Static Pruning for Multi-Representation Dense Retrieval

Antonio Acquavia
University of Pisa, Italy

Craig Macdonald
University of Glasgow, UK

Nicola Tonellotto*
University of Pisa, Italy

ABSTRACT

Dense retrieval approaches are challenging the prevalence of inverted index-based sparse representation approaches for information retrieval systems. Different families have arisen: *single* representations for each query or passage (such as ANCE or DPR), or *multiple* representations (usually one per token) as exemplified by the ColBERT model. While ColBERT is effective, it requires significant storage space for each token's embedding. In this work, we aim to prune the embeddings for tokens that are not important for effectiveness. Indeed, we show that, by adapting standard uniform and document-centric static pruning methods to embedding-based indexes, but retaining their focus on low-IDF tokens, we can attain large improvements in space efficiency while maintaining high effectiveness. Indeed, on experiments conducted on the MSMARCO passage ranking task, by removing all embeddings corresponding to the 100 most frequent BERT tokens, the index size is reduced by 45%, with limited impact on effectiveness (e.g. no statistically significant degradation of NDCG@10 or MAP on the TREC 2020 queryset). Similarly, on TREC Covid, we observed a 1.3% reduction in nDCG@10 for a 38% reduction in total index size.

ACM Reference Format:
Antonio Acquavia, Craig Macdonald, and Nicola Tonellotto. 2023. Static Pruning for Multi-Representation Dense Retrieval. In *ACM Symposium on Document Engineering 2023 (DocEng '23), August 22–25, 2023, Limerick, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3573128.3604896

1 INTRODUCTION

Pre-trained contextualised language models such as BERT have been shown to greatly improve retrieval effectiveness over the previous state-of-the-art methods in many information retrieval (IR) tasks [16]. These contextualised language models are able to learn semantic representations called *embeddings* from the contexts of words and, therefore, better capture the relevance of a document w.r.t. a query, with substantial improvements over the classical approach in the ranking and re-ranking of documents [25]. Most BERT-based models are computationally expensive for estimating query-document similarities in ranking, due to the complexity of the underlying transformer neural network [17, 21, 46]. As such, BERT-based ranking models have been used as second-stage rankers in retrieval cascades, in particular to re-rank candidate documents generated by classical relevance models such as BM25 [27–29]. Recently, several works have proposed investigating whether BERT-based systems are able to identify the relevant passages among all passages in a collection, rather than just among a query-dependent sample; these systems represent a new type of retrieval approach called dense retrieval. In dense retrieval, passages are represented by real-valued vectors, while the query-document similarity is computed by deploying efficient nearest neighbour techniques over specialised indexes, such as those provided by the FAISS toolkit [19].

Recently, two different families of dense retrieval approaches have emerged, based on *single representation* and *multiple representation*. In particular, DPR [20] and ANCE [45] use a single representation, assumed to represent the meaning of an entire document within a single embedding. In contrast, ColBERT [21], which uses multiple representation, indexes an embedding for each token in each document. This causes ColBERT to form a large index of embeddings, which can provide more effective results than single representations, but at the cost of higher mean response times and memory occupancy [31]. While dense retrieval systems leveraging single representations are able to store all the embeddings in few tens of gigabytes and perform exact nearest neighbour search to identify the top document for a query, the dense retrieval systems using multiple representations leverage quantised indexing techniques to generate highly compressed representations of the embeddings, searchable with approximate nearest neighbour algorithms. This approach produces candidate documents that must be re-ranked in a second stage in order to maximise the effectiveness of the returned documents. These second stage rankers leverage the original uncompressed document embeddings to compute the final query-document similarity. To do so, it is necessary to maintain an uncompressed embeddings index together with the quantised index, at the cost of hundreds of gigabytes, and longer query processing times.

A natural solution to address the aforementioned space problem is to try to remove from the embedding index those components that, on average, do not contribute to or do not impact on the effectiveness of the final results. A similar approach has been investigated in the past for sparse retrieval based on inverted index data structures. In this approach, named *static pruning* [9], the space occupancy of an inverted index was reduced by removing documents, terms or document-term pairs from the collection or the inverted index. Among the many alternatives, the removal of very frequent terms in the collection, i.e., with a low inverse document frequency (IDF) has been shown to be one of the most successful [6, 7], as it was able to remove a large portion of terms from documents with limited impact on the average effectiveness. Furthermore, IDF-based static pruning does not require additional information derived from query logs or other user data [2].

In this paper we focus on adapting the IDF-based static pruning strategies to embedding indexes in multi-representation dense retrieval. In doing so, we aim to reduce the space occupancy of the embedding indexes in multi-representation dense retrieval. Since terms are now replaced by embeddings in a continuous vector space, we firstly need to bridge the gap between semantic representations, i.e., embeddings, and lexical tokens, i.e., terms, and then

* Alphabetical Ordering

DocEng '23, August 22–25, 2023, Limerick, Ireland

we cast the term-based static pruning over inverted indexes into an embedding-based static pruning over embedding indexes.

As a result, this paper takes a new perspective on static pruning, showing how simple adaptations of well-understood techniques can be generalised and applied to multi-representation dense retrieval. This modern take shows that even for contextualised embeddings, stopwords and low IDF terms are still unimportant for retrieval, and can be safely removed without significant degradation of retrieval effectiveness. Our experiments on the MSMARCO v1 passage ranking corpus show that by removing all embeddings corresponding to the 100 most frequent BERT tokens, index size is reduced by 45%, without significant degradation of NDCG@10 or MAP on the TREC 2019 or TREC 2020 querysets, and only 3% reduction in MRR on the MSMARCO Dev queryset. Moreover, we validate the generalisation of our results on the TREC Covid test collection. On this collection, we observed a 1.3% reduction in nDCG@10 for a 38% reduction in total index size. Hence, these simple static pruning approaches demonstrate that the space usage of ColBERT dense retrieval can be reduced by almost half, but can also form baselines for more advanced pruning strategies that consider the exact contextualised embeddings. Furthermore, we provide our GitHub repository for this paper containing the Jupyter notebooks to reproduce all of the experiments in this paper.

The remainder of the paper is structured as follows: Section 2 discusses related work in static pruning and provides a background in neural re-ranking and dense retrieval; Section 3 discusses the pruning of multi-representation dense retrieval configurations and elicits our research questions; Experimental setup is discussed in Section 4; Results on MSMARCO are discussed in Section 5; Validation of our main results on TREC Covid are provided in Section 6; Concluding remarks follow in Section 7.

2 RELATED WORK

We first discuss the pruning of classical, i.e. *sparse*, inverted index data structures. Then we introduce the neural re-ranking and dense retrieval approaches.

