The Impact of the Retrieval Stage in Interpolation-based Re-Ranking

Dan-Cristian Ciacu[1]

Supervisor(s): Avishek Anand[1]**, Jurek Leonhardt**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Dan-Cristian Ciacu
Final project course: CSE3000 Research Project
Thesis committee: Avishek Anand, Jurek Leonhardt, Alan Hanjalic

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Efficient and effective information retrieval (IR) systems are needed to fetch a large number of relevant documents and present them based on their relevance to the input queries. Previous work reported the use of sparse and dense retrievers. Sparse retrievers offer low latency but suffer from term mismatch issues, while dense retrievers improve performance at the cost of higher processing times. The literature proposed Fast-Forward Indexes, an interpolation-based re-ranking framework that leverages the benefits of both sparse and dense retrievers.

Although a lot of work was done in the field, most studies evaluate the performance of the proposed models only on the MS Marco dataset, neglecting other datasets. This study extends previous work by exploring how different sparse retrievers, employing no-encoder, uni-encoder, and bi-encoder architectures, perform in an interpolation-based re-ranking setting on datasets originating from various domains. Results show that bi-encoder-based retrievers outperform the other sparse retrievers in terms of recall but with a substantial increase in latency compared to simpler retrievers, which generally showed good performance. Further, when the retrievers were used in an interpolation-based re-ranking setting, they performed similarly in terms of ranking quality.

1 Introduction

The primary goal of an Information Retrieval (IR) system is to retrieve documents (e.g. websites, threads) relevant to the user's request. Given that multiple documents could be found suitable, these are usually ranked based on their similarity to the input query. The ad-hoc setting (i.e. documents have no specific structure) desires (i) a fast retrieval and (ii) a large fraction of the retrieved documents are relevant to the query. In addition, some studies [1, 2, 3] suggest a two-stage retrieval approach (known as Retrieve and Re-rank), where the retrieved documents get re-ranked using a more expensive method.

Retrieval models that stored documents in an inverted index structure became popular [4] due to their remarkable performance and ability to prune a large number of irrelevant documents [2]. This method can be associated with sparse retrieval, due to the sparse nature of the inverted indexes. One of the first models was TF-IDF, where the authors introduced the concept of document relevancy based on term use, instead of term meaning [5]. Later, BM25 was introduced, based on the probabilistic retrieval framework [6]. Currently, BM25 is considered a baseline in the retrieval task, due to its efficiency and simplicity [7]. Additionally, there was a considerable effort to adapt BM25 to different use cases, such as structured search [8] or semantic search [9]. However, the traditional retrieval/weighting models suffer from the vocabulary mismatch problem, in which the relevant results might not contain the terms found in the query, thereby remaining unselected.

To alleviate that, several approaches were suggested. Pseudo Relevance Feedback (PRF) was introduced to mitigate it and it expands the user's query using terms from the top-k ranked documents retrieved in a first pass. Yet, studies show that the expansion terms are often unrelated to the query, reducing the retrieval quality [10]. As an alternative, neural approaches were suggested [11, 12]. The key aspect of these approaches consists of encoding documents and queries into vector representations. In the literature, these models are usually referred to as dense models. Studies showed that dense models perform significantly better than sparse models but with increased latency. The most common approaches employ bi-encoder-based models, where queries and documents are encoded independently, allowing for precomputing the document embeddings. Nonetheless, searching in a dense space using a standard approach, such as kNN, is still non-feasible for ad-hoc retrieval due to the increased latency. Fortunately, recent studies propose ANN (approximate nearest neighbor) search [13, 14] which promises similar performance with reduced latency.

Considering the benefits of both sparse and dense models, recent studies [15, 16, 17, 18] propose neural term-weighting approaches that store weights in inverted indexes (known as learnt sparse retrievers). These methods show impressive results for the retrieval task. Yet, the latency is reasonably similar to traditional sparse retrievers (e.g. BM25) for uni-encoder term-weighting models, while in the bi-encoder-based models, the latency is poorer [15].

Following both different retrieval approaches and the goal of ad-hoc retrieval, Fast-Forward Indexes [2] was introduced. Fast-Forward Indexes is a Retrieve and Re-Rank framework that employs an interpolation-based re-ranking approach. The Retrieve and Re-Rank framework usually employs a simple retrieval model (e.g. sparse retriever) to fetch documents fast (known as the first-stage retrieval) and the retrieved documents are fed to a more expensive model (e.g. dense model). Interpolation-based Re-Ranking consists of re-ranking documents based on an interpolation of the scores from the two stages. Furthermore, Fast-Forward Indexes tries to (i) improve the ranking of long documents, without decreasing performance and (ii) reduce query processing times. The authors achieved that by introducing two novel techniques for index compression and early stopping during searching.

It is important to note that most of the extensive efforts in developing retrieval models overlooked diversity. Most proposed retrieval systems were only evaluated on the MS MARCO corpus [19], using different sets of queries (e.g. TREC-DL-2019 [20] or TREC-DL-2020 [21]). Further, in the context of Fast-Forward Indexes, Leonhardt et al. evaluated the efficiency of their proposed retrieve and re-rank approach on MS MARCO and they only considered BM25 and SPLADE [17] (a neural sparse retriever) as retrieval models. Nonetheless, exploring other alternatives could lead to a more meaningful understanding of the importance of the retrieval stage in Fast-Forward Indexes.

This paper will answer the following question:

RQ *What is the impact of the retrieval stage in the context*

of Fast-Forward Indexes?
To answer the main question (RQ) in a more structured way, the following research sub-questions are proposed:

SQ 1 *What is the impact of the retrieval stage on the performance of Fast-Forward Indexes?*

SQ 2 *How does the selected retrieval model affect the latency of Fast-Forward Indexes?*

The contribution. This paper will explore both traditional sparse retrievers (TF-IDF, BM25) and neural sparse retrievers, such as DeepCT or SPLADE on various datasets for tasks including question-answering, and entity retrieval. Finally, this work shows that SPLADE, a bi-encoder sparse retriever, outperforms other sparse retrievers in terms of recall. Additionally, traditional sparse retrievers generally provide satisfactory results with significantly lower processing latency.

The rest of the paper will be structured as follows. Section 2 provides the necessary background information on the topics discussed in the next sections. Then, section 3 outlines the methodology. Results and their analysis are discussed in sections 4, 5, and 6. Lastly, section 7 presents the summary and future work.

2 Background

This section will discuss the commonly used concepts and techniques for a better understanding of the topic.

2.1 Retrieve and Re-rank

Retrieval models are responsible for fetching documents related to a given query. Simultaneously, they apply a scoring algorithm to each retrieved document to enable a ranking of the documents. Generally, retrieval models follow one of the two architectures: sparse models or dense models.

