Efficient Neural Ranking Using Forward Indexes and Lightweight Encoders

Leonhardt, Jurek; Müller, Henrik; Rudra, Koustav; Khosla, Megha; Anand, Abhijit; Anand, Avishek

DOI
10.1145/3631939

Publication date
2024

Document Version
Final published version

Published in
ACM Transactions on Information Systems

Efficient Neural Ranking Using Forward Indexes and Lightweight Encoders

JUREK LEONHARDT, Delft University of Technology, The Netherlands and L3S Research Center, Germany
HENRIK MÜLLER, L3S Research Center, Germany
KOUSTAV RUDRA, Indian Institute of Technology Kharagpur, India
MEGHA KHOSLA, Delft University of Technology, Netherlands
ABHIJIT ANAND, L3S Research Center, Germany
AVISHEK ANAND, Delft University of Technology, Netherlands

Dual-encoder-based dense retrieval models have become the standard in IR. They employ large Transformer-based language models, which are notoriously inefficient in terms of resources and latency.

We propose Fast-Forward indexes—vector forward indexes which exploit the semantic matching capabilities of dual-encoder models for efficient and effective re-ranking. Our framework enables re-ranking at very high retrieval depths and combines the merits of both lexical and semantic matching via score interpolation. Furthermore, in order to mitigate the limitations of dual-encoders, we tackle two main challenges: Firstly, we improve computational efficiency by either pre-computing representations, avoiding unnecessary computations altogether, or reducing the complexity of encoders. This allows us to considerably improve ranking efficiency and latency. Secondly, we optimize the memory footprint and maintenance cost of indexes; we propose two complementary techniques to reduce the index size and show that, by dynamically dropping irrelevant document tokens, the index maintenance efficiency can be improved substantially.

We perform an evaluation to show the effectiveness and efficiency of Fast-Forward indexes—our method has low latency and achieves competitive results without the need for hardware acceleration, such as GPUs.

J. Leonhardt, K. Rudra, M. Khosla, and A. Anand, Research was primarily conducted while affiliated to L3S Research Center. This work is supported by the European Union — Horizon 2020 Program under the scheme "INFRAIA-01-2018-2019 — Integrating Activities for Advanced Communities", Grant Agreement n.871042, "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" (http://www.sobigdata.eu).
This work is supported in part by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Project SRG/2022/001548 and Microsoft Academic Partnership Grant 2023 Agreement No. 7581365. Koustav Rudra is a recipient of the DST-INSPIRE Faculty Fellowship [DST/INSPIRE/04/2021/003055] in the year 2021 under Engineering Sciences.
Authors' addresses: J. Leonhardt, Delft University of Technology, Department of Software Technology, Web Information Systems, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands and Leibniz University Hannover, L3S Research Center, Appelstr. 9a, 30167 Hannover, Germany; e-mails: L.J.Leonhardt@tudelft.nl, leonhardt@L3S.de; A. Anand, Delft University of Technology, Department of Software Technology, Web Information Systems, Van Mourik Broekmanweg 6 2628 XE Delft, The Netherlands; e-mail: avishek.anand@tudelft.nl; M. Khosla, Delft University of Technology, Department of Intelligent Systems, Multimedia Computing, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands; e-mail: M.Khosla@tudelft.nl; H. Müller and A. Anand, Leibniz University Hannover, L3S Research Center, Appelstr. 9a, 30167 Hannover, Germany; e-mails: {hmueller, aanand}@L3S.de; K. Rudra, Indian Institute of Technology Kharagpur Centre of Excellence in Artificial Intelligence, Kharagpur, West Bengal, 721302, India; e-mail: krudra@cai.iitkgp.ac.in.

1046-8188/2024/04-ART117
https://doi.org/10.1145/3631939

CCS Concepts: • **Information systems** → **Retrieval models and ranking**; *Search engine indexing*; *Search index compression;*

Additional Key Words and Phrases: Information retrieval, IR, ranking, dual-encoders, latency, efficiency

ACM Reference format:
Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2024. Efficient Neural Ranking Using Forward Indexes and Lightweight Encoders. *ACM Trans. Inf. Syst.* 42, 5, Article 117 (April 2024), 34 pages.
https://doi.org/10.1145/3631939

1 INTRODUCTION

Neural rankers are typically based on large pre-trained language models, the most popular example being BERT [14]. Due to their architectural inductive bias (like self-attention units) and complexity, these models are able to capture the semantics of documents very well, mitigating the limitations of lexical retrievers. However, their capabilities come at a price, as the models commonly used often have upwards of hundreds of millions of parameters. This makes training and even inference without specialized hardware infeasible, and it is impossible to rank all documents in a large corpus in a reasonable time. Furthermore, the resources required to run these models produce a considerable amount of emissions, creating a negative impact on the environment [74].

There are two predominant approaches to deal with the inefficiency of neural ranking models. The first one, referred to as *retrieve-and-re-rank* [25, 75], uses an efficient lexical retriever to obtain a candidate set of documents for the given query. The idea is to maximize the recall, i.e., capture most of the relevant documents, in the first stage. Afterward, the second stage employs a complex neural ranker, which *re-ranks* the documents in the candidate set, in order to promote the relevant documents to higher ranks. However, the retrieve-and-re-rank approach typically employs cross-attention re-rankers, which are expensive to compute even for a small set of candidate documents. This limits the first-stage retrieval depth, as low latency is essential for many applications (e.g., search engines).

The second approach skips the lexical retrieval step entirely and uses neural models for retrieval. The *dual-encoder* architecture employs a *query encoder* and a *document encoder*, both of which are neural models which map their string inputs to dense representations in a common vector space. Retrieval is then performed as a *k*-**nearest-neighbor** (*k***NN**) search operation to find the documents whose representations are most similar to the query. This is referred to as *dense retrieval* [34]. Representing queries and documents independently means that most of the computationally expensive processing happens during the indexing stage, where document representations are pre-computed. However, dense retrieval is still slower than lexical retrieval and benefits from GPU acceleration, because the query needs to be encoded during the query-processing phase. Furthermore, we find that dense retrievers generally have lower recall than term-matching-based models at higher retrieval depths.

In this article, we argue that neither of the two approaches is ideal. Instead, our first key idea is to explore the utility of dual-encoders in the re-ranking phase instead of the retrieval phase. Using dual-encoders in the re-ranking phase allows for a drastic reduction of query processing times and resource utilization (i.e., GPUs) during document encoding. Towards this, we first show that simple interpolation-based re-ranking that combines the benefits of lexical (computed using sparse retrieval) and semantic (computed using dual-encoders) similarity can result in competitive and sometimes better performance than using cross-attention. We propose a novel index structure called Fast-Forward indexes, which exploits the ability of dual-encoders to pre-compute document representations, in order to substantially improve the runtime efficiency of re-ranking. We

empirically establish that dual-encoder models show great performance as re-rankers, even though they do not use cross-attention.

Our second observation is that most current dual-encoder models use the same encoder for both documents and queries. While this design decision makes training easier, it also means that queries have to be encoded during runtime using a, potentially expensive, forward pass. We argue that this is suboptimal; rather, queries, which are often short and concise, do not require a complex encoder to compute their representations. We propose lightweight query encoders, some of which do not contain any self-attention layers, and show that they still perform well as re-rankers, while requiring only a fraction of the resources and time. In this work, we propose two families of lightweight query encoders to drastically reduce query-encoding costs without compromising ranking performance.

Lastly, we focus on the aspects of *index footprint* and *index maintenance*. Since dense indexes store the pre-computed representations of documents in the corpus, they exhibit much higher storage and memory requirements compared to sparse indexes [30]. At the same time, maintaining the index, i.e., adding new documents, requires expensive forward passes of the document encoder. We propose two means of reducing the memory footprint: On the one hand, we propose *sequential coalescing* to compress an index by reducing the number of vectors that need to be stored; on the other hand, we experiment with choosing a smaller number of dimensions, which reduces the size of each vector. Finally, we propose efficient document encoders, which dynamically drop irrelevant tokens prior to indexing using a very simple technique.

Our research questions are as follows:

RQ1 How suitable are dual-encoder models for interpolation-based re-ranking in terms of performance and efficiency?
RQ2 Can the re-ranking efficiency be improved by limiting the number of Fast-Forward look-ups?
RQ3 To what extent does query encoder complexity affect re-ranking performance?
RQ4 What is the tradeoff between Fast-Forward index size and ranking performance?
RQ5 Can the indexing efficiency be improved by removing irrelevant document tokens?

We conduct extensive experimentation on existing ranking benchmarks and find that dual-encoder models are very suitable for interpolation-based re-ranking and exhibit highly desirable performance and efficiency tradeoffs. We show that, with further optimizations (*early stopping*—cf. Section 4.2), re-ranking efficiency can be greatly improved by limiting the number of Fast-Forward look-ups. Additionally, we report a good tradeoff between Fast-Forward index size and ranking performance by using our novel *sequential coalescing* algorithm (cf. Section 4.1). Our experiments show that we can indeed train extremely lightweight query encoders without adversely affecting ranking performance. Specifically, our most lightweight query encoders are orders of magnitude faster than $\text{BERT}_{\text{base}}$ models with little performance degradation. More importantly, we can migrate query-processing to CPUs instead of relying on GPUs, improving on the environmental impact. Finally, we show that we can reduce index maintenance costs by around 50% by dynamically removing irrelevant document tokens. Our code is publicly available.

Note that this article extends our previously published work [44], where we introduced Fast-Forward indexes along with the *sequential coalescing* and *early stopping* techniques. This article introduces the following new aspects:

(1) We identify the query encoder as an efficiency bottleneck of Fast-Forward indexes and propose lightweight query encoders.
(2) We show that the dimensionality of queries and documents can be reduced in order to reduce index size and compute dot products faster.

(3) We propose a *selective document encoder* that dynamically identifies irrelevant document tokens and drops them prior to indexing, reducing index maintenance cost.
(4) We perform additional experiments, including analyses of the tradeoffs between efficiency and performance. We discuss the limitations of our method and its out-of-domain performance.

2 RELATED WORK

Classical ranking approaches, such as BM25 [70] or the query likelihood model [41], rely on the inverted index that stores term-level statistics like term frequency, inverse document frequency and positional information. We refer to this style of method as *sparse*, since it assumes sparse document representations. The recent success of large pre-trained language models (e.g., BERT) shows that *semantic* or contextualized information is essential for many language tasks. In order to incorporate such information in the relevance measurement, Dai and Callan [12, 13] proposed DEEP-CT, which stores contextualized scores for terms in the inverted index for text ranking. SPLADE [18] aims at enriching sparse document representations using a trained contextual Transformer model and sparsity regularization on the term weights. Similarly, DeepImpact [61] enriches the document collection with expansion terms to learn improved term impacts. In our work, we employ efficient sparse models for high-recall first-stage retrieval and perform re-ranking using semantic models in a subsequent step.

The ability to accurately determine semantic similarity is essential in order to alleviate the vocabulary mismatch problem [11, 13, 57, 59, 64]. Computing the semantic similarity of a document given a query has been heavily researched in IR using smoothing methods [37], topic models [84], embeddings [63], personalized models [56], and so on. In these classical approaches, ranking is performed by interpolating the semantic similarity scores with the lexical matching scores from the first-stage retrieval. More recently, *dense* neural ranking methods, which employ large pre-trained language models, have become increasingly popular. Dense rankers do not explicitly model terms, but rather compute low-dimensional dense vector representations through self-attention mechanisms in order to estimate relevance; this allows them to perform semantic matching. However, the inherent complexity of dense ranking models usually has a negative impact on latency and cost, especially with large corpora. Therefore, besides performance, efficiency has been another major concern in developing neural ranking models.

There are two common architectures of dense ranking models: *Cross-attention* models take a concatenation of a query and a document as input. This allows them to perform query-document attention in order to compute the corresponding relevance score. These models are typically used as re-rankers. *Dual-encoder models* employ two language models to independently encode queries and documents as fixed-size vector representations. Usually, a similarity metric between query and document vector determines their relevance. As a result, dual-encoders are mostly used for dense retrieval, but also, less commonly, for re-ranking.

We divide the remainder of the related work section into subcategories for cross-attention models, dual-encoder models, and *hybrid models*, which employ both lexical and semantic rankers. Finally, we briefly cover inference efficiency for BERT-based models.

2.1 Cross-Attention Models

The majority of cross-attention approaches have been dominated by large contextual models [1, 10, 27, 29, 45, 58]. The input to these ranking models is a concatenation of the query and document. This combined input results in higher query processing times since each document has to be processed in conjugation with the query string. Thereby, cross-attention models usually re-rank a relatively small number of potentially relevant candidates retrieved in the first stage by

efficient sparse methods. The expensive re-ranking computation cost is then proportional to the retrieval depth (e.g., 1000 documents).

Another key limitation of using cross-attention models for document ranking is the maximum acceptable number of input tokens for Transformer models, which exhibit quadratic complexity w.r.t. input length. Some strategies address this limitation by document truncation [58], or chunking documents into passages [10, 72]. However, the performance of chunking-based strategies depends on the chunking properties, i.e., passage length or overlap among consecutive passages [73]. Recent proposals include a two-stage approach, where a query-specific summary is generated by selecting relevant parts of the document, followed by re-ranking strategies over the query and summarized document [28, 43, 46, 48]. Due to the efficiency concerns, we do not consider cross-attention methods in our work but focus on dual-encoders instead.