2.1 Static Pruning

The static pruning of inverted indexes deals with removing information stored in the inverted index to improve the efficiency of top k query processing with negligible or minimal negative impact on the effectiveness. The most simple and widely applied pruning approach is the removal of stopwords from the inverted index, which rarely contribute to the final document retrieval scores.

Static pruning approaches in the literature can be classified according to the different components of an inverted index that they prune. An inverted index is composed by posting lists, one for each unique term in the collection; each posting list contains postings, one for each document in which the corresponding term appears at least once. *Uniform* strategies focus on completely removing terms [6, 9, 10, 34] or documents [34] from the inverted index. The usefulness of a term or a document is measured through its *importance*, i.e., a quantification of its contribution to the relevance scores. Carmel *et al.* [9] measured term importance through the k-th relevance score in the term's posting list and removed the terms with an importance below a global threshold, while Blanco and Barreiro [6] proposed to completely remove posting lists corresponding to stopword terms, identified through their IDF scores. Ntoulas *et al.* [34] proposed to remove documents whose global scores, e.g., PageRank, fall below a given threshold.

While uniform strategies consider global importance thresholds only to make their pruning decisions, it is also possible to select thresholds on a term/document basis, hence selecting to prune only a portion of the posting lists, depending on the usefulness of each posting. In *term-centric* methods, the pruning decision is taken on a posting's rank w.r.t. the other postings in the corresponding posting list. Carmel *et al.* [9] proposed to prune postings with relevance scores below the k-th score of all scores in the posting list. De Moura *et al.* [15] further refined this strategy to take into account co-occurring terms. Ntoulas *et al.* [34] proposed to take into account jointly a global score and a term-centric score when taking pruning decisions. To select the postings to prune, Blanco and Barreiro [7] exploited the probabilistic ranking principle as a decision criterion over which postings should to be pruned. However, all of these term-centric methods rely on posting lists and posting scores, and hence cannot be easily adapted to a dense retrieval scenario where terms and/or documents are represented with embeddings and stored in a metric index, such as FAISS [19].

On the other hand, in *document-centric* methods, the pruning decision is taken on a posting's rank within the document it refers to. Büttcher and Clarke [8] proposed to prune a posting according to the Kullback-Leibler divergence between the document language model and the language model of the whole collection. For every document, they perform a pseudo-relevance feedback step at indexing time, and only keep postings for the top feedback terms extracted from that document in the index, discarding everything else. Totha and Carterette [39] statistically compared the in-document term frequency to the in-collection term frequency to decide whether to prune the given term. Altingovde *et al.* [2] proposed to combine both term-centric and document-centric pruning strategies with popularity, and proposed several advanced pruning strategies where the pruning decision is taken according to the number of times a posting is used in processing a training query log. While all pruning strategies discussed so far assume the availability of the inverted index, Altin *et al.* [1] discuss some pre-indexing pruning strategies to remove partially or completely documents from the document collection before the inverted index construction process, for example by discarding long documents or terms appearing in a stopwords list.

It is worth noting that static pruning of indexes permits information retrieval systems to create more aggressively pruned indexes for fast retrieval of more common queries, while keeping larger unpruned indexes for infrequent/more difficult queries [3, 34, 35, 37]. For example, Skobeltsyn *et al.* [37] showed that a combination term-centric and document-centric pruning in multi-layered tiering was able to handle up to 85% of queries with one quarter of the resources of the search engine's full index.

In contrast, the pruning of indexes applicable for dense retrieval is comparatively understudied. We discuss neural re-ranking approaches and dense retrieval approaches next.

2.2 Neural Re-ranking and Dense Retrieval

Various works study and show the superior effectiveness of pre-trained transformer networks, e.g., BERT, on ranking tasks. BERT is shown effective at *passage* re-ranking [33] and *document* re-ranking [29], by fine-tuning the transformer network to classify the query and passage/document[1] pairs as relevant or non-relevant, as measured using the embeddings of a special [CLS] (classification) token. The performance of BERT is also investigated in different retrieval-related tasks, such as keyword-based and natural language question answering [14, 44].

Most transformer-based neural ranking approaches are evaluated by re-ranking the documents identified by a classical inverted index and using relevance models such as BM25, in a multi-stage ranking architecture [24, 32]. However, by relying solely on an inverted index, there is the possibility that relevant documents that would have been highly scored by an effective neural ranking model are not included in the initial candidate set, for instance due to vocabulary mismatch. Instead, by utilising documents encoded as vectors at indexing time and queries encoded as vectors at query processing time, dense retrieval approaches [21, 45] are of growing interest. In dense retrieval, the top-ranked documents for a given query are computed by identifying the most similar document embeddings to a given query embedding, employing a nearest neighbour search procedure. Nearest neighbour search with single representations is efficient, but can be less effective than nearest neighbour search on multiple representations [25, 31], and leads to significant efforts to improve the performance of single representations, such as deploying hard negatives [47] and distillation of knowledge from more expensive cross-encoders [18] or ColBERT models [26].

On the other hand, when multiple representations are exploited, as pioneered by Khattab and Zaharia [21], a multi-stage dense retrieval approach can be executed, where the first stage conducts an *approximate* but highly efficient nearest neighbour search, retrieving a candidate set of documents to be exactly scored by the second stage. The use of multiple representations in dense retrieval poses several challenges. The time taken to score all retrieved documents can be expensive, and the space required to store the embeddings of the tokens in the documents can exceed the available memory, hence requiring a great number of time-consuming disk accesses. A straightforward approach to deal with this issue is to reduce the dimension of the embeddings, as done by ColBERT [21] and miniLM [42]. Tonellotto and Macdonald [40] investigated some efficiency improvement by pruning query term embeddings that are estimated not to be useful for retrieving relevant documents. They showed that a subset of the original query embeddings can be used for effective retrieval while reducing the number of documents requiring to be exactly scored. This subset of query embeddings contains only the embeddings corresponding to tokens with a high inverse collection frequency, i.e., query embeddings of common tokens in the document collection are ignored. Lassance *et. al* [23] studied the impact of token pruning in ColBERT during model training, obtaining a space reduction of 30% at the cost of a new

[1] In the remainder of this paper, we focus on passage ranking, and use passages/document nomenclature interchangably. In general, document ranking can be effectively achieved by scoring documents based on their highest scoring passage [14].

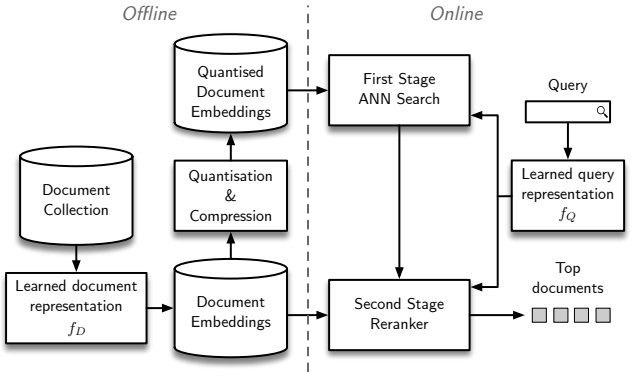


Figure 1: Dense retrieval architecture.

neural model training phase. Differently from [23], in this work we experiment with static pruning strategies that *do not require any further training* of the ColBERT model, but remove embeddings once they have been computed.

