Towards Effective, Efficient and Explainable Neural Information Retrieval

Thibault Formal

Sorbonne Université
Institut des Systèmes Intelligents et de Robotiques (ISIR)
Paris, France

Towards Effective, Efficient and Explainable Neural Information Retrieval

---

PhD Thesis
By **Thibault Formal**

Supervised by

Benjamin Piwowarski
Stéphane Clinchant

January 2023

Abstract

Information Retrieval is undergoing a paradigm shift. Keyword-based approaches, which have withstood the test of time, are challenged by a new generation of neural retrievers based on large Pre-trained Language Models. Such approaches can faithfully represent the content of documents and queries beyond the words they use – delivering the promises of a truly semantic search experience. As models grow larger, they also become more opaque. This is a true obstacle to their widespread adoption in commercial search engines, which are increasingly faced with providing users with transparent, trustworthy, and explainable results.

In this thesis, we first propose an original approach to the ad-hoc retrieval problem by learning how to represent queries and documents as sparse vectors in the vocabulary space. This results in a model that is *effective*, *efficient*, *robust*, and whose representations can be *interpreted* by design. We then propose to analyze neural ranking models from an Information Retrieval perspective, by focusing on *lexical match* and *term importance*. We first show how ColBERT – a state-of-the-art approach – relies on such aspects despite its semantic nature. We additionally extend the findings to other models by showing how the ability to perform keyword matching is architecture-dependent and heavily influenced by the presence of query terms in the training set – questioning the generalization capabilities of neural ranking models when it comes to exactly matching important query terms.

Acknowledgements

My Ph.D. journey started three years ago, only two months before the global pandemic of COVID-19. This was the start of an amazing – and sometimes difficult – adventure that allowed me to develop into a true researcher. It would not have been possible without the support of many colleagues, friends, and family members.

First and foremost, I want to thank my two supervisors, Benjamin Piwowarski and Stéphane Clinchant, for trusting me. I have learned more from you than I thought possible. You taught me how to do research and pursue new challenging directions and provided me with constant guidance and support. I could not have wished for better mentors.

I also wish to thank all the people I had the chance to work with, especially Carlos, Simon, and Hervé, as well as the remaining members of the Search & Recommendation team at Naver Labs Europe: Jean-Michel, Till, Romain, and Thibaut. More generally, thanks to everyone I had the chance to collaborate or interact with: Quentin, Michel, Arnaud, Matt, Arthur, Vaishali, Matthias, Augustin, and many more. And the members of the MLIA team during my (too short) visits to the lab. I have also been fortunate to meet wonderful researchers and friends during travels and conferences – special s/o to you, David.

A tous mes amis, qui ont contribué par bien des aspects à cette thèse – Pierre, Julien, Simon, Isma, Paul, Nath, Lambert, Neil et tous ceux que je n'ai pas la place de citer mais qui se reconnaîtront.

A ma famille, pour son soutien inconditionnel depuis toutes ces années – sans avoir une idée très claire de ce que je peux bien "chercher".

Et enfin, à Alix, merci tout simplement. A cette vie qui nous attend.

Albums without which the writing of this thesis would have been impossible:
- Miles Davis, *In A Silent Way*
- Bowery Electric, *Lush Life*
- Croatian Amor, *The World*
- Turnstile, *Nonstop Feeling*
- Ulver, *Bergtatt*

“*I've had a total re-calibration of my mind, you know. I mean, it's like, I've been banging my head against this 19th-century type, um, what? Thought mode? Construct? Human construct? Well, the wall doesn't exist. It's not there, you know. I mean, they tell you, look for the light at the end of the tunnel. Well, there is no tunnel. There's just no structure. The underlying order is chaos.*

Denise Montgomery as “Having a Breakthrough Day” in Slacker directed by Richard Linklater (1990)

“*I see more than others do because I know where to look.*

Flavor text from the *Magic: The Gathering* card Brainstorm (Mercadian Masques edition, 1999)