2.2 Dual-Encoders

Dual-encoders learn dense vector representations for queries and documents using contextual models [34, 35]. The dense vectors are then indexed in an offline phase [32], where retrieval is akin to performing an **approximate nearest neighbor** (**ANN**) search given a vectorized query. This allows dual-encoders to be used for both retrieval and re-ranking. Consequently, there has been a large number of follow-up works that boost the performance of dual-encoder models by improving pre-training [5, 20, 21, 39, 82], optimization [23], and negative sampling [68, 86, 88] techniques, or employing distillation approaches [51, 54, 90]. Lindgren et al. [53] propose a *negative cache* that allows for efficient training of dual-encoder models. LED [89] uses a SPLADE model to enrich a dense encoder with lexical information. Lin et al. [50] propose Aggretriever, a dual-encoder model which aggregates and exploits all token representations (instead of only the classification token). In this work, we use dual-encoders for computing semantic similarity between queries and passages. Some approaches have also proposed architectural modifications to the aggregations between the query and passage embeddings [6, 27, 31]. Nogueira et al. [67] propose a simple document expansion model. We use dual-encoder models to perform efficient semantic re-ranking in our work.

Efficiency improvements of dual-encoder-based ranking and retrieval focus mostly on either inference efficiency of the encoders or memory footprint of the indexes. TILDE [92] and TILDEv2 [91] efficiently re-rank documents using a deep query and document likelihood model instead of a query encoder. The SpaDE model [7] employs a *dual document encoder* that has a *term weighting* and *term expansion* component; it improves inference efficiency by using a vastly simplified query representation. Li et al. [47] employ *dynamic lexical routing* in order to reduce the number of dot products in the late interaction step. Cohen et al. [8] use auto-encoders to compress document representations into fewer dimensions in order to reduce the overall size. Dong et al. [15] propose an approach to split documents into variable-length segments and dynamically merge them based on similarity, such that each document has the same number of segments prior to indexing. Hofstätter et al. [26] introduce ColBERTer, an extension of ColBERT [35], which removes irrelevant word representations in order to reduce the number of stored vectors. In a similar fashion, Lassance et al. [40] propose a *learned token pruning* approach, which is also used to reduce the size of ColBERT indexes by dropping tokens that are deemed irrelevant. Yang et al. [87] propose a *contextual quantization* approach for pre-computed document representations (such as the ones used by ColBERT) by compressing document-specific representations of terms.

In most of the previous work, dual-encoders are used in a *homogeneous* or *symmetric* fashion, meaning that both the query and document encoders have the same architecture or even share weights (*Siamese* encoders). Jung et al. [33] show that the characteristics of queries and documents are different and employ *light fine-tuning* in order to adapt each encoder to its specific role.

Kim et al. [36] use model distillation for asymmetric dual-encoders, where the query encoder has fewer parameters than the document encoder. Lassance and Clinchant [38] separate the query and document encoder of SPLADE models in order to improve efficiency. In this work, we explore the use of light-weight query encoders for more efficient re-ranking.

2.3 Hybrid Models

Hybrid models combine sparse and dense retrieval. The most common approach is a simple linear combination of both scores [51]. CLEAR [23] takes the relevance of the lexical retriever into account in the loss function of the dense retriever. COIL [22] performs contextualized exact matching using pre-computed document token representations. COILcr [17] extends this approach by factorizing token representations and approximating them using canonical representations in order to make retrieval more efficient.

Unlike classical methods, where score interpolation is the norm, semantic similarity from neural contextual models (e.g., cross-attention or dual-encoders) is not consistently combined with the matching score. Recently, Wang et al. [83] showed that the interpolation of BERT-based models and lexical retrieval methods can boost the performance. Furthermore, they analyze the role of interpolation in BERT-based dense retrieval strategies and find that dense retrieval alone is not enough, but interpolation with BM25 scores is necessary. Similarly, Askari et al. [2] find that even providing the BM25 score as part of the input text improves the re-ranking performance of BERT models.

2.4 Inference Efficiency

Several methods have been proposed to improve the inference efficiency of large Transformer-based models, which have quadratic time complexity w.r.t. the input length. PoWER-BERT [24] progressively eliminates word vectors in the subsequent encoder layers in order to reduce the input size. DeeBERT [85] implements an *early-exit* mechanism, which may stop the computation after any Transformer layer based on the entropy of its output distribution. SkipBERT [81] uses a technique where intermediate Transformer layers can be skipped dynamically using pre-computed look-up tables. We use a simple Selective BERT approach which dynamically removes irrelevant document tokens in order to make document encoding more efficient.

3 PRELIMINARIES

In this section, we introduce core concepts that are essential to this work, such as retrieval, re-ranking, and interpolation.

3.1 Interpolation-based Re-Ranking

The retrieval of documents or passages given a query often happens in two stages [75]: In the first stage, a term frequency-based (**sparse**) retrieval method (such as BM25 [71]) retrieves a set of documents from a large corpus. In the second stage, another model, which is usually much more computationally expensive, **re-ranks** the retrieved documents again.

In **sparse retrieval**, we denote the top-k_S documents retrieved from the sparse index for a query q as K_S^q. The sparse score of a query-document pair (q, d) is denoted by $\phi_S(q, d)$. For the **re-ranking** part, we focus on self-attention models (such as BERT [14]) in this work. These models operate by creating (internal) high-dimensional dense representations of queries and documents, focusing on their semantic structure. We refer to the outputs of these models as **dense** or **semantic** scores and denote them by $\phi_D(q, d)$. Due to the quadratic time complexity of self-attention w.r.t. the document length (and decreasing performance with increasing document length [55]), long documents are often split into passages, and the score of a document is then computed as the maximum of its

passage scores:

$$\phi_D(q, d) = \max_{p_i \in d} \phi_D(q, p_i). \tag{1}$$

This approach is referred to as *maxP* [10].

The retrieval approach for a query q starts by retrieving K_S^q from the sparse index. For each retrieved document $d \in K_S^q$, the corresponding dense score $\phi_D(q, d)$ is computed. This dense score may then be used to re-rank the retrieved set to obtain the final ranking. However, it has been shown that the scores of the sparse retriever, ϕ_S, can be beneficial for re-ranking as well [1]. To that end, an interpolation approach is employed [4], where the final score of a query-document pair is computed as

$$\phi(q, d) = \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \phi_D(q, d). \tag{2}$$

Setting $\alpha = 0$ recovers the standard re-ranking procedure.

Since the set of documents retrieved by the sparse model is typically large (e.g., $k_S = 1,000$), computing the dense score for each query-document pair can be very computationally expensive. In this article, we focus on efficient implementations of interpolation-based re-ranking, specifically the computation of the dense scores ϕ_D.

3.2 Dual-Encoder Models

The *dual-encoder* architecture [34] employs neural semantic models to compute *dense vector representations* of queries and documents. Specifically, a *query encoder* ζ and a *document encoder* η map queries and documents to representations in a common a-dimensional vector space. The relevance score $\phi_D(q, d)$ of a query-document pair is then computed as the similarity of their vector representations. A common choice for the similarity function is the dot product, such that

$$\phi_D(q, d) = \zeta(q) \cdot \eta(d),$$

where $\zeta(q), \eta(d) \in \mathbb{R}^a$.

3.2.1 Dense Retrieval. Dual-encoder models are commonly utilized to perform **dense retrieval** [34]. A *dense index* contains pre-computed vector representations $\eta(d)$ for all documents d in the corpus $\mathcal{D}$. To retrieve a set of documents K_D^q for a query q, a kNN search is performed to find the documents whose representations are most similar to the query:

$$K_D^q = \underset{1 \le i \le |\mathcal{D}|}{k\text{-argmax}}(\zeta(q) \cdot \eta(d_i)).$$

In order to make dense retrieval more efficient, ANN search is commonly employed [32, 60]. ANN search can be further accelerated using special hardware, such as GPUs [32].

3.2.2 Training. In contrast to *cross-encoder* models, which are often used for re-ranking (cf. Section 3.1), dual-encoders encode the query and document *independently*, i.e., there is no query-document attention. Typically, dual-encoders for retrieval are trained using a *contrastive* loss function [34],

$$\mathcal{L}(q, d^+, D^-) = -\log\left(\frac{\exp\left(\phi(q, d^+; \theta)/\tau\right)}{\sum_{d \in D^- \cup \{d^+\}} \exp\left(\phi(q, d; \theta)/\tau\right)}\right), \tag{3}$$

where a training instance consists of a query q, a positive (relevant) document d^+, and a set D^- of negative (irrelevant) documents. The temperature τ is a hyperparameter. Since it is usually infeasible to include all negative documents for a query in D^-, there are various *negative sampling* approaches, such as distillation [51], asynchronous indexes [86], or negative caches [53]. In this work, we use a simple *in-batch* strategy [34], where, for a query q, D^- contains a number of *hard negatives* (retrieved by BM25) along with all documents from the other queries in the same training batch.

3.3 Hybrid Retrieval

Hybrid retrieval [23, 51] is similar to interpolation-based re-ranking (cf. Section 3.1). The key difference is that the dense scores $\phi_D(q, d)$ are not computed for all query-document pairs. Instead, ϕ_D is a dense retrieval model (cf. Section 3.2.1), which retrieves documents d_i and their scores $\phi_D(q, d_i)$ using nearest neighbor search given a query q. A hybrid retriever combines the retrieved sets of a sparse and a dense retriever.

For a query q, we retrieve two sets of documents, K_S^q and K_D^q, using the sparse and dense retriever, respectively. Note that the two retrieved sets are usually not equal. One strategy proposed in [51] ranks all documents in $K_S^q \cup K_D^q$, approximating missing scores. In our experiments, however, we found that **only** considering documents from K_S^q for the final ranking and discarding the rest works well. The final score is thus computed as

$$\phi(q, d) = \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \begin{cases} \phi_D(q, d) & d \in K_D^q \\ \phi_S(q, d) & d \notin K_D^q \end{cases}.$$

The re-ranking step in hybrid retrieval is essentially a sorting operation over the interpolated scores and takes negligible time in comparison to standard re-ranking.

4 FAST-FORWARD INDEXES

The hybrid approach described in Section 3.3 has two distinct disadvantages. Firstly, in order to retrieve K_D^q, an (approximate) nearest neighbor search has to be performed, which is time consuming. Secondly, some of the query-document scores are expected to be missed, leading to an incomplete interpolation, where the score of one of the retrievers needs to be approximated [52] for a number of query-document pairs.

In this section, we propose Fast-Forward indexes as an efficient way of computing dense scores for known documents that alleviates the aforementioned issues. Specifically, Fast-Forward indexes build upon dual-encoder dense retrieval models that compute the score of a query-document pair as a dot product

$$\phi_D(q, d) = \zeta(q) \cdot \eta(d),$$

where ζ and η are the query and document encoders, respectively. Examples of such models are ANCE [86] and TCT-ColBERT [52]. Since the query and document representations are independent for two-tower models, we can pre-compute the document representations $\eta(d)$ for each document d in the corpus. These document representations are then stored in an efficient hash map, allowing for look-ups in constant time. After the index is created, the score of a query-document pair can be computed as

$$\phi_D^{FF}(q, d) = \zeta(q) \cdot \eta^{FF}(d),$$

where the superscript FF indicates the look-up of a pre-computed document representation in the Fast-Forward index. At retrieval time, only $\zeta(q)$ needs to be computed once for each query. As queries are usually short, this can be done on CPUs. The main benefit of this method is that the number of documents to be re-ranked can be much higher than with cross-attention models; the scoring operation is a simple look-up and dot product computation.

Note that the use of large Transformer-based query encoders still remains a bottleneck in terms of latency (or, if it is run on GPUs, cost). In Section 5, we focus on lightweight encoder models.

4.1 Index Compression via Sequential Coalescing

A major disadvantage of dense indexes and dense retrieval in general is the size of the final index. This is caused by two factors: Firstly, in contrast to sparse indexes, the dense representations cannot be stored as efficiently as sparse vectors. Secondly, the dense encoders are typically

ALGORITHM 1: Compression of dense maxP indexes by sequential coalescing

Input: list of passage vectors P (original order) of a document, distance threshold δ
Output: coalesced passage vectors P'

$P' \leftarrow$ empty list
$\mathcal{A} \leftarrow \emptyset$
foreach v **in** P **do**
  if *first iteration* **then**
    `// do nothing`
  else if `cosine_distance`$(v, \overline{\mathcal{A}}) \geq \delta$ **then**
    append $\overline{\mathcal{A}}$ to P'
    $\mathcal{A} \leftarrow \emptyset$
  add v to $\mathcal{A}$
  $\overline{\mathcal{A}} \leftarrow$ `mean`$(\mathcal{A})$
end
append $\overline{\mathcal{A}}$ to P'
return P'

Transformer-based, imposing a (soft) limit on their input lengths due to their quadratic time complexity with respect to the inputs. Thus, long documents are split into passages prior to indexing (*maxP* indexes).

As an increase in the index size has a negative effect on efficiency, both for nearest neighbor search and Fast-Forward indexing as used by our approach, we exploit a *sequential coalescing* approach as a way of dynamically combining the representations of consecutive passages within a single document in maxP indexes. The idea is to reduce the number of passage representations in the index for a single document. This is achieved by exploiting the *topical locality* that is inherent to documents [42]. For example, a single document might contain information regarding multiple topics; due to the way human readers naturally ingest information, we expect documents to be authored such that a single topic appears mostly in consecutive passages, rather than spread throughout the whole document. Our approach aims at combining consecutive passage representations that encode similar information. To that end, we employ the cosine distance function and a *threshold* parameter δ that controls the degree of coalescing. Within a single document, we iterate over its passage vectors in their original order and maintain a set $\mathcal{A}$, which contains the representations of the already processed passages, and continuously compute $\overline{\mathcal{A}}$ as the average of all vectors in $\mathcal{A}$. For each new passage vector v, we compute its cosine distance to $\overline{\mathcal{A}}$. If it exceeds the distance threshold δ, the current passages in $\mathcal{A}$ are combined as their average representation $\overline{\mathcal{A}}$. Afterward, the combined passages are removed from $\mathcal{A}$ and $\overline{\mathcal{A}}$ is recomputed. This approach is illustrated in Algorithm 1. Figure 1 shows an example index after coalescing. To the best of our knowledge, there are no other forward index compression techniques proposed in literature so far.