On the other hand, in [36], the original ColBERT authors described how to quantize the embeddings index, resulting in reduced space usage. In this work we take a different viewpoint – instead of quantizing the embeddings of each token, we aim to remove some embeddings entirely. These two directions are entirely orthogonal, and could be later combined. In short, in this work, we aim at adapting static pruning approaches to work on embedding-based document representations. In particular, leveraging the statistical information on lexical tokens in a collection, namely inverse document frequency, we propose to remove the corresponding embeddings at collection level or document level, as we discuss in detail in the following section.

3 PRUNING MULTIPLE REPRESENTATIONS

We now describe the general architecture of ColBERT multiple representation dense retrieval (Section 3.1), how to apply static pruning upon its embeddings (Section 3.2), how to compute the importance of tokens for pruning (Section 3.3). We then summarise our research questions (Section 3.4).

3.1 Multi-representation Dense Retrieval

The general architecture for dense retrieval follows the representation-focused model for neural ranking, i.e., queries and documents are represented by a set of fixed-length real-valued vectors, and a similarity score is computed from the query representations and the document representations, as depicted in Figure 1.

In particular, queries and documents are sequences of tokens from a given vocabulary V. Any token is represented by a real-valued vector of dimension k, called an embedding. More formally, let $f_Q : V^n \rightarrow \mathbb{R}^{n \times k}$ be a learned function mapping a given query token t_i in a query of n tokens to the query embedding ϕ_i, i.e., $\{\phi_1, \ldots, \phi_n\} = f_Q(t_1, \ldots, t_n)$. Similarly, let $f_D : V^n \rightarrow \mathbb{R}^{n \times k}$ be a (potentially different) learned function mapping a given document token t_j in a document of n tokens to the document embedding ψ_j, i.e., $\{\psi_1, \ldots, \psi_n\} = f_D(t_1, \ldots, t_n)$.

For a query q and a document d, their final similarity score $s(q, d)$ is obtained by summing up the maximum similarity between the

query token embeddings and document token embeddings:

$$s(q,d) = \sum_{i=1}^{|q|} \max_{j=1,\ldots,|d|} \phi_i^T \psi_j. \quad (1)$$

In dense retrieval, the document token embeddings from all documents in the collection are pre-computed *offline* through the application of the f_D learned function and stored into a document embeddings index data structure for vectors supporting similarity searches (left side of Fig. 1). In a multi-stage architecture, these embeddings are further processed *offline* by specialised techniques, such as those provided by the FAISS toolkit [19]. These techniques produce quantised and compressed embeddings, amenable to be efficiently searched using an approximate nearest neighbour (ANN) search technique (top left of Fig. 1). At query processing time, i.e., *online*, a user query is processed by the f_Q learned function, and may also be augmented with additional *masked tokens* to provide a query expansion-like role [21]. The resulting embeddings are used in the first stage ANN search, to identify a candidate set of documents to be passed to a second stage reranker. The reranker exploits the query token embeddings and the uncompressed document token embeddings for the documents in the sample to compute the final top documents to return (right side of Fig. 1).

3.2 Static Pruning

Static pruning strategies focus on two building components, namely tokens and documents. These components represent a lexical source of information, and their statistical properties, e.g., number of occurrences and frequency counts, both within documents and in the whole collection, are the main ingredients for query-document scoring functions, such as TF-IDF and BM25 [41]. In particular, these scoring functions can be expressed as sum of score contributions, one per query token:

$$s(q,d) = \sum_{i=1}^{|q|} s(t_i,d) \quad (2)$$

The impact $s(t_i,d)$ of a query token t_i to the overall score $s(q,d)$ depends entirely on the query token and its statistical properties in the document and the whole collection.

In dense retrieval we must take into account embeddings, which carry semantic information about tokens, derived from their contexts in documents. In fact, in multi-representation dense retrieval systems such as ColBERT [21], the scoring function in Eq. (1) depends on the metric properties, i.e., dot product, of the query token embeddings w.r.t. *all* the document token embeddings, due to the max operator. Hence, it is not clear how the static pruning strategies proposed for sparse inverted indexes perform when adapted to dense embedding indexes, since the query token-document score depends on the embeddings of all the tokens appearing in a document. In order to investigate the impact, in terms of space reduction and effectiveness, of static pruning strategies in the context of dense retrieval, we resort to map back document token embeddings to the corresponding tokens, arguing that this simplification is appropriate, as we will discuss in Section 3.3. Moreover, we propose the following adaptations of static pruning strategies for embedding indexes:

- *uniform pruning*: remove all embeddings in *any* document corresponding to the tokens ranked by their global importance, up to a threshold number of globally removed tokens;
- *document-centric pruning*: remove a number of embeddings in each document corresponding to the tokens ranked by their global importance, up to a threshold number of tokens removed per document.

Due to the fact that a query token-document score depends on the embeddings of all the tokens appearing in a document, it is not immediately obvious how to derive an adaptation for term-centric pruning – indeed term-centric pruning relies on pruning of terms in the same posting list, but there are no obvious ways to organise multi-representations into posting lists. We leave this as future work.

Finally, instead of pruning the documents before the application of the learned function f_D, we first compute the document token embeddings, to preserve the contextualised information captured by the pretrained transformer network, then we apply document embedding pruning to remove unimportant embeddings, i.e., those with almost no impact on the computation of the final results, as we discuss next.

3.3 Token Importance

In order to compute the token importance, Tonellotto and Macdonald [40] experimentally showed almost 90% of query token embeddings (3 vs. 32) do not contribute significantly to the recall of relevant documents by the first stage ANN search of a multi-representation dense retrieval system. By sorting the query token embeddings by IDF, it was shown to be sufficient to process only the top 3 embeddings with the highest IDF score, on average. Moreover, Lassance *et al.* [23] showed that ColBERT implicitly captures the notion of IDF to measure query-document token importance. Hence, it looks reasonable to exploit IDF as a proxy for token importance in document token embedding pruning as well. To further support this assumption, Figure 2 shows the ColBERT interaction between the query and document embeddings for the query "what are the social determinants of health" and a relevant document.

