set of generative retrieval techniques here. For example, the wide space of identifiers based on natural language or learned codes. In addition, due to resource constraints our model scaling experiments are not exhaustive, and not all ablation scenarios in Table 3 are scaled to larger model sizes. It could be possible that certain setups improve more at larger parameterizations, although unlikely; as with scaling past 11B. In addition, due to the extreme parameter requirements we do not saturate the scaling of Atomic IDs. Finally, since this work focused on the effectiveness of generative retrieval on large corpora, scaling model size for smaller corpora was outside our scope. Investigating the maximum corpus size for which generative retrieval could provide state-of-the-art performance is a question of practical importance which we leave for future work.

While open problems in generative retrieval have not changed (e.g. how to achieve state-of-the-art results on large corpora, how to update such as model with new documents [19], etc), we believe that our work also raises new open questions for the field. (1) How do we properly leverage large language models and the power of scaling model parameters to benefit generative retrieval on large corpora? While Tay et al. [38] showed this possibility over NQ, the same is not yet observed on MS MARCO even though intuitively expanded model capacity *should* benefit increased corpus scale. (2) How can we design model scaling recipes and derive scaling laws that maximize retrieval performance? In this work we only consider default T5 parameterizations, which may or may not be optimal for memorization heavy tasks. (3) How can we design architectures that can interpolate between the compute trade-offs of Atomic IDs and sequential IDs? We look forward to understand more about these problems in future works.

8 CONCLUSION

We provide the first empirical study of generative retrieval methods over the full MS MARCO passage ranking task of 8.8M passages. Of the various methods from the literature which we consider in this work [38, 41, 45], we find that the use of synthetic queries as a document representation strategy is the only approach that remained effective, and necessary, as we scaled up the corpus size using MS MARCO passages. We also highlight the importance of accounting for the compute cost of techniques; keeping the parameter count fixed, we find that naive methods outperform more sophisticated ones on the full MS MARCO dataset. Our strongest result on MS MARCO passage ranking uses only synthetic queries to Naive IDs as its training task, with the model scaled to T5-XL (3B). This model achieves 26.7 MRR@10. Surprisingly, increasing parameters for the same setting up to XXL (11B) performs worse. All of these findings suggest a need for continued research into generative retrieval, closer attention to method comparisons, and the potential need for fundamental improvements to the paradigm before we can leverage the power of larger language models.

9 ACKNOWLEDGEMENTS

The authors would like to thank Yi Tay, Tal Schuster, and Sanket Vaibhav Mehta for their valuable feedback and discussions.

REFERENCES

[1] Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. XOR QA: Cross-lingual Open-Retrieval Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 547–564. https://doi.org/10.18653/v1/2021.naacl-main.46

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. https://doi.org/10.48550/ARXIV.2204.10628

[3] Xiaoyang Chen, Yanjiang Liu, Ben He, Le Sun, and Yingfei Sun. 2023. Understanding Differential Search Index for Text Retrieval. *arXiv preprint arXiv:2305.02073* (2023).

[4] Xuanang Chen, Jian Luo, Ben He, Le Sun, and Yingfei Sun. 2022. Towards robust dense retrieval via local ranking alignment. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*. 1980–1986.

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2022. Overview of the TREC 2022 deep learning track. In *Text REtrieval Conference (TREC)*. TREC. https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2021-deep-learning-track/

[6] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. https://doi.org/10.48550/ARXIV.2010.00904

[7] Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. 2022. The Efficiency Misnomer. In *International Conference on Learning Representations*. https://openreview.net/forum?id=iulEMLYh1uR

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*.

[9] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 6894–6910. https://doi.org/10.18653/v1/2021.emnlp-main.552

[10] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008* (2018).

[11] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. 2023. Doc2Query–: When Less is More. (2023). https://doi.org/10.48550/ARXIV.2301.03266

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 2 (2006), 1735–1742.

[13] Kai Hui, Honglei Zhuang, Tao Chen, Zhen Qin, Jing Lu, Dara Bahri, Ji Ma, Jai Gupta, Cicero Nogueira dos Santos, Yi Tay, and Donald Metzler. 2022. ED2LM: Encoder-Decoder to Language Model for Faster Document Re-ranking Inference. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, Dublin, Ireland, 3747–3758. https://doi.org/10.18653/v1/2022.findings-acl.295

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

[15] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* (2017).

[16] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick 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)*. 6769–6781.

[17] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[18] Xueguang Ma, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2022. Document Expansion Baselines and Learned Sparse Lexical Representations for MS MARCO V1 and V2. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[19] 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:2212.09744 [cs.CL]

[20] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking Search: Making Domain Experts out of Dilettantes. *SIGIR Forum* 55, 1, Article 13 (jul 2021), 27 pages. https://doi.org/10.1145/3476415.3476428

[21] 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 *CoCo@ NIPS*.

[22] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang (Eds.). 2022. *Sentence-T5: Scaling up Sentence Encoder from Pre-trained Text-to-Text Transfer Transformer*. https://aclanthology.org/2022.findings-acl.146/

[23] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang (Eds.). 2022. *Large Dual Encoders Are Generalizable Retrievers*. https://preview.aclanthology.org/emnlp-22-ingestion/2022.emnlp-main.669.pdf

[24] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 708–718. https://doi.org/10.18653/v1/2020.findings-emnlp.63

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

[26] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *CoRR* abs/1910.14424 (2019). arXiv:1910.14424 http://arxiv.org/abs/1910.14424

[27] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[28] Ronak Pradeep, Yilin Li, Yuetong Wang, and Jimmy Lin. 2022. Neural Query Synthesis and Domain-Specific Ranking Templates for Multi-Stage Clinical Trial Matching. 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, 2325–2330. https://doi.org/10.1145/3477495.3531853

[29] Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. 2021. Vera: Prediction Techniques for Reducing Harmful Misinformation in Consumer Health Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2066–2070. https://doi.org/10.1145/3404835.3463120

[30] 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 (2021).

[31] 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. *Journal of Machine Learning Research* 21 (2020), 1–67.

[32] 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. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[33] 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]

[34] Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling Up Models and Data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189* (2022). https://arxiv.org/abs/2203.17189

[35] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[36] 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. arXiv:2304.04171 [cs.IR]

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. *arXiv preprint arXiv:1409.3215* (2014).

[38] 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 (2022).

[39] Dan Vanderkam, Robert B Schonberger, H. Rowley, and Sanjiv Kumar. 2013. Nearest Neighbor Search in Google Correlate.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[41] 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 (2022).

[42] 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]

[43] 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.

https://doi.org/10.48550/ARXIV.2208.09257
[44] 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. https://doi.org/10.48550/ARXIV.2210.10634
[45] 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. *arXiv preprint arXiv:2206.10128* (2022).
[46] Noah Ziems, Wenhao Yu, Zhihan Zhang, and Meng Jiang. 2023. Large Language Models are Built-in Autoregressive Search Engines. arXiv:2305.09612 [cs.CL]