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Approximate Nearest Neighbors
BoW	Bag-of-Words
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
IBN	In-Batch Negatives
IDF	Inverse Document Frequency
IR	Information Retrieval
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
LSTM	Long Short-Term Memory
LTR	Learning-to-Rank
MIPS	Maximum Inner Product Search
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptron
MRR	Mean Reciprocal Rank
MSA	Multi-Head Self-Attention
nDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
NN	Neural Network
OOD	out-of-domain
P	Precision
PLM	Pre-trained Language Models
PRF	Pseudo-Relevance Feedback

QPP	Query Performance Prediction
R	Recall
REML	Retrieval-Enhanced Machine Learning
RR	Reciprocal Rank
RSJ	Robertson-Sparck Jones
seq2seq	Sequence-to-Sequence
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
VSM	Vector Space Model

Chapter 1

Introduction

Information Retrieval (IR), as an Academic field of study, can be defined as the task of finding *documents* of an *unstructured* nature that satisfy an *information need* from within *large collections* [Manning et al. 2008]. That's what search engines such as Naver or Google are all about, providing quick and easy access to the ever-growing quantity of information available online. We – as *users* – rely on them to find answers to our questions, discover new products and services, and stay informed about the world around us. They have become an essential tool for billions of individuals, as well as businesses and organizations, to find information and make informed decisions. But what we take for granted now is the result of decades of research starting from the 1950s, as well as significant engineering efforts to deploy such systems at scale.

Users engage with retrieval systems to navigate through various content sources, from Web pages, images, videos, e-commerce products, scientific articles – and the list goes on. Despite this increasing diversity, text-based retrieval remains one of the core activities of most search engines. Much of the information available online is still in the form of text, as it is a versatile and universal medium for communication and information sharing, for people of all cultures and backgrounds. Users also naturally express their information needs as natural language (or keyword-based) queries. For such a task – traditionally referred to as *ad-hoc* IR – the search system returns a list of text *documents* such as Web pages from a usually large collection[1], ordered by their estimated relevance to users' *queries*.

[1] Think about the whole Web.

Keyword-based models developed in the 1990s have withstood the test of time and continue to power every search system. They are simple, efficient, and yet effective. *Lexical matching* relies on the hypothesis that relevant documents should contain (important) query words – which holds true in many cases. For example, given the query:

$$q \rightarrow \text{“}\textit{who is robert gray}\text{”}$$

it is reasonable to assume that the entity of interest (“*robert gray*”)

should appear in relevant documents. This fundamental rationale is the central point of decades of IR research.

These approaches are, however, limited to simple statistical properties of words and cannot accurately model their complex relationships. For instance, documents containing *synonyms* of important query words should be able to answer the underlying information need equivalently. In the query:

$$q \rightarrow \text{“}\textit{average annual income data analyst}\text{”}$$

“*income*” could seemingly be replaced by “*salary*” without changing what the user is looking for. Such an issue of searchers relying on different terms to describe their needs[2] – compared to what is found in relevant documents – is commonly referred to as vocabulary mismatch [Furnas et al. 1987], and represents a major challenge for lexical approaches. Similarly, queries can be ambiguous or intrinsically difficult, and having a proper understanding of the *context* to characterize users' *intents* should help systems provide a better search experience [Buckley 2004]. To give a sense of intuition, the query:

$$q \rightarrow \text{“}\textit{why did the us volunterilay enter ww1}\text{”}$$

seems difficult to answer solely based on occurrences of its terms in documents, as it is required, for instance, to infer that “*us*” actually refers to the United States[3].

The Information Retrieval community has dedicated much effort to overcoming such issues, with various degrees of success. While effective, methods like Query Expansion [Lavrenko and Croft 2001] only circumvent the problem. Going beyond keyword matching requires having a proper understanding of the meaning (or *semantics*) of queries and documents, which is more in line with how humans think and process information. The recent breakthroughs brought by Deep Learning (DL) in the field of Natural Language Processing (NLP) have nurtured the hope of a new generation of *semantic search* systems, where models based on neural networks would be able to faithfully *represent* the content of documents and queries beyond the words they use. Word embeddings like word2vec [Mikolov et al. 2013] have significantly influenced the neural IR shift. These methods, based on distributional semantics, allow representing *similar* words (i.e., occurring in similar *contexts*) into close regions of a continuous embedding space. This enables, for instance, to capture word associations such as *synonymy* (different words with the same meaning) or *polysemy* (same word with different meanings). Despite all the promises, they have, however, shown limited success in Information Retrieval [J. Lin 2019; W. Yang et al. 2019a].