4.2 Faster Interpolation by Early Stopping

As described in Section 3.1, by interpolating the scores of sparse and dense retrieval models, we perform implicit re-ranking, where the dense representations are pre-computed and can be looked up in a Fast-Forward index at retrieval time. Furthermore, increasing the sparse retrieval depth k_S, such that $k_S > k$, where k is the final number of documents, improves the performance. A drawback of this is that an increase in the number of retrieved documents also results in an increase in the number of index look-ups.

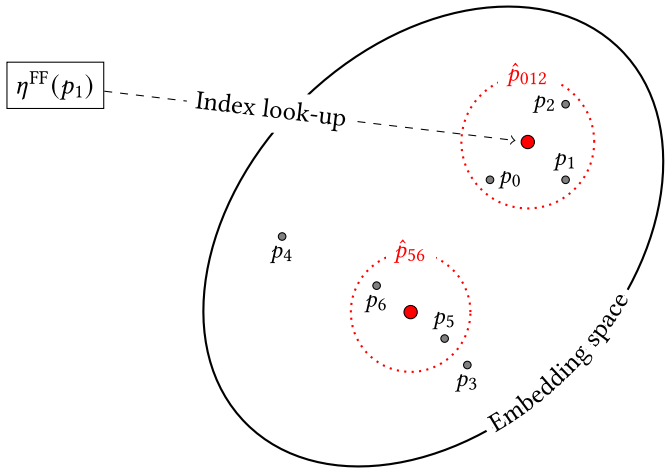


Fig. 1. Sequential coalescing combines the representations of similar consecutive passages as their average. Note that p_3 and p_5 are not combined, as they are not consecutive passages.

ALGORITHM 2: Interpolation with early stopping

Input: query q, sparse retrieval depth k_S, cut-off depth k, interpolation parameter α
Output: approximated top-k scores Q

$Q \leftarrow$ priority queue of size k
$s_D \leftarrow -\infty$
$s_{min} \leftarrow -\infty$
foreach d **in** $\texttt{sparse}(q, k_S)$ **do**
  if Q *is full* **then**
    $s_{min} \leftarrow$ remove smallest item from Q
    $s_{best} \leftarrow \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot s_D$
    if $s_{best} \leq s_{min}$ **then**
      `// early stopping`
      put s_{min} into Q
      break
  `// approximate max. dense score`
  $s_D \leftarrow \max(\phi_D(q, d), s_D)$
  $s \leftarrow \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \phi_D(q, d)$
  put $\max(s, s_{min})$ into Q
end
return Q

Common term pruning mechanisms for term-at-a-time retrieval, such as MaxScore [79] or WAND [3], accelerate query processing for inverted-index-based retrievers; however, these techniques are not compatible with neural ranking models based on contextual query and document representations. Our use case is more similar to *top-k query evaluation*, with algorithms such as the *threshold algorithm* [16] or probabilistic approximations [77], but these approaches usually require sorted access, which is not available for the dense re-ranking scores in our case.

In this section, we propose an extension to Fast-Forward indexes that allows for *early stopping*, i.e., avoiding a number of unnecessary look-ups, for cases where $k_S > k$ by approximating the

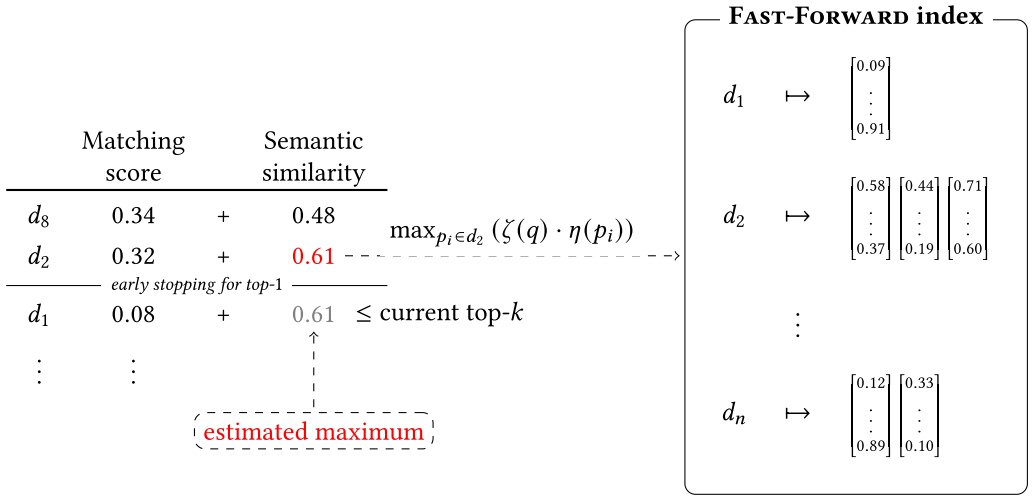


Fig. 2. Early stopping reduces the number of interpolation steps by computing an approximate upper bound for the dense scores. This example depicts the most extreme case, where only the top-1 document is required.

maximum possible dense score. The early stopping approach takes advantage of the fact that documents are ordered by their sparse scores $\phi_S(q, d)$. Since the number of retrieved documents, k_S, is finite, there exists an upper limit s_D for the corresponding dense scores such that $\phi_D(q, d) \leq s_D \forall d \in K_S^q$. Since the retrieved documents K_S^q are ordered by their sparse scores, we can simultaneously perform interpolation and re-ranking by iterating over the ordered list of documents: Let d_i be the ith highest ranked document by the sparse retriever. Recall that we compute the final score as

$$\phi(q, d_i) = \alpha \cdot \phi_S(q, d_i) + (1 - \alpha) \cdot \phi_D(q, d_i).$$

If $i > k$, we can compute the upper bound for $\phi(q, d_i)$ by exploiting the aforementioned ordering:

$$s_{best} = \alpha \cdot \phi_S(q, d_{i-1}) + (1 - \alpha) \cdot s_D.$$

In turn, this allows us to stop the interpolation and re-ranking if $s_{best} \leq s_{min}$, where s_{min} denotes the score of the kth document in the current ranking (i.e., the currently lowest ranked document). Intuitively, this means that we stop the computation once the *highest possible* interpolated score $\phi(q, d_i)$ is too low to make a difference. The approach is illustrated in Algorithm 2 and Figure 2. Since the dense scores ϕ_D are usually unnormalized, the upper limit s_D is unknown in practice. We thus approximate it by using the highest observed dense score at any given step.

4.2.1 Theoretical Analysis. We first show that the early stopping criteria, when using the true maximum of the dense scores, is sufficient to obtain the top-k scores.

THEOREM 4.1. *Let s_D, as used in Algorithm 2, be the true maximum of the dense scores. Then the returned scores are the actual top-k scores.*

PROOF. First, note that the sparse scores, $\phi_S(q, d_i)$, are already sorted in decreasing order for a given query. By construction, the priority queue Q always contains the highest scores corresponding to the list parsed so far. Let, after parsing k scores, Q be full. Now the possible best score s_{best} is computed using the sparse score found next in the decreasing sequence and the maximum of all dense scores, s_D (cf. Algorithm 7). If s_{best} is less than the minimum of the scores in Q, then Q already contains the top-k scores. To see this, note that the first component of s_{best} is the largest

among all unseen sparse scores (as the list is sorted) and s_D is the maximum of the dense scores by our assumption. □

Next, we show that a good approximation of the top-k scores can be achieved by using the sample maximum. To prove our claim, we use the **Dvoretzky—Kiefer—Wolfowitz (DKW)** [62] inequality.

Lemma 4.2. *Let $X_1, X_2, \ldots, X_n$ be n real-valued independent and identically distributed random variables with the cumulative distribution function $F(\cdot)$. Let $F_n(\cdot)$ denote the empirical cumulative distributive function, i.e.,*

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq x\}}, \quad x \in \mathbb{R}.$$

According to the DKW inequality, the following estimate holds:

$$\Pr\left(\sup_{x \in \mathbb{R}} (F_n(x) - F(x)) > \epsilon\right) \leq e^{-2n\epsilon^2} \forall \epsilon \geq \sqrt{\frac{1}{2n} \ln 2}.$$

In the following, we show that, if s_D is chosen as the maximum of a large random sample drawn from the set of dense scores, then the probability that any given dense score, chosen independently and uniformly at random from the dense scores, is greater than s_D is exponentially small in the sample size.

Theorem 4.3. *Let $x_1, x_2, \ldots, x_n$ be a real-valued independent and identically distributed random sample drawn from the distribution of the dense scores with the cumulative distribution function $F(\cdot)$. Let $z = \max(x_1, x_2, \ldots, x_n)$. Then, for every $\epsilon > \frac{1}{\sqrt{2n}} \ln 2$, we obtain*

$$\Pr(F(z) < 1 - \epsilon) \leq e^{-2n\epsilon^2}. \tag{4}$$

Proof. Let $F_n(\cdot)$ denote the empirical cumulative distribution function as above. Specifically, $F_n(x)$ is equal to the fraction of variables less than or equal to x. We then have $F_n(z) = 1$. By Lemma 4.2, we infer

$$\Pr(F_n(z) - F(z) > \epsilon) \leq e^{-2n\epsilon^2}.$$

Substituting $F_n(z) = 1$, we obtain Equation (4). □

This implies that the probability of any random variable X, chosen randomly from the set of dense scores, being less than or equal to s_D, is greater than or equal to $1-\epsilon$ with high probability, i.e.,

$$\Pr(P_D(X \leq s_D) \geq 1 - \epsilon) \geq 1 - e^{-2n\epsilon^2},$$

where P_D denotes the probability distribution of the dense scores. This means that, as our sample size grows until it reaches k, the approximation improves. Note that, in our case, the dense scores are sorted (by corresponding sparse score) and thus the i.i.d. assumption cannot be ensured. However, we observed that the dense scores are positively correlated with the sparse scores. We argue that, due to this correlation, we can approximate the maximum score well.

5 EFFICIENT ENCODERS

BERT models are the de facto standard for both query and document encoders [34, 51, 86]. The encoders are often *homogeneous*, meaning that the architectures of both models are identical, or even *Siamese*, i.e., the same encoder weights are used for both queries and documents. Other approaches are *semi-Siamese* models [33], where *light fine-tuning* is used to adapt each encoder to its input characteristics, or TILDE [92] and TILDEv2 [91], which do not require dense query representations. However, the most common choice remains the use of $\text{BERT}_{\text{base}}$ for both encoders.

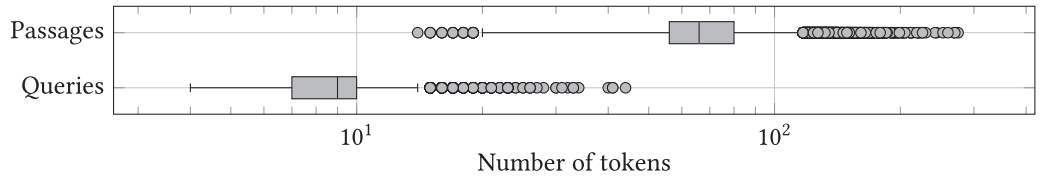


Fig. 3. The distribution of query and passage lengths in the MS MARCO corpus. The statistics are computed based on the development set queries and the first 10 000 passages from the corpus using a $\text{BERT}_{\text{base}}$ tokenizer.

In this article, we argue that the homogeneous structure is not ideal for dual-encoder IR models w.r.t. query processing efficiency, since the characteristics of queries and documents differ [33]. We illustrate those characteristics w.r.t. the average number of tokens in Figure 3. This section focuses on model architectures for both query and document encoding that aim at improving the overall efficiency of the ranking process.

5.1 Lightweight Query Encoders

Query encoders need to be run online during query processing, i.e., the representations cannot be pre-computed. Consequently, query encoding latency is essential for many downstream applications, such as search engines. Our experiments reveal that even encoding a large batch of 256 queries using a $\text{BERT}_{\text{base}}$ model on CPU takes more than 3 seconds (cf. Figure 7(b)), resulting in roughly 12 milliseconds per query (smaller batch sizes or even single queries lead to even slower encoding). Since queries are typically short and concise, we argue that query encoders require lower complexity (e.g., in terms of the number of parameters) than document encoders. Our proposed query encoders are considerably more lightweight than standard $\text{BERT}_{\text{base}}$ models, and thus more efficient in terms of latency and resources.

5.1.1 Attention-based. Attention-based query encoders (such as models based on BERT [14]) use Transformer encoder layers [80] to compute query representations. Each of these layers has two main components—*multi-head attention* and a feed-forward sub-layer—both of which include residual connections and layer normalization operations.

Attention is computed based on three input matrices—the *queries* $\mathbf{Q}$, *keys* $\mathbf{K}$, and *values* $\mathbf{V}$:

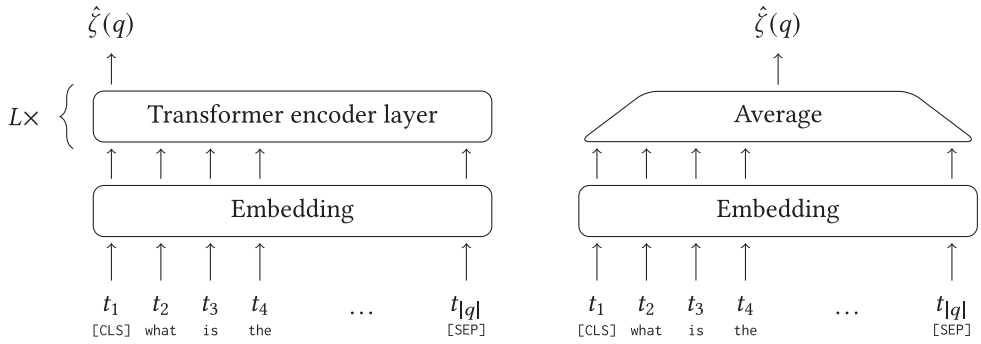
$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

Multi-head attention computes attention multiple times (using A *attention heads* h_i) and concatenates the results, as denoted by $\circ$, i.e.,

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (h_1 \circ \ldots \circ h_A)\,\mathbf{W}^O,$$
$$\text{where } h_i = \text{Attn}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right).$$

The matrices $\mathbf{W}_i^Q \in \mathbb{R}^{H\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{H\times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{H\times d_v}$, and $\mathbf{W}^O \in \mathbb{R}^{Ad_v\times H}$ are trainable parameters, H denotes the dimension of hidden representations in the model, and $d_k = \frac{H}{A}$ is a scaling factor.