In the figure, the darker shading in the matrix is indicative of higher similarity between document embeddings and query embeddings; the document embedding with the highest similarity is selected for a given query embedding by the max operator in Eq. (1), and is indicated by the × symbols; the histogram at the top of the figure indicates the contribution of each query embedding to the final document score. Indeed, on inspection of the max similarities for this query-document pair shows that the highest contributions to the document's score comes from the 'social' 'deter' '##mina' '##nts' document tokens, which are exact matches with the query tokens – indeed, it is well known that ColBERT prefers exact matches [23]. On the other hand, document tokens including 'as', 'the', and 'in' (which have low IDF) are not maximally similar to any query embedding. Hence, these document embeddings are not contributing to the final score of the document for the query, and their removal will cause no difference.

Furthermore, it appears that the low IDF query tokens match with low IDF tokens in the document, but contribute less to the overall score of the document (as illustrated in the histogram at

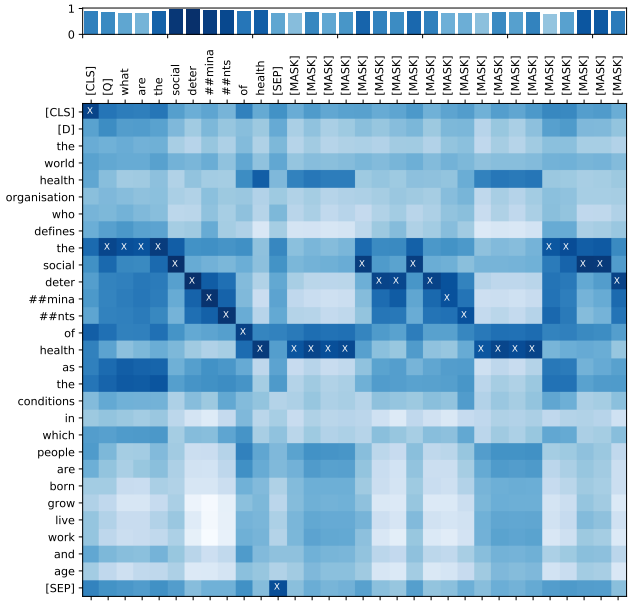


Figure 2: Visualisation of ColBERT interaction between query and document embeddings; the query includes additional embeddings for *masked tokens*, which have an expansion-like role; the selected maximally similar document embedding for each query embedding are denoted with $\times$.

the top of the figure). In particular, 'of' is an exact match, but there are also several inexact matches - e.g. query tokens 'what', 'are' and 'the' are matched with the document token 'the' – hence, these semantic matches are still between low IDF query tokens and low IDF document tokens. This gives support to simply measuring the global importance of a token in terms of its inverse document frequency: the more a token appears in different documents, the less a token is important, as proposed in [40] and supported by Fig. 2. However, as mentioned above, we apply pruning of embeddings *after* application of the learned model, to preserve the contextualisation of the embeddings.

Finally, while the corpus itself is a natural and easy source of token statistics, we also recognise that words that are frequent in queries may not be useful to keep. Indeed, some past work used query logs as a driver for pruning. This is motivated in conventional pruning in that words that do not occur in the queries need not be kept in the index. However, in ColBERT-based dense retrieval, query words can match to differing document words (a *semantic match* [43], rather than a lexical one). Hence, in this work, we also investigate using two large querysets to determine token importance [22].

3.4 Research Questions

The most space consuming data structure in Fig. 1 is the document embeddings index, used in the second stage reranker; our proposed static pruning adaptations target mainly this index. In the following experiments, we will investigate the following research questions:

- **RQ1**: What is the impact, in terms of effectiveness and space reduction, of document-centric pruning in dense retrieval, where we remove in each document a given number of embeddings, corresponding to low IDF terms?
- **RQ2**: What is the impact, in terms of effectiveness and space reduction, of uniform pruning in dense retrieval, where we remove globally a given number of embeddings, corresponding to low IDF terms?

However, in the first stage ANN search, we can exploit the original quantised document embeddings, or learn a new quantised data structure from the document embeddings after static pruning is deployed. We further investigate this approach, addressing the following additional research question.

- **RQ3**: What is the impact, in term of effectiveness, of learning a new quantised document embeddings index for first stage ANN search after static pruning?

4 EXPERIMENTAL SETUP

We now discuss the dataset (Section 4.1), evaluation methodology (Section 4.2) and pruning configurations (Section 4.3) applied in our experiments.

4.1 Dataset

We experiment using the MSMARCO (v1) passage ranking corpus, which consists of 8.8M passages [4]. We apply a ColBERT model [21] using the default settings proposed by Khattab & Zaharia. Our ColBERT model is trained for 200k steps. In particular, queries and passages are tokenised by the BERT WordPiece tokeniser. Thereafter, queries are encoded into 32 embeddings, one for each WordPiece token, and unused tokens being used as '[MASK]' query embeddings, which permit query augmentation [21]. Tokens in passages are encoded into 180 or less embeddings. The average number of embeddings (a.k.a. tokens) per passage in the entire corpus is 77.8, resulting in 185GB of index space (of which the embeddings index is 165GB). All index embeddings are stored as half-precision floating point numbers, i.e., using 16 bits per embedding component.

For retrieval, we use an ANN stage to identify the $k = 1000$ most similar passages, which are then re-ranked by an exact re-scoring stage. Indeed, the heavily-compressed ANN stage used by ColBERT results in scores that are not sufficiently accurate to be used for high-precision ranking, but reranking $k = 1000$ passages results in no loss in effectiveness [30].

For evaluating effectiveness, we use the TREC 2019 queryset, which contains 43 queries with an average of 215.3 relevance judgements per query, and the TREC 2020 queryset, which contains 54 topics with an average of 66.8 relevance judgements per query.

4.2 Experimental Methodology

Our overall aim is to determine which ColBERT dense retrieval configurations result in markedly smaller indexes (a.k.a. space efficiency) without (statistically significant) loss in ranking effectiveness. To aid in conducting experiments efficiently, we initially evaluate pruning effectiveness in a *re-ranking* setting, whereby a candidate set of documents is identified using an unpruned FAISS ANN stage of ColBERT, and then pruned, before being re-ranked.

We measure both effectiveness and space efficiency on the re-ranked documents.

In particular, for measuring effectiveness, we report NDCG@10 and MAP.[2] For significance testing, we apply a paired t-test, to determine statistical significance compared to the unpruned baseline. We do not apply multiple testing correction, as this would reduce the chances of observing significant degradations in effectiveness.[3]

For space efficiency, we measure the average length of the top k documents, in terms of number of embeddings, after pruning (denoted AvgDocLen@k). Indeed, we observe that the AvgDocLen@100 for the TREC 2019 and TREC 2020 querysets are respectively 79.2 and 78.2, which are both very close to the average of 77.8 observed in the corpus as a whole (sample sizes of 4300 and 5400 documents). Moreover, as we show in Section 5.2, these are sufficiently representative of the average document lengths of the pruned indexes.