We had to wait for the Pre-trained Language Models (PLM) revolution to truly witness the development of neural IR. These models, based on the transformer architecture [Vaswani et al. 2017], acquire

[2] Sometimes, users don't even know what they are looking for!

[3] These three queries come from the TREC 2019 Deep Learning dataset (see Section 2.3.2.1). For the former (resp. latter two), BM25 [S. Robertson and Zaragoza 2009; Stephen E. Robertson et al. 1994] – a keyword-based model – performs well (resp. poorly).

general *contextualized* knowledge about language through unsupervised *pre-training* on large text corpora and only need to be *fine-tuned* on downstream tasks with a limited amount of training data. Models like BERT [Devlin et al. 2019] have irreversibly impacted IR – way beyond what the community thought would be possible a few years back – by ultimately shifting the state of the art on traditional retrieval benchmarks [Dai and Callan 2019b; MacAvaney et al. 2019a; R. F. Nogueira and Cho 2019]. The fundamental paradigm shift, however, occurred with the emergence of a new generation of neural models tackling the *retrieval* step in multi-stage ranking pipelines – holding the promise to replace well-established IR models in modern search engines. These approaches aim to address the inherent limitations of keyword matching by directly retrieving, from the complete collection, relevant documents that would have been missed otherwise. Due to the scale of the problem, they have to cope with strict efficiency requirements to provide users with a smooth search experience. Improving the accuracy of search results is, therefore, not enough for systems to be useful, and efficiency constraints have also driven model design. Within the last three years, many approaches have been proposed for this task, leading to unprecedented results on various benchmarks. Both aspects – *effectiveness* and *efficiency* – are central to this thesis.

Retrieval models tend to be increasingly based on machine learning methods, and thus, more *black-box*. In particular, as their reliance on Pre-trained Language Models increases, they become more opaque – making it difficult to provide clear and accurate explanations to users about why a particular result was returned for a given query. Yet, public awareness of privacy, trustworthiness, or fairness issues requires more transparency from search companies. This tension is an obstacle to their widespread adoption and urges the development of dedicated analysis tools or interpretable architectures. Additionally, there have been recent concerns about the ability of neural ranking models to *generalize* [Thakur et al. 2021], either for new terms or domains. Knowing that many production systems use (or will use) models based on PLM, while being exposed to new documents and queries every day, robustness is thus a critical aspect that must be assessed. Interpretability – either from a *structural* or *behavioral* perspective – as well as generalization capabilities of models, are also central to this thesis.

1.0.1 Research Questions

Motivated by these paradigm shifts and the ongoing challenges in neural Information Retrieval, this thesis contributes to answering the following two research questions:

- **RQ1** Can we design *effective*, *efficient*, *interpretable* and *robust* neural IR models to replace proven traditional lexical approaches?

- **RQ2** Despite their semantic nature, do neural ranking models based on PLM rely on well-established IR properties?

While being distinct, these two questions are also interleaved. During the course of this Ph.D., insights drawn from analyzing models have initiated a series of experiments around architectures that explicitly encode the studied phenomena *by design* – eventually resulting in a model meeting the requirements of **RQ1**.

1.0.2 Outline and Structure

We now describe the organization of this thesis, structured into three main parts.

In Chapter 2, we provide the necessary background for readers to follow this document. In particular, we cover the evaluation of retrieval results, before discussing how to *represent* text – from traditional lexical approaches to transformers – and link such representations to the different types of ranking models. We finally provide an in-depth review of Pre-trained Language Models in the context of Information Retrieval.