Since Transformer encoders compute self-attention, the three inputs $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ originate from the same place, i.e., they are projections of the output of the previous encoder layer. The inputs to the first encoder layer originate from a token embedding layer. We denote the embedding operation as $E : \mathbb{N} \mapsto \mathbb{R}^H$, such that $E(t)$ is the embedding vector of a token t.



(a) Attention-based query encoder (b) Embedding-based query encoder

Fig. 4. The query-encoder types used in this work. Note that the positional encoding that is added to BERT input tokens has been omitted in this figure.

Given a BERT-based encoder and a query $q = (t_1, \ldots, t_{|q|})$, where t_i are `WordPiece` tokens, the query representation is computed as

$$\hat{\zeta}_{\text{Attn}}(q) = \text{BERT}_{\text{CLS}}(\texttt{[CLS]}, t_1, \ldots, t_{|q|}, \texttt{[SEP]}),$$

where BERT_{CLS} indicates that the output vector corresponding to the *classification token*, denoted by `[CLS]`, is used. Figure 4(a) shows attention-based query encoders.

The usual choice for query encoders, $\text{BERT}_{\text{base}}$, has $L = 12$ layers, $H = 768$ dimensions for hidden representations and $A = 12$ attention heads. In this work, we investigate how less complex query encoders impact the re-ranking performance. Specifically, we vary three hyperparameters, namely the number of Transformer layers L, hidden dimensions H and attention heads A. The pre-trained BERT models we use are provided by Turc et al. [78].

5.1.2 Embedding-based. Embedding-based query encoders can be seen as a special case of BERT-based query encoders (cf. Section 5.1.1). Setting $L = 0$, we obtain a model without any Transformer encoder layers; what's left is only the token embedding layer E.

Due to the omission of self-attention (and thus, contextualization) altogether, the usage of the `[CLS]` token is not feasible for this approach. Instead, a query $q = (t_1, \ldots, t_{|q|})$ is represented simply as the average of its token embeddings, i.e.,

$$\hat{\zeta}_{\text{Emb}}(q) = \frac{\sum_{t_i \in q} E(t_i)}{|q|}.$$

Embedding-based query encoders are illustrated in Figure 4(b).

5.2 Selective Document Encoders

Document encoders are not run during query processing time, since document representations are pre-computed and indexed. However, the computation of document representations still requires a substantial amount of time and resources. This is particularly important for applications like web search, where *index maintenance* plays an important role, usually due to large amounts of new documents constantly needing to be added to the index. The effect is further amplified by the maxP approach (cf. Equation (1)), where long documents require more than one encoding step. Since documents tend to be much longer and more complex than queries, lightweight document encoders would likely negatively affect performance, and recent research suggests that larger document encoders lead to better results [66]. However, due to the nature of documents obtained from

web pages, we expect a considerable number of document tokens to be irrelevant for the encoding step; examples for this are stop words or redundant (repeated) information. Similar observations have been made in other approaches [26]. Furthermore, recent research [69] has shown that certain aspects, such as the position of tokens, are not essential for large language models to perform well. Our proposed document encoders assign a *relevance score* to each input token and dynamically drop low-scoring tokens before computing self-attention in order to make the document encoding step more efficient.

We refer to this approach as SELECTIVE BERT. It uses a *scoring network* $\Phi : \mathbb{N} \mapsto [0, 1]$ to determine the relevance of each input token before feeding it into the encoding BERT model Ψ. We denote the parameters of the scoring network as θ_Φ and the parameters of the BERT model as θ_Ψ. We use a lightweight, non-contextual scoring network with three 384-dimensional feed-forward layers and ReLU activations. The final layer outputs a scalar that is fed into a sigmoid activation function to compute the final score. SELECTIVE BERT models are trained in two steps.

5.2.1 Pre-Training. The **first step** pre-trains the scoring network. θ_Ψ is initialized using the weights of a pre-trained BERT model (e.g., $\text{BERT}_{\text{base}}$), and θ_Φ is initialized randomly. The complete model is then trained for a single epoch using the same data as during the unsupervised BERT pre-training step [14]. The scoring network Φ is taken into account by multiplying the embedding of an input token t_i by its corresponding score, i.e.,

$$x_i = E(t_i) \cdot \Phi(t_i) + P(t_i),$$

where $E(t_i)$ is the token embedding and $P(t_i)$ is the positional encoding. The resulting representation x_i is then used to compute self-attention in the first encoder layer.

In order to encourage the scoring network to output scores less than one, we introduce a regularization term using the L_1-norm over the scores, where n is the input sequence length:

$$\ell_1 = \sum_{i=0}^{n} \Phi(t_i).$$

The final objective is a combination of the original BERT pre-training loss $\mathcal{L}$ and the scoring regularizer scaled by a hyperparameter λ:

$$\min_{\theta_\Psi, \theta_\Phi} \left[\mathcal{L}(\theta_\Psi, \theta_\Phi) + \lambda \cdot \ell_1(\theta_\Phi) \right].$$

5.2.2 Fine-Tuning and Inference. The **second step**, referred to as *fine-tuning*, only trains the BERT model Ψ, while the scoring network Φ remains frozen for the remainder of the training process. Furthermore, the weights of the BERT model obtained in the previous step, θ_Ψ, are discarded and replaced by the same pre-trained model as before. The training objective during this stage is identical to that of other dual-encoder models (cf. Section 3.2.2).

During fine-tuning and inference (i.e., document encoding), we only retain the tokens with the highest scores; we set a ratio $p \in [0; 1]$ of the original input length to retain. As a result, the length of the input batch is shortened by $1-p$. This is achieved by removing the lowest scoring tokens from the input. Since individual documents within a batch are usually padded, p always corresponds to the longest sequence in the batch. Consequently, padding tokens are always removed first before the scores of the other tokens are taken into account. The process is illustrated in Figure 5.

6 EXPERIMENTAL SETUP

In this section, we outline the experimental setup, including baselines, datasets, and further details about training and evaluation.

6.1 Baselines

We consider the following baselines:

(1) **Sparse retrievers** rely on term-based matching between queries and documents. We consider BM25, which uses term-based retrieval signals. DEEP-CT [12], SPLADE [18], and SpaDE [7] use sparse representations, but contextualize terms in some fashion.
(2) **Dense retrievers** retrieve documents that are semantically similar to the query in a common embedding space. We consider TCT-ColBERT [52], ANCE [86], and the more recent Aggretriever [50]. All three approaches are based on BERT encoders. Large documents are split into passages before indexing (maxP). These dense retrievers use exact (brute-force) nearest neighbor search as opposed to ANN search. We evaluate these methods in both the retrieval and re-ranking settings.
(3) **Hybrid retrievers** interpolate sparse and dense retriever scores. We consider CLEAR [23], a retrieval model that complements lexical models with semantic matching. Additionally, we consider the hybrid strategy described in Section 3.3 as a baseline, using the dense retrievers above.
(4) **Re-rankers** operate on the documents retrieved by a sparse retriever (e.g., BM25). Each query-document pair is input into the re-ranker, which outputs a corresponding score. In this article, we use a BERT-CLS re-ranker, where the output corresponding to the classification token is used as the score. Note that re-ranking is performed using the full documents (i.e., documents are not split into passages). If an input exceeds 512 tokens, it is truncated. Furthermore, we consider TILDEv2 [91] with TILDE expansion.

6.2 Datasets

We evaluate our models and baselines on a variety of diverse retrieval datasets:

(1) The **TREC Deep Learning track** [9] provides test sets and relevance judgments for retrieval and ranking evaluation on the MS MARCO corpora [65]. We use both the passage and document ranking test sets from the years 2019 and 2020 for our experiments. In addition, we use the MS MARCO development sets to determine the optimal values for hyperparameters.
(2) The **BEIR benchmark** [76] is a collection of various IR datasets, which are commonly evaluated in a *zero-shot* fashion, i.e., without using any of the data for training the model. We evaluate our models on a subset of the BEIR datasets, including tasks such as passage retrieval, question answering, and fact checking.

6.3 Evaluation Details

Our ranking experiments are performed on a single machine using an Intel Xeon Silver 4210 CPU and an NVIDIA Tesla V100 GPU. In our initial experiments (Tables 3 and 5), we measured the per-query latency by performing each experiment four times and reporting the average latency, excluding the first measurement. In subsequent experiments (Table 4 and Figures 7(a) and 10(a)), we adjusted our way of measuring; we perform multiple runs of each experiment, where each run contains multiple latency measurements. We then report the average overall measurements of the fastest run. In Tables 3 and 4, latency is reported as the sum of scoring (this includes operations like encoding queries and documents, obtaining representations from a Fast-Forward index, computing the scores as dot-products, and so on), interpolation (cf. Equation (2)), and sorting cost. Any pre-processing or tokenization cost is ignored. Where applicable, dense models use a batch size of 256. The first-stage (sparse) retrieval step is not included, as it is constant for all methods. The Fast-Forward indexes are loaded into the main memory entirely before they are accessed. In Table 5, we report end-to-end latency, which includes retrieval, re-ranking, and tokenization cost.

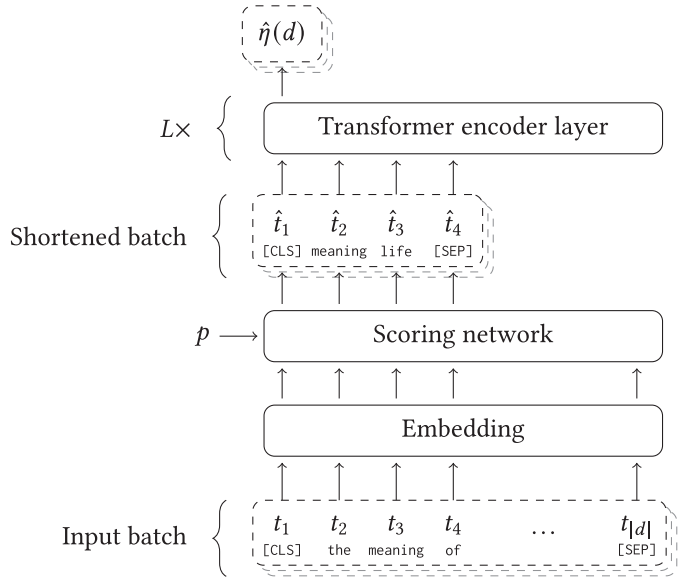


Fig. 5. The fine-tuning and inference phase of Selective BERT document encoders. In the given example, the documents in the input batch are dynamically shortened to four tokens each based on the corresponding relevance scores. Note that the positional encoding that is added to BERT input tokens has been omitted in this figure.

Table 1. The Pre-trained Dense Encoders and Corresponding Indexes We Used in Our Experiments

	MS MARCO (documents)	MS MARCO (passages)
ANCE	`castorini/ance-msmarco-doc-maxp` `msmarco-doc-ance-maxp-bf`	`castorini/ance-msmarco-passage` `msmarco-passage-ance-bf`
TCT-ColBERT	`castorini/tct_colbert-msmarco` `msmarco-doc-tct_colbert-bf`	`castorini/tct_colbert-msmarco` `msmarco-passage-tct_colbert-bf`
Aggretriever	-	`castorini/aggretriever-cocondenser` `msmarco-v1-passage.aggretriever-cocondenser`

In each cell, the first line corresponds to a pre-trained encoder (to be obtained from the HuggingFace Hub) and the second line is a pre-built index provided by Pyserini.

We use the Pyserini [49] toolkit, which provides a number of pre-trained encoders (available on the *HuggingFace Hub*[1]) and corresponding indexes (see Table 1), for our retrieval experiments. Dense encoders (ANCE, TCT-ColBERT, and Aggretriever) output 768-dimensional representations. The sparse BM25 retriever is provided by Pyserini as well. We use the pre-built indexes `msmarco-passage` ($k_1 = 0.82$, $b = 0.68$) and `msmarco-doc` ($k_1 = 4.46$, $b = 0.82$). Furthermore, we use Pyserini to run SPLADE with the provided `msmarco-passage-distill-splade-max` index and the pre-trained DistilSPLADE-max model.

We use the MS MARCO development set to determine the interpolation parameter α. We set $\alpha = 0.2$ for TCT-ColBERT, $\alpha = 0.5$ for ANCE, and $\alpha = 0.7$ for BERT-CLS (Section 7.1). For Aggretriever, we set $\alpha = 0.3$ for BM25 re-ranking and $\alpha = 0.1$ for SPLADE re-ranking. For the dual-encoder models we trained ourselves (Section 7.3–7.5), the value for α is determined based

[1] https://huggingface.co/models

on nDCG@10 re-ranking results on the MS MARCO development set and varies slightly for each model.