4.3 Pruning Implementation and Settings

In the following re-ranking experiments, we apply seven families of ColBERT dense retrieval configuration:

- *Original*: This is the default ColBERT dense retrieval configuration, using all document embeddings.
- *Stopwords (Uniform)*: This uniform pruning removes the document embeddings for tokens matching a pre-determined list of stopwords. In particular, we apply Terrier's stopword list of 733 tokens, but only apply the 403 tokens that have an exact match with a token in the BERT tokenizer[4]. This can be interpreted as a uniform setting, as embeddings are removed from the index regardless of which documents they occur in.
- *Random doc-centric*: In this document-centric pruning approach, we remove τ_d random tokens from each document. Tokens are sampled without replacement.
- *IDF doc-centric*: In this document-centric pruning approach, we remove embeddings that correspond to the τ_d tokens in the document with lowest IDF. In this way, all passages are cut down.
- *IDF uniform*: This uniform pruning approach removes the embeddings corresponding to the τ_u tokens with lowest IDF from the entire corpus. This treatment is applied across the entire index – some passages could be untouched. In practice, the lowest IDF tokens are very similar to tokens that occur in a typical stopwords list (the five lowest IDF BERT tokens in the corpus are: 'the', 'of', 'and', 'a' and 'it').
- *MSMARCO uniform*: While *IDF uniform* computes the IDF on the MSMARCO corpus, this uniform pruning approach computes the IDF using the MSMARCO train queryset, and then removes the embeddings corresponding to the τ_u tokens with lowest IDF from the entire corpus.
- *MSN uniform*: This uniform pruning approach computes the IDF using the MSN query log [11], and then removes the embeddings corresponding to the τ_u tokens with lowest IDF from the entire corpus.

[2] For MAP, we set the relevance label threshold as 2 [12]. [3] We also considered use of Two-One-Sided Test (TOST), however, setting appropriate thresholds, particularly for non-linear measures such as MAP, make them non-trivial to apply. [4] Tackling this would involve extending the work to n-grams of BERT tokens, an added complication. We leave the pruning of such n-grams to future work.

For the latter five approaches, we vary the number of tokens being pruned, i.e. $\tau_d = \{1, 3, 5, 8, 10, 15, 20, 25, 30\}$, and $\tau_u = \{1, 10, 25, 50, 100, 250, 500, 750, 1000, 2000, 3000\}$, We also implemented two additional doc-centric baselines based on IDF computed on the MSMARCO train query set and the MSN query log. In both cases, the resulting curves overlapped with the IDF doc-centric approach, hence, for space constraints, we do not report them. Moreover, we consider another baseline ColBERT configuration, based on *miniLM*. This configuration consists of a dense retrieval baseline that uses a distilled version of BERT called miniLM (which uses less layers and less parameters [42]). In particular, we use a checkpoint of the model fine-tuned for ColBERT-like retrieval in the exact same manner as ColBERT. The miniLM approach reduces the index size not by pruning embeddings, but by reducing their dimensions, from 768 to 32. For more details, see [5]. Hence, this approach is orthogonal to our proposed approach, and we leave to future works an in-depth investigation of the impact of both approaches deployed jointly.

All source code to reproduce these experiments is available from https://github.com/cmacdonald/colbert_static_pruning.

5 RESULTS

Firstly, Section 5.1 discusses the re-ranking results for document-centric and uniform pruning, evaluated for space usage and effectiveness in a re-ranking setting (c.f. RQ1 & 2). Later, in Section 5.2, we discuss the results based on full indexes (c.f. RQ3).

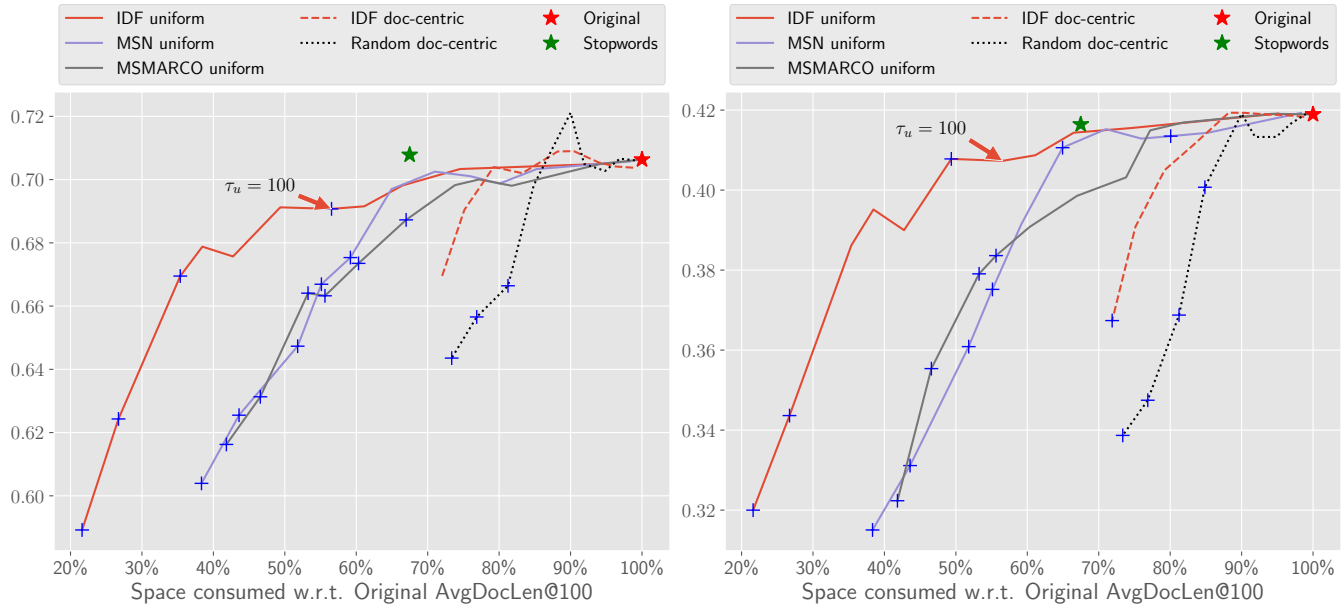
5.1 RQ1 & RQ2 - ReRanking Evaluation

Figures 3 and 4 respectively present our re-ranking performances on the TREC 2019 and TREC 2020 querysets. Graphs are shown for both nDCG@10 and Mean Average Precision (MAP), following [12, 13]. In each graph, effectiveness is shown on the y-axis, while the relative space consumed compared to the Original ColBERT index (in terms of AvgDocLen@100) on the x-axis. Effectiveness values significantly different from Original are denoted with blue + symbols, according to a paired t-test ($p < 0.05$).