In Chapter 3, we present the main contribution of this thesis. We propose SPLADE, a *sparse* bi-encoder model that learns query and document representations grounded in the vocabulary. SPLADE is *effective*, *efficient*, and its *representations* can be interpreted. We additionally study the effect of various training techniques, such as distillation or hard-negative sampling, to improve its effectiveness, leading to state-of-the-art results in both in- and out-of-domain settings.

In Chapter 4, we focus on understanding the properties of neural Information Retrieval models. In particular, we study two specific behavior, namely *lexical match* and *term importance*. We first propose indicators specifically tailored for the ColBERT model and show how it still implicitly relies on such aspects – despite its semantic nature. In a second contribution, we delve deeper into the ability of neural rankers to perform lexical match *off-the-shelf* and show how the training frequency of query terms heavily influences such a property.

1.0.3 Contributions

Besides this introduction and Chapter 2, the content of this manuscript is borrowed from the following contributions, ordered by publication date:

- [Formal et al. 2021b] *A White Box Analysis of ColBERT*. **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.
  ECIR'21. Short paper. Best short paper award ♣.
  → Chapter 4

- [Formal et al. 2021d] *Une Analyse du Modèle ColBERT*. **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.

CORIA'21. Extended Abstract.
→ Chapter 4

- [Formal et al. 2021c] *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking.* **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.
SIGIR'21. Short paper.
→ Chapter 3

- [Formal et al. 2021a] *SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.* **Thibault Formal**, Carlos Lassance, Benjamin Piwowarski and Stéphane Clinchant.
arXiv pre-print arXiv:2109.10086. 2021.
→ Chapter 3

- [Lassance et al. 2021b] *Naver Labs Europe (SPLADE) @ TREC Deep Learning 2021.* Carlos Lassance, **Thibault Formal**, Benjamin Piwowarski, Arnaud Sors, and Stéphane Clinchant.
TREC'21.
→ Chapter 3

- [Formal et al. 2022b] *Match Your Words! A Study of Lexical Matching in Neural Information Retrieval.* **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.
ECIR'22. Short paper.
→ Chapter 4

- [Formal et al. 2022c] *Match Your Words! A Study of Lexical Matching in Neural Information Retrieval (extended abstract).* **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.
CIRCLE'22. Extended Abstract.
→ Chapter 4

- [Formal et al. 2022a] *From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective.* **Thibault Formal**, Carlos Lassance, Benjamin Piwowarski and Stéphane Clinchant.
SIGIR'22. Short paper. *This work was also accepted at the Workshop on Reaching Efficiency in Neural Information Retrieval at SIGIR'22.*
→ Chapter 3

- [Formal et al. *Under review*] *Towards Effective and Efficient Sparse Neural Information Retrieval.* **Thibault Formal**, Carlos Lassance, Benjamin Piwowarski and Stéphane Clinchant.
ACM Transactions on Information Systems. *Under review.*
→ Chapter 3

In addition, this thesis also indirectly benefitted from the following publications:

- [Formal et al. 2020] *Naver Labs Europe @ TREC Deep Learning 2020.* **Thibault Formal**, Benjamin Piwowarski and Stéphane Clinchant.
  TREC'20.

- [Lassance et al. 2021a] *Composite Code Sparse Autoencoders for First Stage Retrieval.* Carlos Lassance, **Thibault Formal** and Stéphane Clinchant.
  SIGIR'21. Short paper.

- [Lupart et al. 2023] *MS-Shift: An Analysis of MS MARCO Distribution Shifts on Neural Retrieval.* Simon Lupart, **Thibault Formal** and Stéphane Clinchant.
  ECIR'23. Long paper.

- [Faggioli et al. 2023] *QPP for Neural IR: Are We There Yet?.* Stéphane Clinchant, Guglielmo Faggioli, Nicola Ferro, **Thibault Formal**, Stefano Marchesin and Benjamin Piwowarski.
  ECIR'23. Long paper.