6.4 Training Details

Our dual-encoder models are trained using the contrastive loss in Equation (3). For each training instance, we sample 8 hard negative documents using BM25. Additionally, we use in-batch negatives and a batch size of 4, resulting in $|D^-| = 32$ negatives for each query. Each model is trained on four NVIDIA A100 GPUs. We set the learning rate to $1 \cdot 10^{-5}$ and use gradient accumulation of 32 batches (this results in an effective batch size of $4 \cdot 4 \cdot 32 = 512$). During training, we perform validation on the MS MARCO development set. Our models are trained until the average precision stops improving for five consecutive iterations. We exclusively train on the MS MARCO passage ranking corpus; the resulting models are then evaluated on multiple datasets (i.e., for BEIR, we do zero-shot evaluation). Our Selective BERT model (cf. Section 5.2) uses $\lambda = 10^{-6}$ during pre-training. We implemented our models and training pipeline using PyTorch,[2] PyTorch-Lightning,[3] and Transformers.[4]

6.4.1 Dual-Encoder Architecture. Our dual-encoder rankers consist of a query encoder ζ and a document encoder η (cf. 3.2):

$$\zeta(q) = ||\mathbf{W}_\zeta \hat{\zeta}(q) + b_\zeta||_2,$$
$$\eta(d) = ||\mathbf{W}_\eta \hat{\eta}(d) + b_\eta||_2.$$

The models $\hat{\zeta}$ and $\hat{\eta}$ map queries and documents to arbitrary vector representations; examples for these models are pre-trained Transformers or the encoders described in Section 5. We include optional trainable linear layers (with corresponding weights $\mathbf{W}_\zeta \in \mathbb{R}^{a \times d_\zeta}$, $\mathbf{W}_\eta \in \mathbb{R}^{a \times d_\eta}$, $b_\zeta \in \mathbb{R}^a$, and $b_\eta \in \mathbb{R}^a$) for heterogeneous encoders, where the dimensions of the representation vectors, d_ζ and d_η, do not match. We further L_2-normalize the representations during training and indexing; we do not normalize the query representations during ranking, as this would only scale the scores, but not change the final ranking.

7 RESULTS

In this section, we perform experiments to show the effectiveness and efficiency of Fast-Forward indexes. Each subsection corresponds to one of our research questions.

7.1 How Suitable Are Dual-Encoder Models for Interpolation-based Re-Ranking in Terms of Performance and Efficiency?

This section focuses on the effectiveness and efficiency of Fast-Forward indexes for re-ranking. We use pre-trained dual-encoders that are homogeneous (i.e., both encoders are identical models) for our experiments.

7.1.1 Interpolation-based Re-Ranking Performance of Dual-Encoder Models. In Table 2, we report the performance of sparse, dense and hybrid retrievers, re-rankers and interpolation.

First, we observe that dense retrieval strategies perform better than sparse ones in terms of nDCG, but have poor recall except on TREC-DL-Psg'19. The contextual weights learned by DEEP-CT are better than tf-idf-based retrieval (BM25), but fall short of dense semantic retrieval strategies (TCT-ColBERT and ANCE) with differences upwards of 0.1 in nDCG. However, the overlap among

[2] https://pytorch.org/
[3] https://pytorchlightning.ai/
[4] https://huggingface.co/

Table 2. Ranking Performance

	TREC-DL-Doc'19			TREC-DL-Doc'20			TREC-DL-Psg'19		
	AP_{1k}	R_{1k}	$nDCG_{10}$	AP_{1k}	R_{1k}	$nDCG_{10}$	AP_{1k}	R_{1k}	$nDCG_{10}$
Sparse Retrieval									
BM25	0.331	0.697	$0.519^{[abc]}$	0.404	0.809	$0.527^{[abc]}$	0.301	0.750	$0.506^{[abc]}$
DEEP-CT	-	-	0.544	-	-	-	0.422	0.756	0.551
Dense Retrieval									
TCT-ColBERT	0.279	0.576	$0.612^{[a]}$	0.372	0.728	$0.586^{[ab]}$	0.391	0.792	0.670
ANCE	0.254	0.510	$0.633^{[a]}$	0.401	0.681	0.633	0.371	0.755	0.645
Hybrid Retrieval									
CLEAR	-	-	-	-	-	-	0.511	0.812	0.699
Re-Ranking									
TCT-ColBERT	0.370	0.697	0.685	0.414	0.809	0.617	0.423	0.750	0.694
ANCE	0.336	0.697	0.654	0.426	0.809	0.630	0.389	0.750	0.679
BERT-CLS	0.283	0.697	$0.520^{[abc]}$	0.329	0.809	$0.522^{[abc]}$	0.353	0.750	$0.578^{[ab]}$
Interpolation									
[a] TCT-ColBERT	0.406	0.697	0.696	0.469	0.809	0.637	0.438	0.750	0.708
[b] ANCE	0.387	0.697	0.673	0.490	0.809	0.655	0.417	0.750	0.680
[c] BERT-CLS	0.365	0.697	0.612	0.460	0.809	0.626	0.378	0.750	0.617

Retrievers use depths $k_S = 1000$ (sparse) and $k_D = 10000$ (dense). Dense retrievers retrieve passages and perform maxP aggregation for documents. Scores for CLEAR and DEEP-CT are taken from the corresponding articles [22, 23]. Superscripts indicate statistically significant improvements using two-paired tests with a sig. level of 95% [19].

retrieved documents is rather low, reflecting that dense retrieval cannot match query and document terms well.

Second, dual-encoder-based (TCT-ColBERT and ANCE) perform better than contextual (BERT-CLS) re-rankers. In this setup, we first retrieve $k_S = 1,000$ documents using a sparse retriever and re-rank them. This approach benefits from high recall in the first stage and promotes the relevant documents to the top of the list through the dense semantic re-ranker. However, re-ranking is typically time-consuming and requires GPU acceleration. The improvements of TCT-ColBERT and ANCE over BERT-CLS (e.g., 0.1 in nDCG) also suggest that dual-encoder-based re-ranking strategies are better than cross-interaction-based methods. However, the difference could also be attributed to the fact that BERT-CLS does not follow the maxP approach (cf. Section 3.1).

Finally, interpolation-based re-ranking, which combines the benefits of sparse and dense scores, significantly outperforms the BERT-CLS re-ranker and dense retrievers. Recall that dense re-rankers operate solely based on the dense scores and discard the sparse BM25 scores of the query-document pairs. The superiority of interpolation-based methods is also supported by evidence from recent studies [5, 6, 22, 23].

7.1.2 Efficient Re-Ranking at Higher Retrieval Depths. Tables 3 and 4 show results of re-ranking, hybrid retrieval, and interpolation on document and passage datasets, respectively. The metrics are computed for two sparse retrieval depths, $k_S = 1,000$ and $k_S = 5,000$.

We observe that additionally taking the sparse component into account in the score computation (as is done by the interpolation and hybrid methods) causes performance to improve with retrieval depth. Specifically, some queries receive a considerable recall boost, capturing more relevant documents with large retrieval depths. Interpolation based on Fast-Forward indexes achieves substantially lower latency compared to other methods. Pre-computing the document representations allows for fast look-ups during retrieval time. As only the query needs to be encoded by the dense model, both retrieval and re-ranking can be performed on the CPU while still offering considerable

Table 3. Document Ranking Performance

	Latency	TREC-DL-Doc'19						TREC-DL-Doc'20					
		k_S = 1000			k_S = 5000			k_S = 1000			k_S = 5000		
	ms	AP_{1k}	R_{1k}	$nDCG_{20}$	AP_{1k}	R_{1k}	$nDCG_{20}$	AP_{1k}	R_{1k}	$nDCG_{20}$	AP_{1k}	R_{1k}	$nDCG_{20}$
Hybrid Retrieval													
BM25, TCT-ColBERT	0 + 582	0.394	0.697	0.655	0.385	0.729	0.645	0.463	0.809	0.615	0.469	0.852	0.621
BM25, ANCE	0 + 582	0.379	0.697	0.633	0.373	0.727	0.628	0.479	0.809	0.624	0.488	0.846	0.632
Re-Ranking													
TCT-ColBERT	1189 + 2	0.370	0.697	0.632	0.334	0.703	0.609[a]	0.414	0.809	0.587[a]	0.405	0.794	0.585[acd]
ANCE	1189 + 2	0.336	0.697	0.614	0.304	0.647	0.607	0.426	0.809	0.595[c]	0.422	0.761	0.604
BERT-CLS	185 + 2	0.283	0.697	0.494[abcde]	0.159	0.559	0.289	0.329	0.809	0.512[abcde]	0.221	0.727	0.375[abcde]
Interpolation													
[a] TCT-ColBERT	1189 + 14	0.406	0.697	0.655	0.411	0.745	0.653	0.469	0.809	0.621	0.478	0.838	0.626
[a] ↳ Fast-Forward	0 + 253	0.406	0.697	0.655	0.411	0.745	0.653	0.469	0.809	0.621	0.478	0.838	0.626
[b] ↳ coalesced	0 + 109	0.379	0.697	0.630	0.379	0.732	0.625	0.440	0.809	0.594[a]	0.447	0.837	0.607
[c] ANCE	1189 + 14	0.387	0.697	0.638	0.393	0.732	0.639	0.490	0.809	0.630	0.502	0.828	0.640
[c] ↳ Fast-Forward	0 + 253	0.387	0.697	0.638	0.393	0.732	0.639	0.490	0.809	0.630	0.502	0.828	0.640
[d] ↳ coalesced	0 + 121	0.372	0.697	0.625	0.375	0.723	0.628	0.471	0.809	0.622	0.479	0.823	0.629
[e] BERT-CLS	185 + 14	0.365	0.697	0.585	0.357	0.708	0.562	0.460	0.809	0.602	0.459	0.839	0.601

Latency is reported per query for k_S = 5000 on GPU and CPU. The coalesced Fast-Forward indexes are compressed to approximately 25% of their original size. Hybrid retrievers use a dense retrieval depth of k_D = 1000. Superscripts indicate statistically significant improvements using two-paired tests with a sig. level of 95% [19].

Table 4. Ranking Performance on TREC-DL-Psg'19

	Latency	k_S = 1000		k_S = 5000	
	ms	AP_{1k}	RR_{10}	AP_{1k}	RR_{10}
Hybrid Retrieval					
BM25, TCT-ColBERT	0 + 307	0.434	0.894	0.454	0.902
BM25, ANCE	0 + 307	0.410	0.856	0.422	0.864
Re-Ranking					
TCT-ColBERT	186 + 2	0.426	0.827	0.439	0.842
ANCE	186 + 2	0.389	0.836	0.392	0.857
BERT-CLS	185 + 2	0.353	0.715	0.275	0.576
Interpolation					
TCT-ColBERT	186 + 14	0.438	0.894	0.460	0.902
↳ Fast-Forward	0 + 114	0.438	0.894	0.460	0.902
↳ early stopping	0 + 72	-	0.894	-	0.902
ANCE	186 + 14	0.417	0.856	0.435	0.864
↳ Fast-Forward	0 + 114	0.417	0.856	0.435	0.864
↳ early stopping	0 + 52	-	0.856	-	0.864
BERT-CLS	185 + 14	0.378	0.809	0.392	0.832

Latency is reported per query for k_S = 5000 on GPU and CPU. Hybrid retrievers use a dense retrieval depth of k_D = 1000.

improvements in query processing time. Note that for BERT-CLS, the input length is limited, causing documents to be truncated, similarly to the *firstP* approach. As a result, the latency is much lower, but in turn the performance suffers. It is important to note here, that, in principle, Fast-Forward indexes can also be used in combination with firstP models.

The hybrid retrieval strategy, as described in Section 3.3, shows good performance. However, as the dense indexes require nearest neighbor search for retrieval, the query processing latency is much higher than for interpolation using Fast-Forward indexes.

Finally, dense re-rankers do not profit reliably from increased sparse retrieval depth; on the contrary, the performance drops in some cases. This trend is more apparent for the document retrieval datasets with higher values of k_S. We hypothesize that dense rankers only focus on semantic matching and are sensitive to topic drift, causing them to rank irrelevant documents in the top-5000 higher.

7.1.3 Varying the First-Stage Retrieval Model. We perform additional passage ranking experiments in Table 5, where we compare various first-stage retrieval methods in combination with re-rankers. The idea is to show how Fast-Forward indexes perform in combination with modern sparse retrievers and how they compare with other re-rankers. Additionally, these experiments give an idea of the *end-to-end* efficiency, as we report the latency as the sum of retrieval, re-ranking, and tokenization. The Aggretriever model [50] we use in combination with Fast-Forward indexes is a recent single-vector dual-encoder model based on coCondenser [21].

Both SpaDE and SPLADE, unsurprisingly, perform substantially better than BM25, as these models use contextualized learnt representations. This boost in performance comes with a large increase in latency, in terms of both indexing and query processing. However, it becomes evident

Table 5. Passage Ranking Performance Using Various First-stage Retrieval Models as Well as Re-rankers

	Latency	MSM-Psg-Dev		TREC-DL-Psg'19			TREC-DL-Psg'20		
	ms	AP_{1k}	RR_{10}	AP_{1k}	RR_{10}	$nDCG_{10}$	AP_{1k}	RR_{10}	$nDCG_{10}$
BM25	14	0.196	0.187	0.301	0.702	0.506	0.288	0.655	0.488
↳ TILDEv2	104	0.338	0.342	0.437	0.836	0.680	0.459	0.868	0.679
↳ Aggretriever									
↳ Fast-Forward	150	0.373	0.369	0.465	0.877	0.700	0.486	0.825	0.717
SpaDE ($k = 5$)	-	-	0.355	0.437	-	0.682	0.453	-	0.677
SPLADE	302	0.375	0.368	0.485	0.901	0.728	0.490	0.830	0.711
↳ TILDEv2	374	0.337	0.342	0.412	0.808	0.654	0.433	0.858	0.648
↳ Aggretriever									
↳ Fast-Forward	420	0.383	0.378	0.489	0.899	0.726	0.500	0.856	0.716

Aggretriever models are used for interpolation-based re-ranking using Fast-Forward indexes. Re-ranking is done with $k_S = 5000$ passages. SpaDE results are taken from the corresponding article [7]. For SPLADE, we use the DistilSPLADE-max model. Latency is reported per query on CPU. For retrieval models (BM25 and SPLADE), latency is reported at retrieval depth $k_S = 1000$. For re-ranking (TILDEv2 and Fast-Forward), latency is reported as the sum of retrieval and re-ranking, both at depth $k_S = 5000$.

that re-ranking BM25 results comes very close to these models in terms of performance, and sometimes even surpasses them, even though the overall latency remains lower. At the same time, Fast-Forward indexes manage to improve the performance of SPLADE by re-ranking (although the improvements are not as big). Interestingly, TILDEv2 does not exhibit this behavior, but rather performs worse when a SPLADE first-stage retriever is used. We assume that the reason for this is that the model was not optimized for this scenario.