We first focus on RQ1, analysing the results of document-centric pruning strategies, namely *random* (black dotted line in the figures) and *IDF doc-centric* (red dashed line in the figures), while varying the number of tokens to drop from each document τ_d. As expected, for all querysets and metrics, the random approach causes a loss of effectiveness w.r.t. the baseline (red star in the figures), even for small values of τ_d, and the loss is generally statistically significant, with the only exception being observed for NDCG@10 on the TREC 2020 queryset. The IDF doc-centric approach shows superior effectiveness performance w.r.t. random, but the effectiveness degradation for MAP on TREC querysets becomes statistically significant for sufficiently values of τ_d, e.g., once 20-25% of embeddings are being removed.

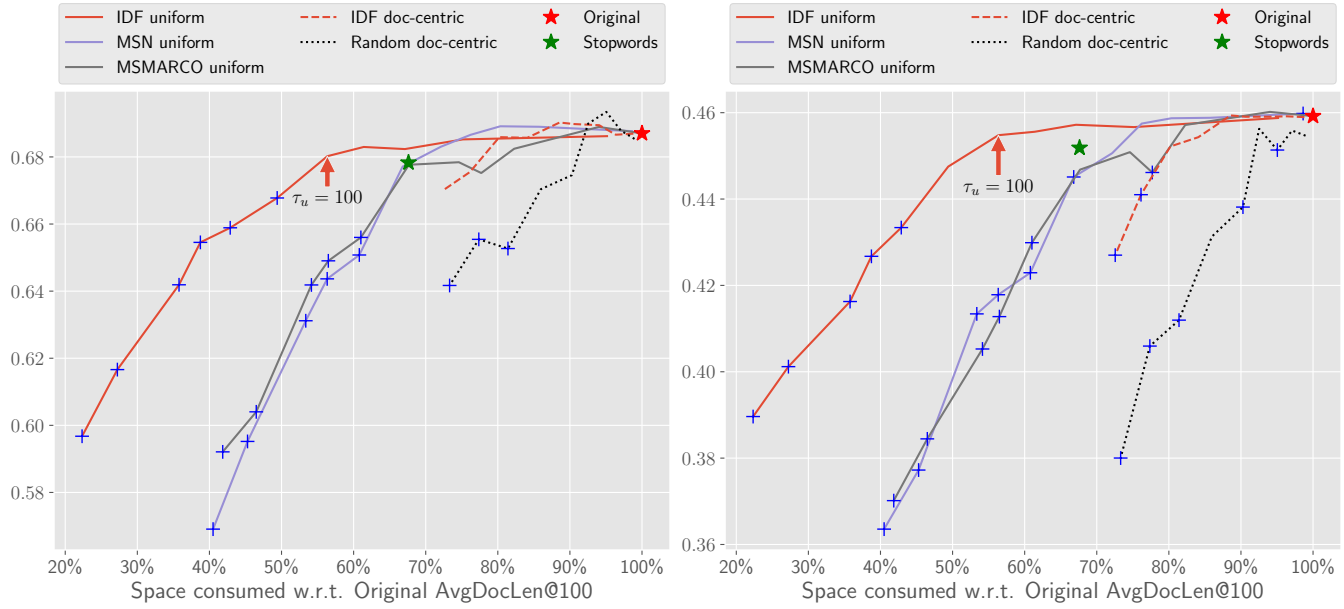
To conclude on RQ1, the IDF doc-centric pruning approach can provide only a very limited benefit in space reduction (around ~20%) without exhibiting statistically significant degradations in effectiveness.

We now turn to RQ2, analysing the results of uniform pruning strategies, namely *stopwords* (green star in the figures), *IDF uniform* (red line in the figures), *MSN uniform* (purple line in the figures), and *MSMARCO uniform* (grey line in the figures). For both querysets and



(a) nDCG@10 (b) MAP

Figure 3: NDCG@10 (left) and MAP (right) vs. space consumed w.r.t. the average document length of the first 100 retrieved documents of the baselines (no pruning, stopwords and random) and the IDF-based strategies (uniform and doc-centric) on TREC 2019, as the number of pruned tokens τ varies. Crosses denote a statistically significant difference in effectiveness compared to Original, according to a paired t-test (p-value < 0.05).



(a) nDCG@10 (b) MAP

Figure 4: NDCG@10 (left) and MAP (right) vs. space consumed w.r.t. the average document length of the first 100 retrieved documents of the baselines (no pruning, stopwords and random) and the IDF-based strategies (uniform and doc-centric) on TREC 2020, as the number of pruned tokens τ varies. Crosses denote a statistically significant difference in effectiveness compared to Original, according to a paired t-test (p-value < 0.05).

effectiveness metrics, the stopwords approach is able to prune ~32% of the embeddings of the top 100 re-ranked documents with little change in effectiveness, particularly on the TREC 2019 queryset. On the other hand, instead of removing the embeddings corresponding to 403 tokens in the stopwords list, IDF uniform is able to reach better results in terms of space occupancy by just removing the $\tau_u = 100$ tokens with the lowest IDF value in the corpus with no statistically significant difference in MAP and NDCG@10 (except NDCG@10 for TREC 2019, and the surrounding values are not significant) resulting in a pruning of ~45% of the embeddings of the top 100 re-ranked documents. On inspection, the pruned tokens for IDF $\tau_u = 100$ contains sub-word tokens not included in the classical stopword list, such as '##e' and '##ing', as well as the digits 1 - 5. This illustrates the appropriateness of IDF pruning based on the corpus itself over manually curated stopword list.

On the same querysets, both *MSN uniform* and *MSMARCO uniform* approaches are only able to prune ~35% of the embeddings of the top 100 re-ranked documents without causing statistically significant damage to effectiveness. Indeed, they are not competitive with the pruning results for *IDF uniform*. This can be explained by noting that frequent terms in the querylog (say 'google'), when pruned from the corpus could prevent relevant documents from being retrieved (thus impacting effectiveness); moreover, as they do not occur frequently in the corpus, they do not allow much pruning to actually occur.

To conclude on RQ2, the IDF uniform pruning approach allows to reduce the space occupancy of the embeddings index by almost 45%, often with no statistically significant differences in effectiveness.

5.2 RQ3 - Pruned Index Evaluation

To verify that space occupancy measured in a re-ranking setting correlates well to actual space savings for the entire index, and to ascertain whether the ANN index should be constructed based on the full index or the ANN index, we next construct two indexes for comparison with the Original index. In particular, we choose IDF uniform pruning, dropping the $\tau_u = 100$ tokens with lowest IDF from the index (which resulted in documents with an AvgDocLen@100 55% that of Original, and no significant degradations on the TREC querysets), and also the stopwords-based pruning (67%).