The re-ranking phase was introduced as a supplementary step to the retrieval phase and it aims to improve the rankings of relevant documents without the need to consider the entire search space [1]. Creating a system leveraging both retrieval and re-ranking phases proved beneficial for the adhoc retrieval task, as it showed notable improvements over other complex neural approaches [1, 2]. Following this idea, interpolation-based re-ranking was introduced, where the scores from both retrieval and re-ranking phases are considered during re-ranking [2].

2.2 Sparse Retrieval

Sparse retrievers became popular in the IR world because they use an inverted index structure to store term-specific statistics, enabling faster retrievals and lower memory usage. The sparse retrieval approaches can be divided into two main categories. First, the traditional term-weighting models, where the weights of each term are computed based on predefined mathematical formulas and using basic document statistics. Second, exploiting neural networks to (i) perform term-weighting and (ii) expand documents and queries, aiming to alleviate the vocabulary mismatch problem.

Traditional term-weighting

The first retrieval systems were based on simple statistics from a given query and a set of documents. The most used statistics were the term-frequency and the inverse document-frequency. Term-frequency (also known as "Tf") represents the relative number of occurrences of a term in a document, as illustrated in Equation 1. On the other hand, the inverse document-frequency ("Idf") is the logarithmically scaled inverse fraction of the number of documents that contain a specific term (Equation 2).

Table 1: Example of Document Expansion Using DocT5Query

Document:	Inborn errors of bile acid synthesis can produce life-threatening cholestatic liver disease (which usually presents in infancy) and progressive neurological disease presenting later in childhood or in adult life.[...]
Generated query:	what type of disease do inborn errors of bile acid synthesis cause
Target query:	is autoimmune hepatitis a bile acid synthesis disorder

$$\text{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

$$\text{idf}(t,D) = \log \frac{N}{n_t} \tag{2}$$

The terms used in the formulas are: $f_{t,d}$ represents the frequency of term t in document d, $\sum_{t' \in d} f_{t',d}$ is the total number of terms in document d, N is the total number of documents available in the corpus, n_t is the number of documents containing at least one occurrence of term t.

Neural term-weighting

For neural term-weighting, also known as learnt term-weighting, two fundamental architectures exist. The first includes separate encoders (i.e. neural networks) for documents and queries, known as *bi-encoders*. A notable advantage of the *bi-encoder* architecture is that it allows for pre-computing the document representations. Therefore, at search time, only the query needs to be encoded, allowing for faster retrieval. For example, SparTerm [22] uses a pre-trained language model (PLM) to convert frequency-based details to a term importance distribution across the entire vocabulary space.

Uni-encoders use only one encoder and it is used for either documents or queries. DeepCT [16] is a well-known approach that uses a deep-learning model to learn contextualized term weights from a passage during document indexing.

Given their enhanced performance compared to the traditional term-weighting approaches, the neural-based models still face the term mismatch problem [15].

Document & Query Expansion

Document expansion is a technique to augment the original document with additional terms. The main objective of the method is to trigger more matching documents during search, in an attempt to alleviate the term mismatch problem. Furthermore, query expansion works similarly.

A popular document expansion approach is "DocT5Query" [23]. This approach generates a series

of potential questions that a document could answer, which are appended to the original document. The questions are generated using the T5 sequence-to-sequence transformer [24], which processes the document's terms and produces the corresponding questions. The main advantage of the document expansion techniques is that they do not add additional overhead to the query processing times, since the expansion is performed offline, during indexing. Table 1 shows an example of a query produced by the model.

Regarding the query expansion procedures, "TILDEv2" [25] is a state-of-the-art model that promises comparable performance to the more complex models, such as BERT [26] or ANCE [27], but with reduced latency. The main idea behind this approach is to use a neural model to learn contextualized information about the passages and expand the queries.

Classical Ranking Functions

Retrieving relevant documents is important, but presenting the most relevant ones first is crucial. To achieve that, ranking (or scoring) functions are used to generate scores that determine relevancy. TF-IDF [5] is one of the early scoring models to be used in the sparse retrieval task. It is computed as the product of "Tf" and "Idf". Similarly, BM25 is another scoring model that still leverages "Tf" and "Idf". It incorporates document length normalization and additional parameters to improve the scoring. Another approach is the Query Likelihood model [28], which represents each document as a language model, allowing for ranking based on the likelihood of the document generating the query.

2.3 Dense Retrieval

Dense retrieval is an alternative to sparse retrieval where neural networks are employed to translate documents and queries to *feature vectors* (also known as *dense vectors*).

The main architecture for dense retrievers is based on *dual-encoders*, often referred to as *bi-encoders*. This structure proposes an independent encoding for queries and documents, allowing for pre-computing document representations during indexing. Yet, the underlying encoders could either be identical (known as *Siamese encoder*) or different.

Recent studies [29, 30, 31] in the direction of dense models typically shared a common characteristic. They all build on top of BERT [26] to further improve the performance. BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a Language Model (LM) that leverages transformers [32], which is a deep-learning model that uses self-attention to capture relationships between all terms and assigns attention weights to the terms based on the relevance. BERT stands out by being designed to read in both directions at the same time, unlike other models where reading is uni-directional [33].

Given that feature vectors cannot be stored in an inverted index structure, dense retrievers come with the problem of retrieving relevant documents due to the inherent complexity of searching through a set of embeddings. One first solution is to apply kNN (k-nearest neighbors) search. Even though its results are accurate, its time complexity grows proportionally with both the number of documents and the number of dimensions, making it unfeasible for the ad-hoc retrieval task. An alternative would be to use ANN (approximate nearest neighbors) search to retrieve the candidates. This method is affected by the inability to guarantee the optimal solution, yet it provides comparable results in a reasonable amount of time.

2.4 Fast-Forward Indexes

Fast-forward Indexes emerged as a Retrieve and Re-rank approach with various improvements over the standard procedure. The goal of the authors was two-fold: (1) to rank long documents with increased accuracy and (2) to improve query processing times [2].

In Fast-Forward Indexes, the retrieval is performed by a sparse model, and the initial sparse score for a $(query, document)$ pair is denoted by $\phi_S(q, d)$. The re-ranking phase involves a dense model to achieve a more meaningful relation between a query and the retrieved documents, and the dense score is expressed as $\phi_D(q, d)$. For computing the final rankings, Leonhardt et al. [2] used an interpolation-based re-ranking, as it was proved that including the sparse scores (ϕ_S) in the re-ranking stage could be beneficial [34]. The interpolation is regulated by the hyperparameter α closed in the unit interval $[0, 1]$ (as shown in Equation 3).

$$\phi(q, d) = \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \phi_D(q, d) \tag{3}$$

To reduce the query processing times, Leonhardt et al. [2] propose two innovative concepts for computing the dense scores. First, neural models usually have an upper limit on the input's length, requiring long documents to be split. The split causes the creation of multiple feature vectors, increasing the index size and search time considerably. To alleviate that, Sequential Coalescing for index compression was introduced. It consists of computing a similarity score between two consecutive passages, and if the similarity score is above a threshold, the passages are grouped.