7.2 Can the Re-Ranking Efficiency be Improved by Limiting the Number of Fast-Forward Look-Ups?

We evaluate the utility of the early stopping approach described in Section 4.2 on the TREC-DL-Psg'19 dataset. Figure 6 shows the average number of look-ups performed in the Fast-Forward index during interpolation w.r.t. the cut-off depth k. We observe that, for $k = 100$, early stopping already leads to a reduction of almost 20% in the number of look-ups. Decreasing k further leads to a significant reduction of look-ups, resulting in improved query processing latency. As lower cut-off depths (i.e., $k < 100$) are typically used in downstream tasks, such as question answering, the early stopping approach for low values of k turns out to be particularly helpful.

Table 4 shows early stopping applied to the passage dataset to retrieve the top-10 passages and compute reciprocal rank. It is evident that, even though the algorithm approximates the maximum dense score (cf. Section 4.2), the resulting performance is identical, which means that the approximation was accurate in both cases and did not incur any performance hit. Furthermore, the query processing time is decreased by up to half compared to standard interpolation. This means that presenting a small number of top results (as is common in many downstream tasks) can yield substantial speed-ups. Note that early stopping depends on the value of α, hence the latency varies between TCT-ColBERT and ANCE.

7.3 To What Extent Does Query Encoder Complexity Affect Re-Ranking Performance?

In this section, we investigate the role of the query encoder in interpolation-based re-ranking using Fast-Forward indexes.

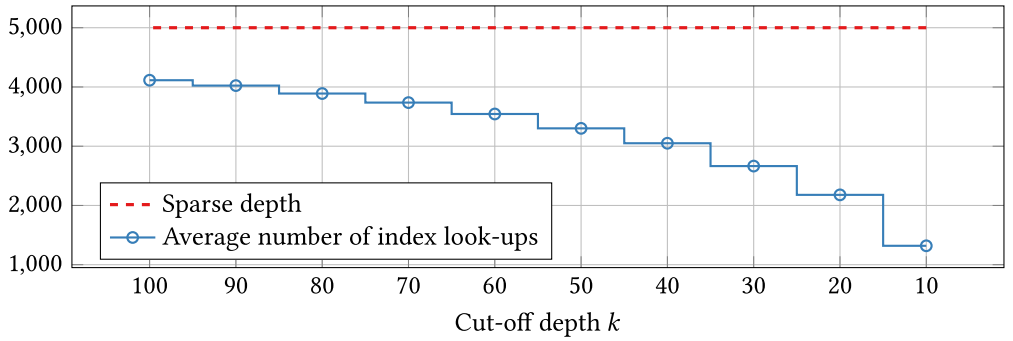


Fig. 6. The average number of FAST-FORWARD index look-ups per query for interpolation with early stopping at varying cut-off depths k on TREC-DL-PSG'19 with $k_S = 5000$ using ANCE.

7.3.1 The Role of Self-Attention. First, we train a large number of dual-encoder models (as described in Section 6.4) and successively reduce the complexity of the query encoder. At the same time, we monitor the effects on performance and latency. The query encoders we analyze correspond to the *attention-based query encoders* in Section 5.1.1 and the *embedding-based query encoders* in Section 5.1.2. Since the embedding-based encoders are, technically speaking, a special case of the attention-based ones, we plot the results together in Figure 7. The document encoder we use is a $\text{BERT}_{\text{base}}$ model, which has $L = 12$ layers and $H = 768$ hidden dimensions; it is the same across all experiments. For the query encoder, we start with $\text{BERT}_{\text{base}}$ as well and reduce both the number of layers and hidden dimensions. All pre-trained BERT models we use for this experiment are provided by Turc et al. [78]. If the output dimensions of the encoders do not match, we add a single linear layer to the query encoder (cf. Section 6.4.1).

Figure 7(a) illustrates the time each encoder requires to encode a batch of queries on a CPU; as expected, a reduction in either the number of layers or hidden dimensions has a positive impact on encoding latency, and the most lightweight attention-based model ($L = 2, H = 128$) is significantly faster than $\text{BERT}_{\text{base}}$ (27 milliseconds vs. 3.1 seconds). Furthermore, the complete omission of self-attention in the embedding-based encoder ($L = 0, H = 768$) results in even faster encoding (13 milliseconds).

Next, we analyze to what extent the drastic reduction of complexity affects the ranking performance. Figure 7(b)–(d) shows the corresponding FAST-FORWARD re-ranking performance on passage development and test sets. It is evident that the absolute difference in performance between the encoders is relatively low; this is especially true on MSM-PSG-DEV and TREC-DL-PSG'19. In fact, the embedding-based query encoder does not always yield worse performance than the attention-based encoders, specifically on TREC-DL-PSG'19. On TREC-DL-PSG'20, the highest absolute difference of 0.05 is the largest among the three datasets.

These results suggest that query encoders do not need to be overly complex; rather, in most cases, either considerably smaller attention-based or even embedding-based models can be used. The embedding-based encoders are particularly useful since they are essentially a look-up table and hence require no forward pass other than computing the average of all token embeddings.

7.4 What is the Tradeoff between FAST-FORWARD Index Size and Ranking Performance?

This research question investigates how index size influences ranking performance and latency. In detail, we reduce index size in two different ways: First, we apply sequential coalescing (cf. Section 4.1) in order to reduce the *number of vector representations* in the index. Second, we train

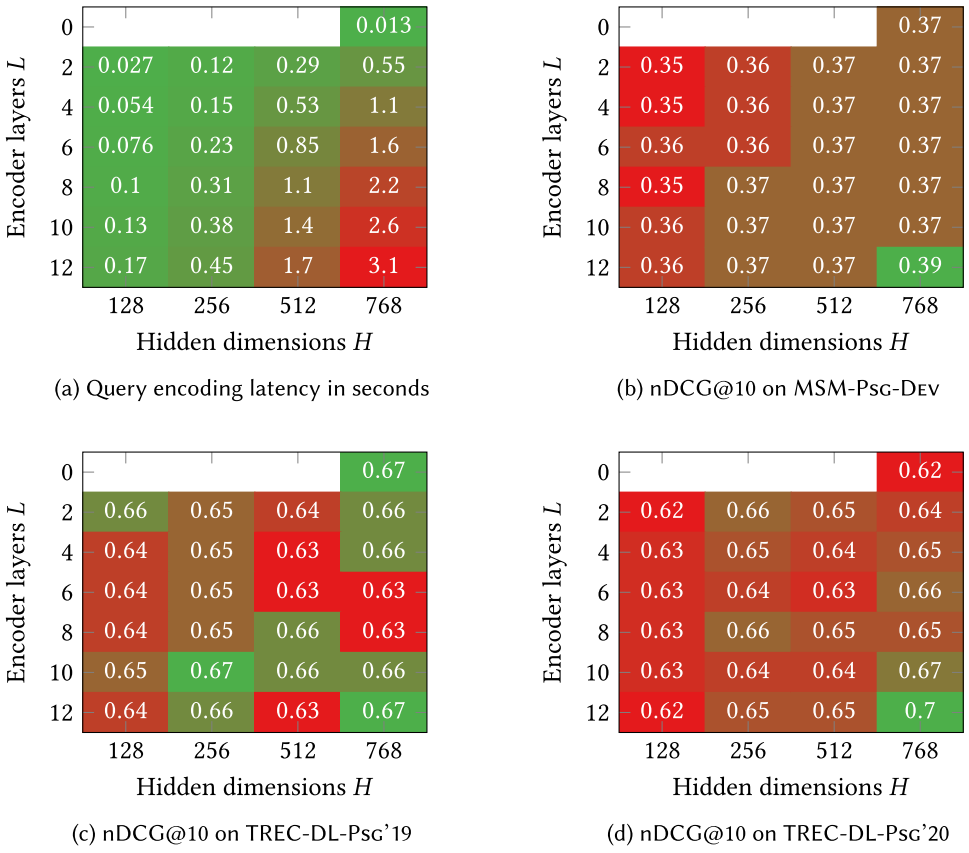


Fig. 7. Query encoding latency and Fast-Forward ranking performance of dual-encoders with various query encoder models. The sparse retrieval depth is $k_S = 5000$. L and H correspond to the number of Transformer layers and dimensions of the hidden representations, respectively. $L = 0$ corresponds to embedding-based query encoders, which are initialized with pre-trained token embeddings from $\text{BERT}_{\text{base}}$, and $L > 0$ corresponds to attention-based query encoders, where the number of attention heads is $A = \frac{H}{64}$. The document encoder is a BERT model with 12 layers and 768-dimensional representations in all cases. Query encoding latency is measured on CPU with a batch size of 256 queries from MSM-Psg-Dev (tokenization cost is excluded, as it is identical for all models).

query and encoders to output *lower-dimensional vector representations*. Note that these methods are not mutually exclusive, but rather complementary.

7.4.1 Sequential Coalescing. In order to evaluate this approach, we first take the pre-trained TCT-ColBERT dense index of the MS MARCO corpus, apply sequential coalescing with varying values for δ and evaluate each resulting compressed index using the TREC-DL-Doc'19 test set.

The results are illustrated in Figure 8. It is evident that, by combining the passage representations, the number of vectors in the index can be reduced by more than 80% in the most extreme case, where only a single vector per document remains. At the same time, the performance is correlated with the granularity of the representations. However, the drops are relatively small. For example, for $\delta = 0.025$, the index size is reduced by more than half, while the nDCG decreases by roughly 0.015 (3%).

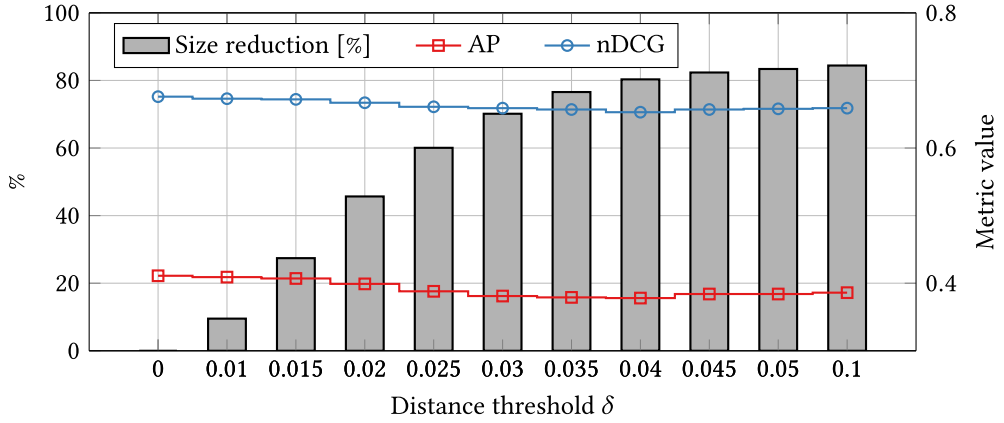


Fig. 8. Sequential coalescing applied to TREC-DL-Doc'19. The plot shows the index size reduction in terms of the number of passages and the corresponding metric values for Fast-Forward interpolation with TCT-ColBERT.

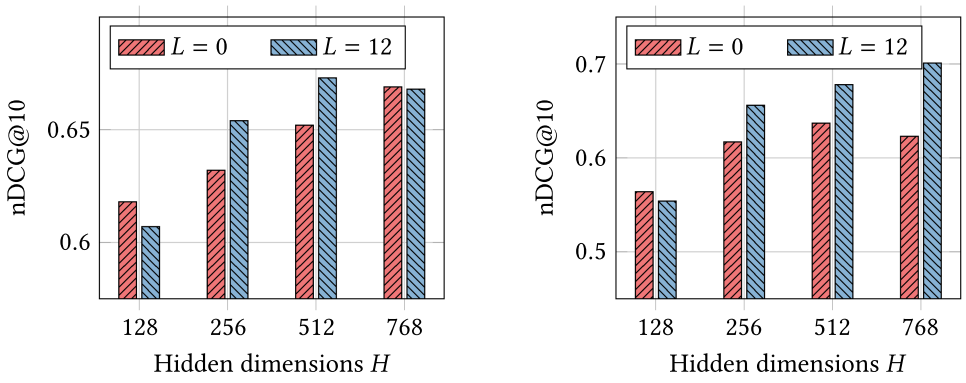
Additionally, Table 3 shows the detailed performance of coalesced Fast-Forward indexes on the document datasets. We chose the indexes corresponding to $\delta = 0.035$ (TCT-ColBERT) and $\delta = 0.003$ (ANCE), both of which are compressed to approximately 25% of their original size. This is reflected in the query processing latency, which is reduced by more than half. The overall performance drops to some extent, as expected, however, these drops are not statistically significant in all but one case. The tradeoff between latency (index size) and performance can be controlled by varying the threshold δ.

7.4.2 The Effect of Representation Size. In this experiment, we investigate the degree to which the dimension of the query and document representations influences the final ranking performance of the models. The idea is motivated by recent research [66], which suggests that the representation vectors are not the bottleneck of dual-encoder models, but rather the document encoder complexity is. Since the dimensionality of the representations directly influences the index size, it is desirable to keep it as low as possible.