Table 1 reports the total index size and effectiveness of the Original unpruned ColBERT dense retrieval index on the two querysets, as well as the relative index size of the pruned indexes, and their effectiveness. For each pruned index, we report two rows: one using the original FAISS ANN index from the unpruned index, and a new FAISS index computed on the pruned index. Finally, for the two TREC querysets, we also report Recall@1000 (denoted R@1000).

Our first observation on analysis of Table 1 is that the relative decreases in the size of both the embeddings indexes and the FAISS indexes are very similar to the relative decreases in AvgDocLen@100 reported for these indexes in Section 5.1. This supports our use of AvgDocLen@100 as a proxy for the space usage of the entire index, which has marked benefits for testing different pruning settings compared to creating a new index.

Next, the index created by the distilled miniLM model (and which uses highly compressed embeddings), is, as expected, substantially smaller than all other indices (Original or pruned), but at the cost of the lowest effectiveness across all measures and querysets.

Next, we note the effectiveness results using the original FAISS are exactly inline with those reported in Figures 3 & 4. Interestingly, when using the new pruned FAISS index, we observe that applying the pruned FAISS ANN index produces slight degradations in effectiveness for most metrics (exception: stopwords index, TREC 2019). This suggests that some pruned embeddings with low IDF have a small role in retrieving relevant documents in the ANN phase, suggesting that their embeddings may be slightly contextualised. Hence, to answer RQ3, we conclude that learning a new quantised document embeddings index for first stage ANN search after static pruning can cause some small degradations in terms of effectiveness, but it reduces the ANN index size by ~45%. Should this reduction in effectiveness be undesirable, the original FAISS index can be reused with the pruned index, with a small (3-5%) overhead on space.

Finally, while not a goal of this study, we also empirically investigated the benefit of pruning on response time efficiency. We observed that while FAISS index size is reduced by 55% there is no corresponding efficiency benefit. This is because the ANN settings (number of partitions, set as a rounded function of the number of embeddings) used by ColBERT is unchanged by the reduction in embeddings. Reducing the number of partitions by 50% produced a 13% benefit in response times, but at the cost of very small decreases in effectiveness. We leave further refinement of the automatic tuning of the FAISS index partitions to future work.

Overall, the results in Table 1 demonstrate the overall efficacy of both stopword and IDF uniform pruning – embeddings that correspond to tokens with low IDF are not useful for multiple representation dense retrieval (even if they have a role in contextualising the embeddings for more important terms). Comparing stopwords and IDF uniform pruning, the latter results in smaller indices with effectiveness that is generally statistically indistinguishable in 9/12 cases.

6 VERIFICATION ON TREC-COVID

To validate the generalisation of our results, we further demonstrate the efficacy of IDF uniform pruning on the TREC-Covid test collection. The underlying CORD19 corpus contains the title & abstracts of 171k papers collected during 2020. In particular, we apply the BEIR [38] variant of the corpus, and test effectiveness using the 50 topics with relevance assessments.

In our experiments, we use the IDF values computed on MSMARCO. Indeed, initial experiments found that due to the domain-specific nature of the corpus (i.e. all documents included in the corpus were concerned with Covid-19), effectiveness when pruned using IDF computed on the corpus itself was lower, as tokens such as 'co', '19' and 'corona' were pruned, while these are important for effectiveness. Indeed, its well known that adding 'covid' or 'corona' to TREC Covid query formulations could actually improve effectiveness. We directly apply $\tau = 100$ following the results in Table 1.

Table 2 reports the resulting index sizes and effectiveness on the TREC Covid queries, using identical notation to Table 1. From the table, we make the following observations: IDF uniform pruning reduces index size more than Stopwords, but at the cost of a little more effectiveness (although no effectiveness changes are

Table 1: Index sizes and effectiveness on the TREC 2019 and TREC 2020 querysets. Pruned index sizes are expressed as a percentage of the Original index. In each column, the highest effectiveness across the pruned indexes is boldfaced. * denotes significant degradations compared to Original ($p < 0.05$).

Setting	Index Size			TREC 2019			TREC 2020		
	Total	ANN	Embeddings	nDCG@10	MAP	R@1000	nDCG@10	MAP	R@1000
Original	185 GB	16 GB	165 GB	0.706	0.419	0.671	0.687	0.459	0.733
miniLM	30%	100%	25%	0.632*	0.368*	0.625*	0.664	0.423*	0.695
Stopwords (orig. FAISS)	70%	100%	67%	0.708	**0.416**	0.671	**0.678**	0.452	**0.733**
Stopwords (new FAISS)	67%	69%	67%	**0.709**	**0.416**	**0.681**	0.674	0.449	0.713*
IDF uniform $\tau = 100$ (orig. FAISS)	60%	100%	55%	0.691*	0.407	0.680	0.674	**0.455**	**0.733**
IDF uniform $\tau = 100$ (new FAISS)	55%	54%	55%	0.690*	0.398*	0.678	0.667	0.442	0.709

Table 2: Index sizes, efficiency, effectiveness on TREC Covid. Notation as per Table 1. There are no significant differences in effectiveness compared to the Original ($p < 0.05$).

Setting	Index Size			Effectiveness	
	Total	ANN	Embeddings	nDCG@10	MAP
Original	6.5GB	5.5GB	526MB	0.680	0.154
Stopwords (orig. FAISS)	72%	100%	72%	0.685	0.151
Stopwords (new FAISS)	69%	73%	72%	**0.689**	0.143
IDF uniform $\tau = 100$ (orig. FAISS)	65%	100%	65%	0.671	**0.154**
IDF uniform $\tau = 100$ (new FAISS)	62%	64%	65%	0.672	0.147

significant at $p < 0.05$); replacing the unpruned FAISS index with a pruned index results in slight drops in MAP, but nDCG@10 remains stable - this drop mirrors the observation for the TREC 2020 MS-MARCO passage ranking dataset. Overall, we conclude that the IDF uniform pruning (with $\tau = 100$) is again useful for reducing index sizes (achieving a 38% reduction of the index), at least providing statistically indistinguishable effectiveness to an unpruned index (e.g. 1.3% reduction in nDCG@10: 0.680→0.672), and more pruning of the index compared to Stopwords removal.

7 CONCLUSIONS

In this paper, we re-examined classical static pruning approaches, and demonstrated how they could be generalised to consider embedding-based dense retrieval indexes. In particular, we proposed both document-centric and uniform pruning methods, based on the IDF of the corresponding BERT WordPiece tokens. By removing the embeddings associated with the terms with the lowest IDF, we can markedly reduce the index size, while minimising degradation in effectiveness. Indeed, our experiments on the MSMARCO v1 passage ranking corpus show that by removing embeddings corresponding to the 100 most frequent BERT tokens, the total index size is reduced by 45%, and effectiveness is only marginally reduced (e.g. without significant degradation of NDCG@10 or MAP on the TREC 2020 queryset; 4% reduction in NDCG@10 on TREC 2019 queryset). Similarly, on TREC Covid, we observed a statistically indistinguishable 1.3% reduction in nDCG@10 for a 38% reduction in total index size.