Second, an early stopping technique was introduced. It is known that retrieving more documents than the final number of documents (K) improves performance. Yet, this implies an increased number of look-ups. To overcome that, the Leonhardt et al. [2] propose stopping the search when the estimated current interpolated score (using the maximum dense score found so far) is worse than the minimum interpolated score from the top-K documents.

3 Methodology

This section presents the methodology used to answer the research questions. This exploratory work compares the performance of different sparse retrievers when used in an interpolation-based re-ranking environment. The sparse retrievers employ both traditional term-weighting (explained in subsection 3.1) and neural term-weighting (discussed in subsection 3.2). In subsection 3.3, the datasets used for comparison are introduced, and the metrics used to measure the performance of each retriever are defined. Lastly, hyperparameter optimization and the implementation of the selected models are discussed in subsection 3.5 and in subsection 3.6, respectively.

The dense retrievers were excluded from the evaluations for two reasons. Firstly, Fast-Forward Indexes were introduced as a Retrieve and Re-rank approach, where the retrieval

Table 2: Example of Query Expansion using SPLADE and WordPiece Tokenization. The original query is expanded (new terms are **bold**) into multiple terms with associated relevance scores. The '##' prefix indicates subword tokens generated by the WordPiece tokenizer

Original query:	0 dimensional biomaterials show inductive properties
New query:	Dimensional (187.1), Properties (167.8), ##uc (165.1), **Bio** (158.1), Ind (155.9), ##mate (155.2), 0 (152.7), ##rial (140.3), Show (134.8), Zero (133.0), ##tive (120.1), Dimensions (104.7), Property (100.8), **Characteristics** (67.2)

Table 3: Representative query for each dataset

FiQA	Can I pay off my credit card balance to free up available credit?
NFCorpus	Using Diet to Treat Asthma and Eczema
SciFact	TMEM27 is a marker for beta cells
Quora	Why creativity is important?
HotpotQA	Who was born first, Bryan Callen or Brendan Schaub?
FEVER	Roman Atwood is a content creator
DBPedia	Who founded Intel?
MS MARCO	what is a virtual interface

is performed using a term-frequency-based retriever [2]. Secondly, the authors of Fast-Forward Indexes evaluated the performance of two dense retrievers (TCT-ColBERT and ANCE) against the interpolation-based retrieve and re-rank approach, and the dense retrievers performed substantially worse [2].

3.1 Traditional term-weighting

TF-IDF & BM25

TF-IDF and BM25 are both among the fundamental traditional sparse retrievers. BM25's scoring function is given by the following formula:

$$
\begin{aligned}
BM25(q,d) &= \sum_{q_i:\mathrm{tf}(q_i,d)>0} h_q^{\mathrm{BM25}}(q,t)\cdot h_d^{\mathrm{BM25}}(d,t) \\
h_q^{\mathrm{BM25}}(q,t) &= \frac{\mathrm{idf}(t)\cdot \mathrm{tf}_{t,q}\cdot(k_1+1)}{\mathrm{tf}_{t,q}+k_1} \\
h_d^{\mathrm{BM25}}(d,t) &= \frac{\mathrm{tf}_{t,d}\cdot(k_3+1)}{\mathrm{tf}_{t,d}+k_3\cdot(1-b+b\cdot\frac{|d|}{avgdl})}
\end{aligned}
\tag{4}
$$

where: $IDF(q_i)$ is the inverse document frequency of the query term q_i, $f(q_i, d)$ is the term frequency of q_i in the document d, $|d|$ is the length of the document d, $avgdl$ is the average document length in the collection, and k_1, k_3, b are free parameters.

3.2 Neural term-weighting

DeepCT. Translated to Deep Contextualized Term Weighting, DeepCT is one of the first retrieval models that tackle the problem of integrating context in the term weights, while still storing them in an inverted index approach [16]. The authors propose using BERT to extract contextual features of each term [16]. At the end, the BERT transformer outputs an embedding for each term which reflects the learned contextualized information. To make the resulting embeddings compatible with the standard ranking functions, DeepCT converts each contextualized feature vector to a *term importance score* (i.e. a scalar weight), which then serves as the weight for each term [16]. An interesting aspect of DeepCT is its *query-independent* attribute (also referred to as a *uni-encoder*), meaning that the content of the documents can be computed during indexing, while the queries are not encoded.

DeepImpact. Building on the idea of computing contextualized term weights in an uni-encoder setting, similar to DeepCT, DeepImpact [18] leverages document expansion to further improve retrieval performance. The authors suggested a two-stage approach, also known as *Inject and Re-write*. The *Inject* phase makes use of DocT5Query [23] to generate queries that the document could answer. The rationale behind this idea was to expand the document with terms that were not originally included. The latter stage, *Re-write*, is responsible for recomputing the weights of both original and injected terms. To achieve that, it feeds all terms of the document to BERT which generates an embedding for each input token [18]. Then, the embeddings corresponding to the first occurrence of each term are processed by the *Impact Scores Encoder*, a two-layer MLP (multi-layer perceptron) with ReLU activations [18], which determines the final weights assigned to each term. DeepImpact stands out by incorporating a document expansion technique in a shot to alleviate the vocabulary mismatch. The evaluations show a substantial improvement on MS Marco over DeepCT, and additionally, DeepImpact outperforms methods relying solely on document expansion, proving that term weighting could be beneficial [18].

SPLADE. A neural sparse retriever that learns term relevance for both documents and queries [17]. One particularity of SPLADE is that it predicts the relevance of the terms in BERT WordPiece vocabulary. More explicitly, the BERT WordPiece vocabulary is a set of 30522 sub-words, known as "wordpieces" [35] and its objective is to restrict the number of unique tokens in a Language Model environment. Further, SPLADE uses the WordPiece tokenizer to convert the original input to "wordpieces", which then allows for computing a relevance score for each entry in the BERT vocabulary. This can be seen as a document or query expansion technique since the relevance scores are calculated for the terms in the BERT vocabulary, indicating that the input is augmented by the relevant terms from the vocabulary, along with their weights. An example of query augmentation using SPLADE is presented in Table 2. Additionally, the authors propose a sparsity regulator, responsible for reducing the number of floating-point operations required to compute the score of a document, reducing the search time [17].

uniCOIL. Sharing the dual-encoder architecture with SPLADE, **uni**fied **Co**ntextualized **I**nverted **L**ists (uniCOIL) was introduced to push the performance boundaries of neural sparse retrievers. It is important to note that uniCOIL is based on COIL [36], as uniCOIL only brings a small change to the model. Similar to the previous approaches, COIL leverages BERT to create contextual feature vectors of 768 dimensions for each token of the input. Notably, the authors proposed using a mapping matrix to reduce the vector dimensionality to only 32 dimensions, as they discovered that the reduction does not affect performance [36].