In order to analyze the effect, we train a number of dual-encoder models (cf. 3.2.2), where all hyperparameters except the hidden dimension H and number of attention heads A are kept the same. We show results for embedding-based ($L = 0$) and attention-based ($L = 12$) query encoders in Figure 9. There is a tradeoff between the dimensionality of representations and ranking performance, which is expected; this tradeoff is exhibited by both embedding-based and attention-based query encoders. Overall, the results show that the performance reduction is rather small for $H = 512$ and even $H = 256$ (compared to $H = 768$), considering that it goes hand in hand with a reduction in index size of approximately 33% and 67%, respectively.

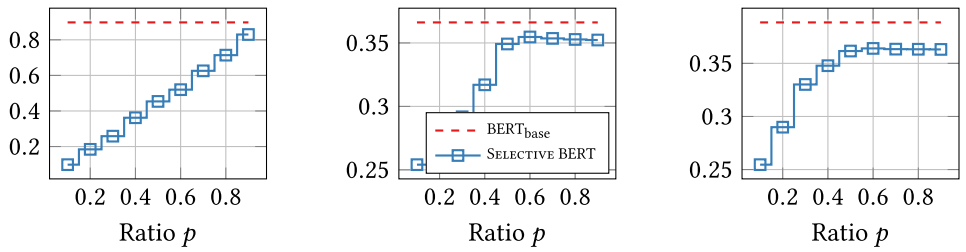
7.5 Can the Indexing Efficiency be Improved by Removing Irrelevant Document Tokens?

In this experiment, we focus on the Selective BERT document encoders proposed in Section 5.2. In order to analyze the index efficiency and ranking performance, we train two dual-encoders (cf. Section 6.4) with Selective BERT document encoders, where $L = 12$ and $H = 768$. The query encoders have $L = 0$ (embedding-based) and $L = 12$ (attention-based), respectively, and $H = 768$. During fine-tuning (cf. Section 5.2.2), we fix the hyperparameter $p = 0.75$, which controls the ratio of tokens to be removed from the documents; afterward, we create a number of indexes, where



(a) Performance on TREC-DL-Psc'19 (b) Performance on TREC-DL-Psc'20

Fig. 9. Fast-Forward ranking results for $k_S = 5,000$ of embedding-based ($L = 0$) and attention-based ($L = 12$) query encoders. The representation dimension H is always the same for both encoders. The document encoders use $L = 12$ layers and $A = \frac{H}{64}$ attention heads in all cases.



(a) Encoding latency [sec] (b) nDCG@10, $L = 0$ (c) nDCG@10, $L = 12$

Fig. 10. Evaluation of Fast-Forward indexes created using Selective BERT models. The document encoders are $\text{BERT}_{\text{base}}$ models with $L = 12$ and $H = 768$. During fine-tuning, we set the parameter $p = 0.75$ (percentage of tokens to keep). We then vary $p \in [0, 1]$ during the indexing stage, resulting in progressively higher indexing efficiency (Figure 10(a)). The corresponding Fast-Forward ranking performance on MSM-Psg-Dev is shown in Figure 10(b) for an embedding-based query encoder ($L = 0$) and in Figure 10(c) for an attention-based query encoder ($L = 12$). Document encoding latency is measured on GPU with a batch size of 256 passages from the MS MARCO corpus (tokenization cost is excluded, as it is identical for all models).

we vary p between 0.1 and 0.9, and compute the corresponding indexing time (using GPUs) and ranking performance. The results are plotted in Figure 10.

The document encoding latency (Figure 10(b)) increases nearly linearly with the ratio of tokens to keep (p). Even though the BERT model has a quadratic complexity w.r.t. input length, this is expected, as there is a certain amount of overhead introduced by the scoring network and the reconstruction of the batches. More interestingly, the ranking performance (Figure 10(b) and (c)) is mostly unchanged for $p \geq 0.5$ in both cases, however, neither models manage to match the performance of their respective baselines (the same configuration with a standard BERT model instead of Selective BERT). We hypothesize that the reason for this could be the choice of $p = 0.75$ during the fine-tuning step.

Overall, our results show that up to 50% of document tokens can be removed without much of a performance reduction. Encoding half of the number of tokens results in approximately halving

Table 6. Retrieval Results of Dual-Encoder Models Using Lightweight Query Encoders and Some Baselines

	TREC-DL-Psg'19			TREC-DL-Doc'19		
	AP_{1k}	R_{1k}	$nDCG_{10}$	AP_{1k}	R_{1k}	$nDCG_{10}$
Sparse Retrieval						
BM25	0.301	0.750	0.506	0.331	0.697	0.519
Dense Retrieval						
ANCE	-	-	0.648	-	-	0.628
TCT-ColBERT	-	-	0.670	-	-	-
Our Models						
$L = 0, H = 768$	0.198	0.486	0.424	0.100	0.263	0.342
$L = 12, H = 768$	0.318	0.691	0.545	0.201	0.457	0.504

For TREC-DL-Doc'19, the dense retrieval depth is set to $k_D = 10000$ and maxP aggregation is applied (cf. Equation (1)). Our model with $L = 0$ uses an embedding-based query-encoder, and the one with $L = 12$ uses an attention-based query encoder. The document encoder is a $BERT_{base}$ model ($L = 12$, $H = 768$) in both cases.

the time required to encode documents. This has a large impact on efficient index maintenance in the context of dynamically increasing document collections. For future work, the Selective BERT architecture can be further refined, for example, by introducing improved (contextualized) scoring networks.

8 DISCUSSION

In this section, we reflect upon our work and present possible limitations.

8.1 Efficient Encoders for Dense Retrieval

Our research questions and experiments have focused exclusively on interpolation-based re-ranking using dual-encoders and Fast-Forward indexes. However, the most common application of dual-encoders in the field of IR is the use as dense retrieval models; a natural question that occurs is, whether the encoders proposed in Section 5 can be used for more efficient dense retrieval.

In Table 6, we present passage and document retrieval results on the MS MARCO corpus. Dense retrievers use a FAISS [32] vector index; no interpolation or re-ranking is performed. It is immediately obvious that our models do not achieve competitive results; on the contrary, the embedding-based encoder yields far worse performance than dense retrievers and even BM25, and even the attention-based encoder fails to improve over sparse retrieval.

From these results, we infer that the models we trained are not suitable for dense retrieval. However, we assume that the main reason for this is not the architecture of the query encoder, but instead the following:

— We use a simple in-batch negative sampling strategy [34], which has been shown to be less effective than more involved strategies [51, 53, 86, 88].
— The hardware we use for training the models is limiting w.r.t. the batch size and thus the number of negative samples, i.e., we cannot use a batch size greater than 4.
— We perform validation and early stopping based on re-ranking.

Considering the points above, we expect that our dual-encoder models, including ones with lightweight encoders, could also be used in retrieval settings if the shortcomings of the training setup are addressed, for example, by using more powerful hardware and state-of-the-art training approaches. On the other hand, we argue that the fact that our models perform well in the re-ranking

Table 7. Zero-shot Ranking Results on BEIR Datasets (nDCG@10) Using Embedding-based ($L = 0$) and Attention-based ($L = 12$) Query Encoders

		Fast-Forward	
	BM25	$L = 0, H = 768$	$L = 12, H = 768$
MS MARCO	0.477	0.653	0.677
Fever	0.649	0.715	0.777
FiQA	0.254	0.282	0.313
Quora	0.808	0.761	0.804
HotpotQA	0.602	0.628	0.674
DBpedia-Entity	0.320	0.331	0.393
SciFact	0.691	0.676	0.698
NFCorpus	0.327	0.327	0.330

The document encoder is a BERT model with 12 layers and 768-dimensional representations. The sparse retrieval depth is $k_S = 5000$.

setting (see Section 7) shows that it is both easier and more efficient (in terms of time and resources) to train models to be used with Fast-Forward indexes instead of for dense retrieval.

8.2 Out-of-Domain Performance

In the previous sections, we found that Fast-Forward indexes and lightweight query encoders show good performance in in-domain ranking tasks. This raises the question of whether the models generalize well to out-of-domain tasks.

In order to ascertain the out-of-domain capabilities of our models, we evaluate them on a number of test sets from the BEIR benchmark. The evaluation happens in a zero-shot fashion, meaning that we use the same models as before and do not re-train them on the respective datasets. The results are shown in Table 7. It is apparent that the attention-based query encoder yields better results than the embedding-based one in all cases, but the difference varies across datasets. Since both models were trained on MS MARCO, they perform well on the BEIR version of that dataset, as expected; notable differences in performance are observed on Fever and DBpedia-Entity, however, both models manage to improve the BM25 results. Finally, on Quora, SciFact, and NFCorpus, re-ranking does not lead to a performance improvement, but rather fails to improve or even degrades the results. We assume that the corresponding tasks either require specific in-domain knowledge of the model or would benefit greatly from query-document attention (cross-attention).

8.3 Threats to Validity

In this section, we outline and discuss certain aspects of the experimental evaluation in this article which result in possible threats to the validity of the results.

8.3.1 Performance of BERT-CLS. In Tables 3 and 4, we report the performance of dual-encoder ranking models, along with a cross-attention model (BERT-CLS). We found that BERT-CLS performed notably worse, especially when the sparse retrieval depth k_S is increased. This result is unexpected, especially considering the fact that the cross-attention architecture allows for query-document attention.

In addition to the architecture itself, the models differ in the way they are trained: ANCE and TCT-ColBERT use complex distillation and negative sampling approaches, along with contrastive loss functions (cf. Equation (3)), while BERT-CLS is trained using simple pairwise loss. It is thus reasonable to assume that the negative sampling approach has a positive impact on the performance. Specifically, the contrastive loss trains the models to identify relevant documents among a very

large number of irrelevant documents, while the pairwise loss focuses on re-ranking mostly related documents, which could explain the performance drop for higher retrieval depths.

Furthermore, it is important to note that, even if BERT-CLS performed similarly to the dual-encoder models, the difference in efficiency would remain the same, leaving the claims we make unaffected.

8.3.2 Latency Measurements. As Fast-Forward indexes aim at improving ranking efficiency, we mainly focus on the query processing latency, which is reported in Tables 3–5 and Figure 7. As the experiments in the article have been performed over a longer period of time, there have been slight changes with respect to, for example, hardware or implementations. Consequently, the numbers in latency might not be directly comparable **across experiments**. Thus, we made sure to make each experiment self-contained, such that these comparisons are not necessary; rather, our results highlight relative latency improvements **within** each experiment, where all measurements are comparable. In general, one should also keep in mind that latency can be heavily influenced by the way a method is implemented.

8.3.3 Hybrid Retrieval Baselines. In Tables 3 and 4, we presented, along with the results of our own method, some hybrid retrieval baselines. Table 1 shows the corresponding indexes that we used for the dense retrievers. It is important to note that those are *brute-force* indexes, i.e., they perform exact kNN retrieval. It is thus to be expected that the latency of hybrid retrieval can be further reduced by employing approximate dense retrieval instead; this would likely go hand in hand with a reduction in performance though.

9 CONCLUSION

In this article, we proposed Fast-Forward indexes, a simple yet effective and efficient look-up-based interpolation method that combines lexical and semantic ranking. Fast-Forward indexes are based on dense dual-encoder models, exploiting the fact that document representations can be pre-processed and stored, providing efficient access in constant time. Using interpolation, we observed increased performance compared to hybrid retrieval. Furthermore, we achieved improvements of up to 75% in memory footprint and query processing latency due to our optimization techniques, *sequential coalescing* and *early stopping*.

Moreover, we introduced efficient encoders for dual-encoder models: Embedding-based and lightweight attention-based query encoders can be used to compute query representations significantly faster without compromising performance too much. Selective BERT document encoders dynamically remove irrelevant tokens from input documents prior to indexing, reducing the document encoding latency by up to 50% and thus making index maintenance much faster.

Our method solely requires CPU computations for ranking, completely eliminating the need for expensive GPU-accelerated re-ranking.

REFERENCES

[1] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3490–3496. DOI: https://doi.org/10.18653/v1/D19-1352

[2] Arian Askari, Amin Abolghasemi, Gabriella Pasi, Wessel Kraaij, and Suzan Verberne. 2023. Injecting the BM25 Score as Text Improves BERT-Based Re-rankers. In *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 66–83.

[3] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*. Association for Computing Machinery, 426–434. DOI: https://doi.org/10.1145/956863.956944

[4] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems* 42, 1, Article 20 (2023), 35 pages. DOI: https://doi.org/10.1145/3596512

[5] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=rkg-mA4FDr

[6] Xiaoyang Chen, Kai Hui, Ben He, Xianpei Han, Le Sun, and Zheng Ye. 2021. Co-BERT: A context-aware bert retrieval model incorporating local and query-specific context. arXiv:2104.08523. Retrieved from https://arxiv.org/abs/2104.08523

[7] Eunseong Choi, Sunkyung Lee, Minjin Choi, Hyeseon Ko, Young-In Song, and Jongwuk Lee. 2022. SpaDE: Improving sparse representations using a dual document encoder for first-stage retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM'22)*. Association for Computing Machinery, 272–282. DOI: https://doi.org/10.1145/3511808.3557456

[8] Nachshon Cohen, Amit Portnoy, Besnik Fetahu, and Amir Ingber. 2022. SDR: Efficient neural re-ranking using succinct document representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 6624–6637. DOI: https://doi.org/10.18653/v1/2022.acl-long.457

[9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Ian Soboroff. 2021. TREC deep learning track: Reusable test collections in the large data regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 2369–2375. DOI: https://doi.org/10.1145/3404835.3463249

[10] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. Association for Computing Machinery, 985–988. DOI: https://doi.org/10.1145/3331184.3331303

[11] Zhuyun Dai and Jamie Callan. 2019. An evaluation of weakly-supervised DeepCT in the TREC 2019 deep learning track. In *Proceedings of the TREC*.