- [Hai Le et al. 2023] *CoSPLADE: Contextualizing SPLADE for Conversational Information Retrieval.* Nam Le Hai, Thomas Gerald, **Thibault Formal**, Jian-Yun Nie, Benjamin Piwowarski and Laure Soulier.
  ECIR'23. Long paper.

Finally, code and models have been open-sourced†. In particular, SPLADE models on Naver's HuggingFace Hub‡ have been downloaded a large number of times[4]. SPLADE has also been successfully deployed in different search services at Naver – highlighting its value in real-world scenarios.

† https://github.com/naver/splade

‡ https://huggingface.co/naver

[4] Around $100k$ within the last month for `naver/splade-cocondenser-ensembledistil` (April 12, 2023).

Chapter 2

Representation Learning for Information Retrieval

Outline

This chapter provides a broad introduction to Information Retrieval, emphasizing the new wave of ranking systems based on Pre-trained Language Models. We cover general concepts, such as evaluating search results, before discussing how to represent text – from traditional lexical approaches to transformers – and link such representations to the different types of ranking models. We finally provide an in-depth review of Pre-trained Language Models such as BERT in the context of Information Retrieval.

2.1 Introduction

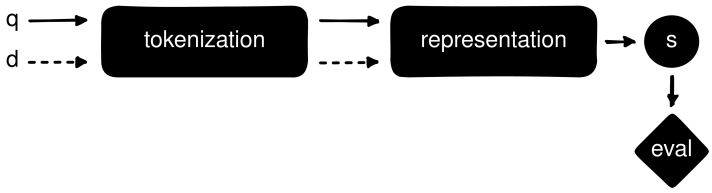
Information Retrieval systems are central to many information access services and can operate in very different settings involving text, images, speech, multi-modal, multi-lingual contents, etc. In the remainder of this document, we are interested in traditional text-based search, where users submit an open-domain (or "ad-hoc") query to the system and expect a list of ranked text documents in return. A document is defined in a very broad sense here: it could refer to any item of interest – depending on the context – such as news articles, scientific publications, or the HTML content of Web pages.

Definitions and notations In ad-hoc search, information needs are usually expressed as short keyword-based queries q – although users tend to increasingly rely on proper natural language queries[5]. Let $\mathcal{C} = \{d_1, d_2, ..., d_{|\mathcal{C}|}\}$ be the collection of documents. The goal of an IR system $\mathcal{S}$ is to return to the user a *ranked* list of documents $\pi_{\mathcal{S}}(q) =$

[5] For instance, spoken queries to a voice assistant.

$(d_1, d_2, ...)_{\mathcal{S}}$, where relevant information is expected to be found, preferably in the top of the list. This is usually accomplished by assigning a relevance score $s(q, d)$ for documents in $\mathcal{C}$, and ordering documents in descending order of their scores. Figure 2.1 shows a simplified view of the different steps composing an IR pipeline. In Section 2.4.2, we cover how text (i.e., a sequence of characters) is usually processed into small representative units (*tokenization*). Section 2.4 focuses on techniques dedicated to *represent* text for the retrieval task. Sections 2.5 and 2.6 introduce traditional and neural Information Retrieval models used to *score* documents. Section 2.2 covers the *evaluation* of retrieval results. In this Chapter, we aim to provide a light description of various key IR concepts that are necessary to follow this thesis – interested readers can refer to [Amini and Gaussier 2013; Manning et al. 2008] for additional details.

Figure 2.1: Simplified view of the Information Retrieval pipeline. Queries (q) and documents (d) undergo the same steps, but some parts for the latter (dotted lines) can usually be done offline, i.e., prior to query inference.