The newer model, suggests reducing the dimensionality even further, bringing the token dimension to one, and applying a ReLU function to the token values to ensure they remain positive for all tokens [37]. Further, the encoding process happens for both queries and documents, yet documents are processed during indexing. The final score for a $(query, document)$ pair is calculated by summing the maximum dot products of embeddings for each shared token.

3.3 Datasets

The classification of the datasets is inspired by BeIR (Benchmarking-IR), an evaluation benchmark containing public datasets from diverse domains [38]. Furthermore, to get a more practical understanding of the differences between the chosen datasets, Table 3 presents a typical query for each dataset. To experiment with the following datasets, IR-Datasets [39] was used.

Question Answering is the task of retrieving documents that might answer the question. It can be divided into two sections. First, there is the general question answering (e.g. HotpotQA or Quora), where the corpus contains documents touching various fields. On the other hand, field-specific question answering focuses the attention on an explicit area. **FiQA - 2018** (Financial Question Answering) serves the gap in assessing the ability of models to capture the unique semantic characteristics of the financial sector [40]. The corpus was composed by crawling the threads opened in the Investment section of StackExchange between 2009 and 2017. **HotpotQA** provides a set of over 5,000,000 Wikipedia-based documents. The distinctive aspect of HotpotQA is the nature of the queries, as they require reasoning over multiple paragraphs of the documents to find the correct answer [41]. **Quora** dataset was initially released by Quora as part of an online challenge [42] to develop systems able to detect duplicate questions. This dataset contains 500,000+ documents from various domains, collected from Quora posts.

The bio-medical datasets underline more complex documents and queries using medical terms. **NFCorpus** was released in an effort to mitigate the discrepancy between user queries and medical information [43]. Further, it provides around 3,600 documents crawled from *NutritionFacts.org*, a website where doctors translate medical research papers to blog posts for the general public.

The entity retrieval domain focuses on retrieving documents relevant to the entity mentioned in the query [38]. For example, for the query "Who founded Intel?", a possible retrieved document provides information about Intel, the chip manufacturer. **DBPedia-Entity** is considered the standard entity retrieval dataset [44]. It contains 4.6 million documents collected from DBpedia, which contains Wikipedia pages stored in a knowledge graph structure.

The fact checking category is responsible for verifying a claim against a large collection of sources [38]. **FEVER** (Fact Extraction and Verification) is a fact checking dataset that includes 5.4 million documents gathered from Wikipedia. Moreover, the queries were extracted from Wikipedia pages and altered without any knowledge of the domain [45]. **SciFact** differs from FEVER as it is focused on scientific entity retrieval. The documents were retrieved from a large corpus of scientific articles encompassing different domains, while the queries were composed by experts [46].

Lastly, **MS MARCO Passage** [19] was selected for the passage retrieval task. MS MARCO consists of 8.8M documents gathered from Bing's results. The dataset includes around 1M real-world queries, divided across different sets.

3.4 Metrics

To measure the effectiveness of the selected retrieval models, recall (R), mean average precision (MAP), mean reciprocal rank (MRR), and normalized discounted cumulative gain (nDCG) were used. All mentioned metrics are defined at a depth K, illustrating the number of top recommendations considered. Recall at depth K (noted as *Recall@K*) is defined as $\frac{\sum_{i=1}^{K} \text{rel}_i}{N}$, where i is the position in the list of retrieved documents, N is the total number of relevant documents for the current query and rel_i is a binary variable illustrating whether i-th document is relevant for the given query or not. MAP is defined as $\frac{1}{r}\sum_{i=1}^{k} precision@i \cdot rel_i$, where r is the total number of relevant documents for the current query and $precision@i = \frac{\sum_{i=1}^{i} rel_i}{i}$, illustrating the fraction of relevant documents in the top-i positions. Further, MRR is defined as $\frac{1}{|Q|}\sum_{i=1}^{|Q|} \frac{1}{rank_i}$, where $|Q|$ is the number of queries, and $rank_i$ is the position of the first relevant document for query i. Lastly, nDCG is interested in the ranking quality compared to an ideal ordering. It uses DCG@K, which is defined as $\sum_{i=1}^{K} \frac{2^{rel_i}-1}{\log_2(i+1)}$, and rel_i is the relevance level of i-th document. Therefore, nDCG is defined as the ratio between DCG@K and the ideal DCG@K, obtained when the list of retrieved documents is sorted in decreasing order by relevance. To compute the metrics, the functionality offered by *PyTerrier* [47] was used.

3.5 Hyperparameter Optimization

To optimize the hyperparameter α for the interpolation-based re-ranking setting, an exhaustive grid search was performed. Eight possible alphas (0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7) were applied during the re-ranking process to evaluate their impact on the validation sets of each selected dataset. For datasets with over 1000 queries, a random sample of 500 queries was used to reduce running times.

3.6 Models Implementation

The BM25 model was implemented using the *PyTerrier* library [47]. The library uses a formula identical to the one

Table 4: Performance in a **retrieval-only** setting. Comparison of R@1000 and nDCG@10 across various retrieval models on multiple datasets. For each dataset, the best model in terms of R@1000 is marked **bold** and the best model in terms of nDCG@10 is underlined. Statistical significant improvements using two-paired tests ($p < 0.05$) are indicated using superscripts.