[12] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for Ad-Hoc search. In *Proceedings of the Web Conference 2020 (WWW'20)*. Association for Computing Machinery, 1897–1907. DOI: https://doi.org/10.1145/3366423.3380258

[13] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. Association for Computing Machinery, 1533–1536. DOI: https://doi.org/10.1145/3397271.3401204

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. DOI: https://doi.org/10.18653/v1/N19-1423

[15] Sibo Dong, Justin Goldstein, and Grace Hui Yang. 2022. SEINE: SEgment-based indexing for neural information retrieval. In *Proceedings of the Workshop on Reaching Efficiency in Neural Information Retrieval, the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[16] Ronald Fagin, Amnon Lotem, and Moni Naor. 2001. Optimal aggregation algorithms for middleware. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'01)*. Association for Computing Machinery, 102–113. DOI: https://doi.org/10.1145/375551.375567

[17] Zhen Fan, Luyu Gao, Rohan Jha, and Jamie Callan. 2023. COILcr: Efficient semantic matching in contextualized exact match retrieval. In *Proceedings of the Advances in Information Retrieval*. Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.), Springer Nature Switzerland, Cham, 298–312.

[18] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 2288–2292. DOI: https://doi.org/10.1145/3404835.3463098

[19] Luke Gallagher. 2019. Pairwise t-test on TREC Run Files. Retrieved from https://github.com/lgrz/pairwise-ttest/. Accessed April 2021.

[20] Luyu Gao and Jamie Callan. 2021. Condenser: A pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 981–993. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.75

[21] Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 2843–2853. DOI: https://doi.org/10.18653/v1/2022.acl-long.203

[22] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. 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, 3030–3042. DOI: https://doi.org/10.18653/v1/2021.naacl-main.241

[23] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Proceedings of the Advances in Information Retrieval*. Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani (Eds.), Springer International Publishing, Cham, 146–160.

[24] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, 10 pages.

[25] Vishal Gupta, Manoj Chinnakotla, and Manish Shrivastava. 2018. Retrieve and re-rank: A simple and effective IR approach to simple question answering over knowledge graphs. In *Proceedings of the 1st Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, Brussels, Belgium, 22–27. DOI: https://doi.org/10.18653/v1/W18-5504

[26] Sebastian Hofstätter, Omar Khattab, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Introducing neural bag of whole-words with ColBERTer: Contextualized late interactions using enhanced reduction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM'22)*. Association for Computing Machinery, 737–747. DOI: https://doi.org/10.1145/3511808.3557367

[27] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery. DOI: https://doi.org/10.1145/3404835.3462891

[28] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: Learning to select passages for neural document ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 1349–1358. DOI: https://doi.org/10.1145/3404835.3462889

[29] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable and time-budget-constrained contextualization for re-ranking. In *Proceedings of the ECAI*.

[30] Sebastian Hofstätter, Nick Craswell, Bhaskar Mitra, Hamed Zamani, and Allan Hanbury. 2022. Are We There Yet? A Decision Framework for Replacing Term Based Retrieval with Dense Retrieval Systems. arXiv:2206.12993. Retrieved from https://arxiv.org/abs/2206.12993

[31] Kyoung-Rok Jang, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Heecheol Seo. 2021. Ultra-high dimensional sparse representations with binarization for efficient text retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 1016–1029. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.78

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

[33] Euna Jung, Jaekeol Choi, and Wonjong Rhee. 2022. Semi-siamese bi-encoder neural ranking model using lightweight fine-tuning. In *Proceedings of the ACM Web Conference 2022 (WWW'22)*. Association for Computing Machinery, 502–511. DOI: https://doi.org/10.1145/3485447.3511978

[34] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wentau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.550

[35] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 39–48. DOI: https://doi.org/10.1145/3397271.3401075

[36] Seungyeon Kim, Ankit Singh Rawat, Manzil Zaheer, Sadeep Jayasumana, Veeranjaneyulu Sadhanala, Wittawat Jitkrittum, Aditya Krishna Menon, Rob Fergus, and Sanjiv Kumar. 2023. EmbedDistill: A Geometric Knowledge Distillation for Information Retrieval. arXiv:2301.12005. Retrieved from https://arxiv.org/abs/2301.12005

[37] John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*. Association for Computing Machinery, 111–119. DOI: https://doi.org/10.1145/383952.383970

[38] Carlos Lassance and Stéphane Clinchant. 2022. An efficiency study for SPLADE models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. Association for Computing Machinery, 2220–2226. DOI: https://doi.org/10.1145/3477495.3531833

[39] Carlos Lassance, Hervé Déjean, and Stéphane Clinchant. 2023. An Experimental Study on Pretraining Transformers from Scratch for IR. *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Davis, Udo Kruschwitz, Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 504–520.

[40] Carlos Lassance, Maroua Maachou, Joohee Park, and Stéphane Clinchant. 2022. Learned token pruning in contextualized late interaction over BERT (ColBERT). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. Association for Computing Machinery, 2232–2236. DOI: https://doi.org/10.1145/3477495.3531835

[41] Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*. ACM, 120–127. DOI: https://doi.org/10.1145/383952.383972

[42] Jurek Leonhardt, Avishek Anand, and Megha Khosla. 2020. Boilerplate removal using a neural sequence labeling model. In *Companion Proceedings of the Web Conference 2020 (WWW'20)*. Association for Computing Machinery, 226–229. DOI: https://doi.org/10.1145/3366424.3383547

[43] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. 2023. Extractive explanations for interpretable text ranking. *ACM Transactions on Information Systems* 41, 4, Article 88 (2023), 31 pages. DOI: https://doi.org/10.1145/3576924

[44] Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. 2022. Efficient neural ranking using forward indexes. In *Proceedings of the ACM Web Conference 2022 (WWW'22)*. Association for Computing Machinery, 266–276. DOI: https://doi.org/10.1145/3485447.3511955

[45] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2023. PARADE: Passage representation aggregation for document reranking. *ACM Trans. Inf. Syst.* 42, 2 (2023), 26. https://doi.org/10.1145/3600088

[46] Minghan Li and Eric Gaussier. 2021. KeyBLD: Selecting key blocks with local pre-ranking for long document information retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 2207–2211. DOI: https://doi.org/10.1145/3404835.3463083

[47] Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2022. CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval. (2022). arXiv:2211.10411. Retrieved from https://arxiv.org/abs/2211.10411

[48] Minghan Li, Diana Nicoleta Popa, Johan Chagnon, Yagmur Gizem Cinar, and Eric Gaussier. 2023. The power of selecting key blocks with local pre-ranking for long document information retrieval. *ACM Transactions on Information Systems* 41, 3, Article 73 (2023), 35 pages. DOI: https://doi.org/10.1145/3568394

[49] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. *P*yserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2356–2362. DOI: https://doi.org/10.1145/3404835.3463238

[50] Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. 2023. Aggretriever: A simple approach to aggregate textual representations for robust dense passage retrieval. *Transactions of the Association for Computational Linguistics* 11 (2023), 436–452. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00556/116046/Aggretriever-A-Simple-Approach-to-Aggregate

[51] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Association for Computational Linguistics, Online, 163–173. DOI: https://doi.org/10.18653/v1/2021.repl4nlp-1.17

[52] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Association for Computational Linguistics, Online, 163–173. DOI: https://doi.org/10.18653/v1/2021.repl4nlp-1.17

[53] Erik Lindgren, Sashank Reddi, Ruiqi Guo, and Sanjiv Kumar. 2021. Efficient training of retrieval models using negative cache. In *Proceedings of the Advances in Neural Information Processing Systems*. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34, Curran Associates, Inc., 4134–4146. Retrieved from https://proceedings.neurips.cc/paper/2021/file/2175f8c5cd9604f6b1e576b252d4c86e-Paper.pdf

[54] Chang Liu, Chongyang Tao, Xiubo Geng, Tao Shen, Dongyan Zhao, Can Xu, Binxing Jiao, and Daxin Jiang. 2022. Adam: Dense Retrieval Distillation with Adaptive Dark Examples. arXiv:2212.10192. Retrieved from https://arxiv.org/abs/2212.10192

[55] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00369/100684/Sparse-Dense-and-Attentional-Representations-for

[56] Julia Luxenburger, Shady Elbassuoni, and Gerhard Weikum. 2008. Matching task profiles and user needs in personalized web search. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. Association for Computing Machinery, 689–698. DOI: https://doi.org/10.1145/1458082.1458175

[57] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via prediction of importance with contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 1573–1576. DOI: https://doi.org/10.1145/3397271.3401262

[58] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. Association for Computing Machinery, 1101–1104. DOI: https://doi.org/10.1145/3331184.3331317

[59] Joel Mackenzie, Zhuyun Dai, Luke Gallagher, and Jamie Callan. 2020. Efficiency implications of term weighting for passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. Association for Computing Machinery, 1821–1824. DOI: https://doi.org/10.1145/3397271.3401263

[60] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. DOI: https://doi.org/10.1109/TPAMI.2018.2889473

[61] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 1723–1727. DOI: https://doi.org/10.1145/3404835.3463030

[62] P. Massart. 1990. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability* 18, 3 (1990), 1269–1283. http://www.jstor.org/stable/2244426

[63] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A Dual Embedding Space Model for Document Ranking. arXiv:1602.01137. Retrieved from https://arxiv.org/abs/1602.01137

[64] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. 2019. Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks. arXiv:1907.03693. Retrieved from https://arxiv.org/abs/1907.03693

[65] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*. Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne (Eds.), Vol. 1773, CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

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

[67] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online Preprint* (2019). https://scholar.google.com/scholar?cluster=842206559672245113O&hl=en&as_sdt=0,5

[68] Prafull Prakash, Julian Killingback, and Hamed Zamani. 2021. Learning robust dense retrieval models from incomplete relevance labels. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 1728–1732. DOI: https://doi.org/10.1145/3404835.3463106

[69] David Rau and Jaap Kamps. 2022. The role of complex NLP in transformers for text ranking. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR'22)*. Association for Computing Machinery, 153–160. DOI: https://doi.org/10.1145/3539813.3545144

[70] Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389. DOI: https://doi.org/10.1561/1500000019

[71] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[72] Koustav Rudra and Avishek Anand. 2020. Distant supervision in BERT-based adhoc document retrieval. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. Association for Computing Machinery, 2197–2200. DOI: https://doi.org/10.1145/3340531.3412124

[73] Koustav Rudra, Zeon Trevor Fernando, and Avishek Anand. 2021. An In-depth Analysis of Passage-Level Label Transfer for Contextual Document Ranking. arXiv:2103.16669. Retrieved from https://arxiv.org/abs/2103.16669

[74] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, reuse, recycle: Green information retrieval research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'22)*. Association for Computing Machinery, 2825–2837. DOI: https://doi.org/10.1145/3477495.3531766

[75] R. F. Simmons. 1965. Answering english questions by computer: A survey. *Communications of the ACM* 8, 1 (1965), 53–70. DOI: https://doi.org/10.1145/363707.363732

[76] 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 *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. Retrieved from https://openreview.net/forum?id=wCu6T5xFjeJ

[77] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. 2004. Top-k query evaluation with probabilistic guarantees. In *Proceedings of the 30th International Conference on Very Large Data Bases—Volume 30 (VLDB'04)*. VLDB Endowment, 648–659.

[78] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. arXiv:1908.08962. Retrieved from https://arxiv.org/abs/1908.08962

[79] Howard Turtle and James Flood. 1995. Query evaluation: Strategies and optimizations. *Information Processing and Management* 31, 6 (1995), 831–850. DOI: https://doi.org/10.1016/0306-4573(95)00020-H

[80] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 6000–6010.

[81] Jue Wang, Ke Chen, Gang Chen, Lidan Shou, and Julian McAuley. 2022. SkipBERT: Efficient inference with shallow layer skipping. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 7287–7301. DOI: https://doi.org/10.18653/v1/2022.acl-long.503

[82] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533. Retrieved from https://arxiv.org/abs/2212.03533

[83] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. BERT-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR'21)*. Association for Computing Machinery, 317–324. DOI: https://doi.org/10.1145/3471158.3472233

[84] Xing Wei and W. Bruce Croft. 2006. LDA-based document models for Ad-Hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*. Association for Computing Machinery, 178–185. DOI: https://doi.org/10.1145/1148170.1148204

[85] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2246–2251. DOI: https://doi.org/10.18653/v1/2020.acl-main.204

[86] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=zeFrfgyZln

[87] Yingrui Yang, Yifan Qiao, and Tao Yang. 2022. Compact token representations with contextual quantization for efficient document re-ranking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 695–707. DOI: https://doi.org/10.18653/v1/2022.acl-long.51

[88] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. *Optimizing dense retrieval model training with hard negatives*. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 1503–1512. DOI: https://doi.org/10.1145/3404835.3462880

[89] Kai Zhang, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. 2023. LED: Lexicon-enlightened dense retriever for large-scale retrieval. In *Proceedings of the ACM Web Conference 2023*, Association for Computing Machinery, New York, NY, 3203–3213. https://doi.org/10.1145/3543507.3583294

[90] Yucheng Zhou, Tao Shen, Xiubo Geng, Chongyang Tao, Guodong Long, Can Xu, and Daxin Jiang. 2022. Fine-Grained Distillation for Long Document Retrieval. arXiv:2212.10423. Retrieved from https://arxiv.org/abs/2212.10423

[91] Shengyao Zhuang and Guido Zuccon. 2021. Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. arXiv:2108.08513. Retrieved from https://arxiv.org/abs/2108.08513

[92] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term independent likelihood MoDEl for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. Association for Computing Machinery, 1483–1492. DOI: https://doi.org/10.1145/3404835.3462922

Received 1 March 2023; revised 7 September 2023; accepted 24 October 2023