We are interested in supervised ranking models based on Neural Networks (NNs). We define a linear layer as $f(x) = Wx + b$. Similarly, a standard two-layer Multi-Layer Perceptron (MLP) can be defined as $f(x) = W_2\text{ReLU}(W_1x + b_1) + b_2$, where the ReLU activation is given by $\text{ReLU}(x) = \max(0, x)$. We sometimes need to model a distribution from a series of scalars $z = (z_1, ..., z_n)$. To this end, we rely on the softmax:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \forall i \in \{1, ..., n\}$$

We follow a standard supervised setting: given a loss function $\mathcal{L}$, we train parametrized ranking models f_θ with Stochastic Gradient Descent (SGD) – or a variation of it – by updating parameters as follows:

$$\theta \leftarrow \theta - \lambda \widetilde{\nabla}_\theta \mathcal{L} \tag{2.1}$$

where λ is a hyperparameter (the step size, or learning rate), and $\widetilde{\nabla}_\theta$ the gradient estimation over a batch $\mathcal{B}$ of b samples. Please refer to [Goodfellow et al. 2016] for an in-depth review of Deep Learning techniques.

2.2 Evaluating IR Systems

We have loosely defined an Information Retrieval system as returning a ranked list of documents, from a generally large collection $\mathcal{C}$, for a given information need. In Web search, it is usually expressed by the user in the form of a short, keyword-based, text query[6]. Before diving into how we can design effective ranking models, we first need a way to *evaluate* and *compare* different rankings, i.e., ordered lists of documents. In short, we want to know what makes a good ranking. Intuitively, systems should return documents "about" the given topic, ideally putting them on top of the list. Relevance is a core concept in IR. It is, however, intricate and may be highly subjective. We are usually interested in *topical relevance*, i.e., to which extent the document topics cover the information need. This binary view can sometimes be extended to represent relevance at multiple levels (or grades), for instance, from 0 (non-relevant) to 4 (perfectly relevant). Such relevance judgments are usually obtained through evaluation campaigns, where annotators are required to assess the relevance of a pool of documents given an information need (Section 2.3). As it is infeasible to annotate every document in $\mathcal{C}$ – w.r.t. a given topic – pooling strategies are adopted to select a small set of likely relevant documents for every topic. Thus, we typically consider as non-relevant documents that have not been annotated[7]. Relevance judgments can also be derived from implicit feedback, such as click logs. In this case, the signal is abundant and cheaper to obtain but also noisier and subject to various biases [Joachims et al. 2007]. In the following, we assume access to such labels – regardless of how they have been obtained – and introduce different evaluation metrics used in this thesis.

Let us first consider the set of ordered documents (or ranked list) and their corresponding scores, returned by a given system for a query q: $\pi_L = (d_i, s_i)_{i=1:L}$ – or equivalently by considering only the ranks, $\pi = (i, s_i)$[8]. We have $d_1 \succ d_2 \succ ... \succ d_L$, and $s_1 > s_2 > ... > s_L$. L is the number of results retrieved by the system. In practice, it is usually unspecified but implicitly constrained by the cut-off (or rank) k at which some standard measures are evaluated. We also assume to have access to relevant judgments (*qrels*) $y_i = \text{rel}(d_i, q)$. In the case of binary relevance annotations where $y_i \in \{0, 1\}$[9], we define Precision (P) as the fraction of relevant documents in the ranked list:

$$\text{P}(q, \pi) = \frac{\sum_{i \in \pi} y_i}{|\pi|} \tag{2.2}$$

Similarly, we define Recall (R) as the fraction of relevant documents from the complete collection $\mathcal{C}$ that are retrieved by the system:

$$\text{R}(q, \pi) = \frac{\sum_{i \in \pi} y_i}{\sum_{j=1:|\mathcal{C}|} y_j} \tag{2.3}$$

[6] Note that while a query is merely a poor expression of the complex underlying information need, we make no further distinction between the two.

[7] Which can lead to unfair evaluations, see also Section 2.3.

[8] To lighten notations, we remove the dependency on the query q, and the system $\mathcal{S}$; one should keep in mind that documents d_i denote different documents depending on those.

[9] Graded relevance judgments can be binarized based on a threshold.