	BM-25[1]		TF-IDF[2]		DeepCT[3]		DeepImpact[4]		uniCOIL[5]		SPLADE[6]	
	R@1000	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10
FiQA	$0.774^{4,5}$	0.253	$0.769^{4,5}$	0.254	$0.773^{4,5}$	0.265	0.747	0.251	0.733	$0.275^{1,2,4}$	**0.842**$^{1-5}$	0.346$^{1-5}$
NFCorpus	$0.361^{3,4}$	0.322	$0.363^{3,4}$	0.323	0.351^{4}	0.320	0.325	0.313	0.445^{1-4}	0.325^{4}	**0.579**$^{1-5}$	0.340$^{1-5}$
Scifact	0.970	0.672	0.970	0.666	0.970	0.667	0.956	0.643	0.968	0.672^{4}	**0.990**$^{1-5}$	0.678^{4}
Quora	0.993^{3-5}	0.768^{3-5}	0.992^{3-5}	0.768^{3-5}	$0.990^{4,5}$	$0.750^{4,5}$	0.981	0.654^{5}	0.984^{4}	0.619	**0.999**$^{1-5}$	0.832$^{1-5}$
HotpotQA	$0.852^{2,3}$	0.513^{2}	0.850^{3}	0.512	0.840	$0.576^{1,2}$	$0.882^{1-3,5}$	$0.647^{1-3,5}$	0.850^{3}	0.622^{1-3}	**0.895**$^{1-5}$	0.678$^{1-5}$
DBPedia	$0.660^{4,5}$	0.274	$0.660^{4,5}$	0.274	$0.669^{4,5}$	$0.319^{1,2}$	0.627	$0.334^{1,2}$	0.611	$0.324^{1,2}$	**0.783**$^{1-5}$	0.424$^{1-5}$
Fever	0.925	0.427	0.925	0.428	$0.946^{1,2}$	$0.585^{1,2}$	0.967^{1-3}	$0.772^{1-3,6}$	0.969^{1-4}	0.803$^{1-4,6}$	**0.972**$^{1-5}$	0.763^{1-3}
MS MARCO Passage	0.736	0.480	0.736	0.478	0.744	$0.530^{1,2}$	0.729	0.666^{1-3}	0.737	0.631^{1-3}	**0.830**$^{1-5}$	0.729$^{1-5}$

presented in Equation 4. Regarding the free parameters, the values suggested by the library were used: $k_1 = 1.2$, $k_3 = 8$, $b = 0.75$. Similarly, for TF-IDF, the implementation offered by *PyTerrier* [47] was used, which follows the standard formula. For SPLADE [17], the implementation offered by *PyTerrier SPLADE* [48] was used in the evaluations. This library uses the *splade-cocondenser-ensembledistil* checkpoint, which was trained on the MS MARCO Passage dataset. Furthermore, this pre-trained model implements the standard SPLADE, but with a more advanced training technique [49]. For DeepCT, the implementation provided by *PyTerrier DeepCT* [50] was used and it employs the exact model from the DeepCT paper [16]. To evaluate uniCOIL [37] and DeepImpact [18], the *SPRINT* framework [51] was used, along with the pre-trained models suggested by *SPRINT*. Lastly, for the re-ranking stage of Fast-Forward Indexes, a pre-trained version of TCT-ColBERT [52], which was trained on the MS MARCO Passage dataset, was used.

The document expansion, required by DeepImpact, was performed using the base "docT5query" model [53], trained on the MS MARCO Passage dataset. Additionally, for each dataset, three queries were generated per document, as increasing the number of queries would not produce any additional meaningful terms.

4 Results

This section presents the results of the evaluations introduced in the methodology.

4.1 RQ1: What is the impact of the retrieval stage on the performance of Fast-Forward Indexes?

The recall at the top 1000 retrieved documents and the Normalized Discounted Cumulative Gain on all selected retrieval models are reported in Table 4. Furthermore, Table 5 reports the ranking quality of pairing each selected retrieval model with TCT-Colbert in an interpolation-based re-ranking setting. The first part of Table 5 reports MAP@1000 and nDCG@10 on some datasets, while the second presents MRR@10 and nDCG@10 on the remaining datasets. The split is necessary as some datasets have more relevant documents per query, making the average precision a more suitable metric. In contrast, other datasets have few relevant documents per query, making the reciprocal rank a more appropriate metric. Furthermore, it is important to acknowledge that the recall scores remain unchanged from the retrieval stage, as only the top 1000 retrieved documents are re-ranked and displayed, while the others are discarded. The key findings are split between retrieval-only performance and the performance in an interpolation-based re-ranking setting.

Retrieval-only performance. First, the traditional term-weighting models, BM25 and TF-IDF, show similar performance in terms of recall, with BM25 achieving a slightly better recall on certain datasets. Interestingly, DeepCT and DeepImpact, two neural sparse retrievers that share the idea of learning contextualized information from the terms, achieve comparable recall scores to the simpler methods. However, the additional document expansion technique employed by DeepImpact does not improve the recall; in fact, it decreases it on most datasets, except on HotpotQA. The reduced performance might be caused by the document expansion model's training on MS MARCO Passage, which, while beneficial for in-domain usage (i.e. trained and used on the same corpus), fails to provide meaningful additions in an out-of-domain environment. The boost in recall for DeepImpact on HotpotQA supports this claim. HotpotQA and MS MARCO were both intended for the general question-answering task, suggesting a similar structure of the datasets. The two bi-encoder-based retrievers, SPLADE and uniCOIL, showed mixed results in terms of recall. SPLADE presented outstanding performance compared to all other models. On the NFCorpus dataset, SPLADE retrieved 59% and 64% more relevant documents in the top 1000 compared to BM25 and DeepCT, respectively. On the other hand, uniCOIL performed similarly to or worse than traditional retrieval models on most datasets, still it showed better recall on NFCorpus.

In the retrieval-only setting, bi-encoder-based retrievers (i.e. uniCOIL and SPLADE) show increased performance in ranking quality. SPLADE outperformed all other retrieval models in nDCG@10 and its performance generalized well over multiple domains, as illustrated in Table 4. The other bi-encoder approach, uniCOIL, surpassed the traditional sparse retrievers on half of the considered datasets, while on some datasets it performed comparably to them. Yet, both uniCOIL and DeepImpact showed significantly worse ranking quality for Quora. Furthermore, all encoder-based retrieval models showed better ranking performance than the no-encoder retrievers on the MS MARCO Passage dataset. Lastly, it can be observed that there are three datasets (i.e. SciFact, Quora,

Table 5: Ranking performance in interpolation-based re-ranking setting, with MAP and nDCG@10 reported across various datasets and retrieval models. The initial **retrieval** was performed using the specified retrieval model, and **re-ranking** was done with TCT-ColBERT. In the table's first section, the best model in terms of MAP is marked in **bold**. In the second section, the top model in terms of MRR@10 is highlighted in **bold**. In the entire table, the best model in terms of nDCG@10 is underlined. Statistical significant improvements using two-paired tests ($p < 0.05$) are indicated using superscripts.