We believe that this adaptation of classical pruning techniques to the dense retrieval domain shows promise, and can be used as a strong baseline for further investigations of more advanced pruning techniques for dense retrieval. Indeed, we recognise that token-level pruning may be counter-intuitive for some frequent but polysemous words (which will have contextualised embeddings) such as 'us', which can be either a country or a pronoun. We leave such advanced pruning techniques to future work.

ACKNOWLEDGEMENTS

This work is supported, in part, by the spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing funded by European Union – NextGenerationEU, and the FoReLab project (Departments of Excellence).

REFERENCES

[1] Soner Altin, Ricardo Baeza-Yates, and B. Barla Cambazoglu. 2020. Pre-indexing Pruning Strategies. In *Proc. SPIRE*. 177–193.
[2] Ismail S. Altingovde, Rifat Ozcan, and Özgür Ulusoy. 2012. Static Index Pruning in Web Search Engines: Combining Term and Document Popularities with Query Views. *ACM TOIS* 30, 1 (2012), 1–28.
[3] Ricardo Baeza-Yates, Aristides Gionis, Flavio P. Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. 2008. Design trade-offs for search engine caching. *ACM TWEB* 2 (2008), 20:1–20:28. Issue 4.
[4] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proc. InCoCo@NIPS*.
[5] Jo Kristian Bergum. 2021. Pretrained Transformer Language Models for Search - part 4. https://blog.vespa.ai/pretrained-transformer-language-models-for-search-part-4/
[6] Roi Blanco and Alvaro Barreiro. 2007. Static Pruning of Terms in Inverted Files. In *Proc. ECIR*. 64–75.
[7] Roi Blanco and Alvaro Barreiro. 2010. Probabilistic Static Pruning of Inverted Files. *ACM TOIS* 28, 1 (2010), 1–33.
[8] Stefan Büttcher and Charles L. A. Clarke. 2006. A Document-Centric Approach to Static Index Pruning in Text Retrieval Systems. In *Proc. CIKM*. 182–189.
[9] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. 2001. Static Index Pruning for Information Retrieval Systems. In *Proc. SIGIR*. 43–50.
[10] Ruey-Cheng Chen and Chia-Jung Lee. 2013. An Information-Theoretic Account of Static Index Pruning. In *Proc. SIGIR*. 163–172.
[11] Nick Craswell, Rosie Jones, Georges Dupret, and Evelyne Viegas. 2009. Proceedings of the 2009 Workshop on Web Search Click Data.
[12] Nick Craswell, Bhaskar Mitra, Daniel Campos, and Emine Yilmaz. 2020. Overview of the TREC 2019 Deep Learning Track. In *Proc. TREC 2019*.
[13] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 Deep Learning Track. In *Proc. TREC 2020*.
[14] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proc. SIGIR*. 985–988.

[15] Edleno S. de Moura, Célia F. dos Santos, Daniel R. Fernandes, Altigran S. Silva, Pavel Calado, and Mario A. Nascimento. 2005. Improving Web Search Efficiency via a Locality Based Static Pruning Method. In *Proc. WWW*. 235–244.

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

[17] Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *OSIRRC@SIGIR*.

[18] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proc. SIGIR*. 113–122.

[19] J. Johnson, M. Douze, and H. Jegou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 03 (2021), 535–547.

[20] 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 *Proc. EMNLP*. 6769–6781.

[21] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proc. SIGIR*. 39–48.

[22] Hoang Thanh Lam, Raffaele Perego, and Fabrizio Silvestri. 2010. On Using Query Logs for Static Index Pruning. In *Proc. WI-IAT*, Vol. 1. 167–170.

[23] Carlos Lassance, Maroua Maachou, Joohee Park, and Stéphane Clinchant. 2022. Learned Token Pruning in Contextualized Late Interaction over BERT (ColBERT). In *Proc. SIGIR*. 2232–2236.

[24] Jimmy Lin. 2019. The Neural Hype, Justified! A Recantation. *SIGIR Forum* 52, 2 (2019).

[25] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond.* Morgan & Claypool Publishers.

[26] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proc. RepL4NLP Workshop*.

[27] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proc. SIGIR*. 49–58.

[28] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proc. SIGIR*. 1573–1576.

[29] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proc. SIGIR*. 1101–1104.

[30] Craig Macdonald and Nicola Tonellotto. 2021. On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieval. In *Proc. CIKM*. 3318–3322.

[31] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. On Single and Multiple Representations in Dense Passage Retrieval. In *Proc. IIR*.

[32] Irina Matveeva, Chris J. C. Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *Proc. SIGIR*. 437–444.

[33] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:1901.04085

[34] Alexandros Ntoulas and Junghoo Cho. 2007. Pruning Policies for Two-Tiered Inverted Index with Correctness Guarantee. In *Proc. SIGIR*. 191–198.

[35] Cristian Rossi, Edleno S. de Moura, Andre L. Carvalho, and Altigran S. da Silva. 2013. Fast Document-at-a-time Query Processing Using Two-tier Indexes. In *Proc. SIGIR*. 183–192.

[36] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *CoRR* abs/2112.01488 (2021). arXiv:2112.01488

[37] Gleb Skobeltsyn, Flavio Junqueira, Vassilis Plachouras, and Ricardo Baeza-Yates. 2008. ResIn: A Combination of Results Caching and Index Pruning for High-performance Web Search Engines. In *Proc SIGIR*. 131–138.

[38] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proc. NeurIPS*.

[39] Sree Lekha Thota and Ben Carterette. 2011. Within-Document Term-Based Index Pruning with Statistical Hypothesis Testing. In *Proc. ECIR*. 543–554.

[40] Nicola Tonellotto and Craig Macdonald. 2021. Query Embedding Pruning for Dense Retrieval. In *Proc. CIKM*. 3453–3457.

[41] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Foundations and Trends in Information Retrieval* 12, 4–5 (2018), 319–492.

[42] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proc. NeurIPS*, Vol. 33. 5776–5788.

[43] Xiao Wang, Craig Macdonald, and Iadh Ounis. 2022. Improving zero-shot retrieval using dense external expansion. *Information Processing & Management* 59, 5 (2022), 103026. https://doi.org/10.1016/j.ipm.2022.103026

[44] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proc. SIGIR*. 55–64.

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

[46] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proc. CIKM*. 497–506.

[47] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proc. SIGIR*. 1503–1512.