	BM-25[1]		TF-IDF[2]		DeepCT[3]		DeepImpact[4]		uniCOIL[5]		SPLADE[6]	
	MAP	nDCG@10	MAP	nDCG@10	MAP	nDCG@10	MAP	nDCG@10	MAP	nDCG@10	MAP	nDCG@10
FiQA	0.265	0.316	0.263	0.314	0.264	0.316	0.258	0.310	0.263	0.314	**0.300**$^{1-5}$	0.356$^{1-5}$
NFCorpus	0.157^{3,4}	0.335	0.156^{3,4}	0.334	0.152	0.329	0.149	0.325	0.162$^{2-4}$	0.332	**0.172**$^{1-5}$	0.345$^{3-5}$
HotpotQA	0.552	0.637	0.553	0.637	0.553	0.635	**0.611**$^{1-3,5,6}$	0.690$^{1-3,5,6}$	0.584$^{1-3}$	0.669$^{1-3}$	0.602$^{1-3,5}$	0.684$^{1-3,5}$
DBPedia	0.279^{3,5}	0.399^{3,5}	0.277	0.395^{3}	0.267	0.369	0.277^{5}	0.398^{3,5}	0.264	0.380	**0.318**$^{1-5}$	0.429$^{1-5}$
MSMarco Passage	0.435	0.684	0.438	0.692	0.444	0.695	0.454	0.722	0.436	0.694	**0.504**	0.735^{5}
	BM-25[1]		TF-IDF[2]		DeepCT[3]		DeepImpact[4]		uniCOIL[5]		SPLADE[6]	
	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10
SciFact	**0.662**	0.698	0.658	0.691	0.656	0.689	0.656	0.674	0.661	0.691	0.649	0.681
Quora	0.836$^{3-5}$	0.845$^{3-5}$	0.838$^{1,3-5}$	0.846$^{1,3-5}$	0.799^{4,5}	0.812^{4,5}	0.766^{5}	0.781^{5}	0.758	0.772	**0.849**$^{1-5}$	0.857$^{1-5}$
Fever	0.681	0.700	0.682	0.701^{1}	0.722^{1,2}	0.735^{1,2}	0.819$^{1-3,6}$	0.816$^{1-3,6}$	**0.829**$^{1-4,6}$	0.826$^{1-4,6}$	0.773$^{1-3}$	0.778$^{1-3}$

and Fever) on which all models perform similarly, achieving recall scores close to perfection. This could happen due to either of two factors: (i) these datasets have a small number of relevant documents per query or (ii) there is a high lexical overlap between queries and documents, allowing retrieving the most relevant documents.

Interpolation-based re-ranking performance. In the first part of Table 5, where multiple documents are relevant to a query, the models performed similarly in terms of nDCG, except for SPLADE, which mostly outperformed other models. Experiments on HotpotQA showed some interesting facts. First, DeepImpact performed slightly better than SPLADE in terms of both MAP and nDCG@10. Second, the neural sparse retrievers achieved better MAP than the traditional sparse retrievers. One potential reason for this is the increased recall from the retrieval stage, illustrating that the more relevant documents are retrieved, the better MAP is, given that all experiments employed the same re-ranking model. In contrast, the second part of the table indicates mixed results. For SciFact, the similar results for nDCG@10 from the retrieval stage transfer to the re-ranking stage, indicating that re-ranking, in this scenario, could not further improve the ranking quality. For Fever, the bi-encoder-based sparse retrievers significantly outperform the other models, while for Quora, no-encoder-based retrievers surpassed the more complex models.

4.2 RQ2: How does the selected retrieval model affect the latency of Fast-Forward Indexes?

Figure 1 presents the performance of several retrieval models against their query processing times on different datasets. Some interesting observations can be made.

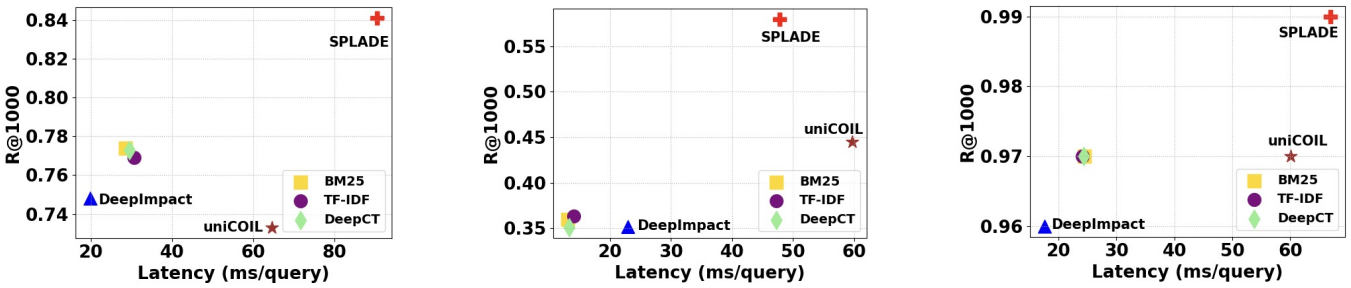
There is a clear distinction between bi-encoder-based (SPLADE and uniCOIL) sparse retrievers and the other sparse retrievers. First, the uni-encoder and the no-encoder approaches are placed on the bottom-left part of the plots, indicating a worse recall, but low latency. On the other hand, bi-encoders are on the right part of the plots, illustrating high latency and, in some cases, higher recall. Moreover, the average query processing times for the no-encoder- and uni-encoder-based retrievers range from 20 to 30ms, while the bi-encoder-based retrievers achieve an average query processing latency ranging from 60ms to approximately 90ms. This translates to simpler retrievers being about 3 times faster in query processing compared to the bi-encoder-based retrievers.

Second, SPLADE achieves the best recall from the selected models but has the highest query processing latency. Another model that shows interesting characteristics is uniCOIL, which has high query processing times but only provides average performance compared to the simpler models. The reason behind the large differences in latency between these models and the simpler ones is due to the bi-encoder architecture, which requires query encoding at search time.

Lastly, SPLADE is faster than uniCOIL on the NFCorpus dataset, although it is significantly slower on the other datasets. A particularity of NFCorpus is its shorter query length compared to the other datasets. Precisely, NFCorpus has an average query length of 21.5 terms, while FiQA and SciFact have 61.3 and 88.8 terms per query, respectively. Given that both approaches use BERT as an encoder, this comes down to the complex "WordPiece" tokenization process, together with the sparsity regulator is faster than the dimensionality reduction used by uniCOIL for shorter queries, while uniCOIL is faster for longer queries.

5 Discussion

The results show that SPLADE provides higher recall and ranking quality compared to the other models in a retrieval-only scenario. However, the increased performance comes at the cost of higher query processing latency, with SPLADE achieving almost the highest latency. Further, it can be observed that traditional retrieval models, such as BM25 and TF-IDF, still provide good recall, often being ranked second when compared to other retrieval models. Additionally, the uni-encoder approaches (e.g. DeepCT and DeepImpact) showed similar recall to the simpler approaches. When the chosen models are evaluated in an interpolation-based re-ranking setting (e.g. Fast-Forward Indexes) the ranking



(a) FiQA (b) NFCorpus (c) SciFact

Figure 1: R@1000 (Y-axis) and the average query latency (X-axis, in ms) on (a) FiQA, (b) NFCorpus, and (c) SciFact of different sparse retrievers employing both traditional and neural term-weighting techniques. The reported latency is the average processing time per query on five runs, computed on the test set. Furthermore, the processing time per query was measured as the sum of the time taken for retrieving and the time taken for re-ranking the documents. Further, the latency was measured using the *timeit* Python library. All experiments were performed on a 10-core M1 Chip with 16GB of RAM.

performance is mostly consistent with the findings from the retrieval-only stage, that is, SPLADE significantly outperformed most of the other retrieval models. Yet, the models that obtained a lower nDCG@10 score after the retrieval stage benefited from the re-ranking stage, achieving comparable performance in terms of nDCG@10 to the more expensive approaches.

The findings align with previous research that suggests bi-encoder models can enhance retrieval performance due to their ability to capture the importance of each term in the context of a document or query [17]. Curiously, Lin and Ma [37] report that uniCOIL performs substantially better than other sparse retrieval models in terms of nDCG@10 on the MS MARCO Passage ranking task. Yet, our experiments of uniCOIL, used in an interpolation-based re-ranking environment, show similar or worse nDCG@10 than simpler models on specific domains, such as SciFact or Quora. In this scenario, the performance of uniCOIL stands out even from the retrieval stage, where it was performing poorer than the other retrievers. Nevertheless, in the context of interpolation-based re-ranking, it can be argued that the chosen α value influences nDCG@10, but these experiments used optimized α values for each $(model, dataset)$ pair to overcome this issue.

The results indicated that encoder-based methods outperformed the standard methods on the MS MARCO Passage dataset. Considering that all the selected encoder-based models were previously trained on the MS MARCO Passage dataset, and evaluating them on MS MARCO shows that replacing traditional term-weighting methods, such as term-frequency or inverse document-frequency, with neural models could improve the ranking quality, as illustrated in Table 4. Yet, the reduced performance compared to no-encoder-based approaches on other datasets indicates a poor generalization of the learned term weights. Furthermore, the comparable recall scores between no-encoder-based and encoder-based retrievals are expected because encoders were used for term-weighting rather than document or query expansion. Moreover, DeepCT showed lower recall compared to BM25 on all datasets, except for the MS MARCO passage dataset, which supports the above claim and the results from a study by Thakur et al. [38].

The implications of these discoveries are significant for the design of a retrieval system. For applications that deal with specialized data, such as scientific research, where many relevant documents are desired, a bi-encoder retrieval model (e.g. SPLADE) could be beneficial as it fetches the highest number of relevant documents, despite its latency drawbacks. On the other hand, applications that handle general information and require fast retrievals, such as web search or question answering, could employ a traditional retrieval system (e.g. BM25 or TF-IDF) in an interpolation-based re-ranking setting, as they showed satisfactory performance with an average query processing time of 25ms.

6 Responsible Research

This section discusses the reproducibility of the reported results using the principles of FAIR research. Additionally, the ethical implications of the presented work will be addressed.

6.1 Reproducibility

The results are *findable* and *accessible* because all the code written during this project and the parameters used to evaluate the models are available on GitHub[1]. The libraries used in the experiments, especially PyTerrier [47] and SPRINT [51], which were responsible for evaluating the performance are also publicly available. In addition, all datasets used for evaluating the different models were offered by IR-Datasets [39], which is also accessible to the public. Lastly, all evaluated neural sparse retrievers made use of pre-trained models, which were available for download from either HuggingFace [54] or GitHub [55]. The results are *reusable* and *interoperable* because the experimental setup is described in detail in Sections 3 and 3.6 and via comments across the uploaded code. Moreover, the output of the experiments is also available, allowing for the reproduction of the presented results.

6.2 Ethical implications

Retrieval or re-ranking approaches employing neural models are prone to bias-related issues. Therefore, before using any

[1]https://github.com/cristianciacu1/neural_ranking_models

such model in production, it is recommended that the potential bias be addressed.

7 Conclusions and Future Work

Summary. This work evaluated the ranking quality of different sparse retrieval models, including traditional retrievers, which make use of mathematical formulas to compute the weights of the terms, and neural-based retrievers which employ neural models to compute the term weights (e.g. SPLADE, DeepCT,) and expand the documents during indexing (e.g. DeepImpact). This study showed that SPLADE achieves the best recall and ranking quality across all datasets. Yet, SPLADE showed a substantial increase in latency, making it unfeasible for tasks requiring fast retrieval. On the other hand, employing simpler approaches, such as BM25 and TF-IDF, in an interpolation-based re-ranking environment, proved to be considerably faster, and, in most cases, the difference in performance compared to the more expensive approaches was not critical, making them strong candidates for the ad-hoc retrieval task.

Future work. Firstly, all neural term-weighting models used in these experiments were previously trained on a specific dataset (e.g. MS MARCO). Future research should consider re-training these models on various other datasets, to allow for a more comprehensive evaluation. Secondly, future studies assessing the performance of different retrievers should include more retrieval methods, such as TextRank [56], a graph-based term-weighting model, or SpaDE [15].

References

[1] Vishal Gupta, Manoj Chinnakotla, and Manish Shrivastava. "Retrieve and re-rank: A simple and effective IR approach to simple question answering over knowledge graphs". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. 2018, pp. 22–27.

[2] Jurek Leonhardt et al. *Efficient Neural Ranking using Forward Indexes and Lightweight Encoders*. en. arXiv:2311.01263 [cs]. Nov. 2023. URL: http://arxiv.org/abs/2311.01263 (visited on 05/08/2024).

[3] Zhiguo Wang et al. "Retrieval, re-ranking and multi-task learning for knowledge-base question answering". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 347–357.

[4] Joeran Beel et al. "Paper recommender systems: a literature survey". In: *International Journal on Digital Libraries* 17 (2016), pp. 305–338.

[5] Karen Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[6] Stephen E Robertson and K Sparck Jones. "Relevance weighting of search terms". In: *Journal of the American Society for Information science* 27.3 (1976), pp. 129–146.

[7] Andrew Trotman, Antti Puurula, and Blake Burgess. "Improvements to BM25 and language models examined". In: *Proceedings of the 19th Australasian Document Computing Symposium*. 2014, pp. 58–65.

[8] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. "Simple BM25 extension to multiple weighted fields". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. 2004, pp. 42–49.

[9] José R Pérez-Agüera et al. "Using BM25F for semantic search". In: *Proceedings of the 3rd international semantic search workshop*. 2010, pp. 1–8.

[10] Guihong Cao et al. "Selecting good expansion terms for pseudo-relevance feedback". In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 2008, pp. 243–250.

[11] Jiafeng Guo et al. "A deep relevance matching model for ad-hoc retrieval". In: *Proceedings of the 25th ACM international on conference on information and knowledge management*. 2016, pp. 55–64.

[12] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. "Learning to match using local and distributed representations of text for web search". In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 1291–1299.

[13] Wen Li et al. "Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement". In: *IEEE Transactions on Knowledge and Data Engineering* 32.8 (2019), pp. 1475–1488.

[14] Sunil Arya et al. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions". In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 891–923.

[15] Eunseong Choi et al. "SpaDE: Improving sparse representations using a dual document encoder for first-stage retrieval". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 272–282.

[16] Zhuyun Dai and Jamie Callan. "Context-aware sentence/passage term importance estimation for first stage retrieval". In: *arXiv preprint arXiv:1910.10687* (2019).

[17] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking". en. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event Canada: ACM, July 2021, pp. 2288–2292. ISBN: 978-1-4503-8037-9. DOI: 10.1145/3404835.3463098. URL: https://dl.acm.org/doi/10.1145/3404835.3463098 (visited on 05/06/2024).

[18] Antonio Mallia et al. "Learning passage impacts for inverted indexes". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 1723–1727.

[19] Payal Bajaj et al. "Ms marco: A human generated machine reading comprehension dataset". In: *arXiv preprint arXiv:1611.09268* (2016).

[20] Nick Craswell et al. "Overview of the TREC 2019 deep learning track". In: *arXiv preprint arXiv:2003.07820* (2020).

[21] Nick Craswell et al. "Overview of the TREC 2020 deep learning track". In: *CoRR* abs/2102.07662 (2021). arXiv: 2102.07662. URL: https://arxiv.org/abs/2102.07662.

[22] Yang Bai et al. "SparTerm: Learning term-based sparse representation for fast text retrieval". In: *arXiv preprint arXiv:2010.00768* (2020).

[23] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. "From doc2query to docTTTTTquery". In: *Online preprint* 6.2 (2019).

[24] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *Journal of machine learning research* 21.140 (2020), pp. 1–67.

[25] Shengyao Zhuang and Guido Zuccon. "Fast passage re-ranking with contextualized exact term matching and efficient passage expansion". In: *arXiv preprint arXiv:2108.08513* (2021).

[26] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[27] Lee Xiong et al. "Approximate nearest neighbor negative contrastive learning for dense text retrieval". In: *arXiv preprint arXiv:2007.00808* (2020).

[28] Victor Lavrenko and W Bruce Croft. "Relevance-based language models". In: *ACM SIGIR Forum*. Vol. 51. 2. ACM New York, NY, USA. 2017, pp. 260–267.

[29] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. "Distilling dense representations for ranking using tightly-coupled teachers". In: *arXiv preprint arXiv:2010.11386* (2020).

[30] Rodrigo Nogueira and Kyunghyun Cho. "Passage Re-ranking with BERT". In: *arXiv preprint arXiv:1901.04085* (2019).

[31] Omar Khattab and Matei Zaharia. "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*. 2020, pp. 39–48.

[32] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[33] Alec Radford et al. "Improving language understanding with unsupervised learning". In: (2018).

[34] Zeynep Akkalyoncu Yilmaz et al. "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)*. 2019, pp. 3490–3496.

[35] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[36] Luyu Gao, Zhuyun Dai, and Jamie Callan. "COIL: Revisit exact lexical match in information retrieval with contextualized inverted list". In: *arXiv preprint arXiv:2104.07186* (2021).

[37] Jimmy Lin and Xueguang Ma. "A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques". In: *arXiv preprint arXiv:2106.14807* (2021).

[38] Nandan Thakur et al. "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models". In: *arXiv preprint arXiv:2104.08663* (2021).

[39] AI2 - Allen Institute for AI. *ir_datasets: A repository of datasets for information retrieval research*. https://github.com/allenai/ir_datasets. 2024.

[40] Macedo Maia et al. "Www'18 open challenge: financial opinion mining and question answering". In: *Companion proceedings of the the web conference 2018*. 2018, pp. 1941–1942.

[41] Zhilin Yang et al. "HotpotQA: A dataset for diverse, explainable multi-hop question answering". In: *arXiv preprint arXiv:1809.09600* (2018).

[42] Kaggle. *Quora Question Pairs*. Accessed: 2024-06-23. 2024. URL: https://www.kaggle.com/c/quora-question-pairs.

[43] Vera Boteva et al. "A full-text learning to rank dataset for medical information retrieval". In: *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer. 2016, pp. 716–722.

[44] Faegheh Hasibi et al. "DBpedia-entity v2: a test collection for entity search". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2017, pp. 1265–1268.

[45] James Thorne et al. "FEVER: a large-scale dataset for fact extraction and VERification". In: *arXiv preprint arXiv:1803.05355* (2018).

[46] David Wadden et al. "Fact or fiction: Verifying scientific claims". In: *arXiv preprint arXiv:2004.14974* (2020).

[47] Terrier Team. *PyTerrier: Python Integration for Terrier Information Retrieval Platform*. https://github.com/terrier-org/pyterrier. 2024.

[48] Craig Macdonald. *pyt_splade: PyTerrier-SPLADE Integration*. https://github.com/cmacdonald/pyt_splade. 2024.

[49] Thibault Formal et al. "From distillation to hard negative sampling: Making sparse neural ir models more effective". In: *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 2022, pp. 2353–2359.

[50] Terrier Team. *PyTerrier-DeepCT: DeepCT Integration with PyTerrier*. https : / / github . com / terrierteam / pyterrier_deepct. 2024.

[51] Nandan Thakur. *Sprint: A Repository for Sprint Projects*. https : / / github . com / thakur - nandan / sprint. 2024.

[52] Hugging Face. *TCT-ColBERT MS MARCO*. 2024. URL: https : / / huggingface . co / castorini / tct_colbert - msmarco.

[53] Castorini. *doc2query-t5-base-msmarco*. https : / / huggingface . co / castorini / doc2query - t5 - base - msmarco. Accessed: 2024-06-23. 2021.

[54] Hugging Face. *Hugging Face: The AI community building the future*. 2024. URL: https : / / huggingface . co/.

[55] GitHub, Inc. *GitHub: Let's Build from Here*. 2024. URL: https://github.com/.

[56] Rada Mihalcea and Paul Tarau. "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.