We are usually interested in such measures evaluated at a given cut-off k:

$$\text{P@}k = \frac{\sum_{i \in \pi_k} y_i}{k} \tag{2.4}$$

$$\text{R@}k = \frac{\sum_{i \in \pi_k} y_i}{\sum_{j=1:|\mathcal{C}|} y_j} \tag{2.5}$$

Remark

Precision and Recall illustrate different ranking trade-offs: you can't always optimize for both! Optimizing for Precision can actually hurt Recall. In the standard two-stage ranking pipeline (Section 2.5.2), we generally target *retrievers* that optimize for Recall, and *re-rankers* that further optimize for Precision-oriented metrics. Furthermore, the cut-off for P is usually low, e.g., $k < 50$, while it is higher for R, for instance, $k = 1000$ in the TREC Deep Learning tracks (Section 2.3.2.1).

Although useful, these metrics do not consider the documents' rank (or position) in the ranked list. Intuitively, a system would be preferred if it puts relevant documents in the very first positions. We can thus have systems with the same P@20, but a substantially different quality of results. Reciprocal Rank (RR) explicitly includes the rank information and is defined as:

$$\text{RR}(q, \pi) = \frac{1}{r(i)} \tag{2.6}$$

where $r(i)$ gives the rank of the first relevant document in π. RR is equal to 0 if no relevant document is found, and can similarly be computed at a given cut-off.

Remark

There are debates regarding the value of RR as a useful IR metric [Ferrante et al. 2021]. However, it is the official metric on the MS MARCO dataset due to the sparse nature of its annotations (Section 2.3.2.1).

So far, we have only defined binary evaluation measures which do not consider nuances between relevance grades. Normalized Discounted Cumulative Gain (nDCG) [Järvelin and Kekäläinen 2002] is a metric that accounts for both positions and the multiple degrees of relevance, for instance, when $y_i \in \{0, 1, 2, 3, 4\}$. DCG is defined as follows:

$$\text{DCG}(q, \pi) = \sum_{i \in \pi} \frac{2^{y_i} - 1}{\log_2(1 + i)} \tag{2.7}$$

As it is not bounded, nDCG is defined by normalizing DCG by its maximum possible value, achieved for the perfect ranking π^*:

$$\text{nDCG}(q, \pi) = \frac{\text{DCG}(q, \pi)}{\text{DCG}(q, \pi^*)} \tag{2.8}$$

Thus, good systems – being closer to the perfect ordering of documents – have nDCG values close to 1. It has become one of the most popular evaluation measures in IR, and is typically evaluated at a given cut-off – nDCG@10 is, for instance, the official measure for the TREC Deep Learning tracks (Section 2.3.2.1).

We have introduced metrics at the query (or topic) level; when evaluating or comparing models, we usually have access to a set of queries. Although it has recently been the subject of debates [Ferrante et al. 2021], averaging results – supposing a uniform distribution of topics – remains the standard way of producing a single performance measure that can be used to compare different systems, for instance[10]:

$$\text{nDCG} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \text{nDCG}(q, \pi^{(q)}) \tag{2.9}$$

[10] We generally report the aggregated results using the same name, except for RR which becomes Mean Reciprocal Rank (MRR).

2.3 Test Collections

In this Section, we describe what constitutes IR collections, how relevance judgments are obtained, and introduce the standard test collections used to train and evaluate models in this thesis.

2.3.1 TREC

A test collection comprises a document collection $\mathcal{C}$, a set of topics $\mathcal{Q}$, and the corresponding relevance judgments [Manning et al. 2008]. Provided with such ingredients, it is then possible to evaluate and carry out comparisons between different IR systems. When enough topics and *qrels* are available, it is also possible to *train* ranking models. Documents and queries can be "easy" to obtain, but gathering relevance judgments is, however, cumbersome. The Text REtrieval Conferences (TREC)[11], organized by the U.S. National Institute of Standards and Technology (NIST), has been pioneering large-scale, community-wide evaluations since 1992, providing the IR community with reusable test collections. Each year, participants – generally groups of researchers – participate in different evaluation campaigns (or tracks), such as ad-hoc retrieval. Results from their systems – a ranked list of items for each test query – are then used in the annotation process to build the relevance judgments.

[11] https://trec.nist.gov/

Indeed, as already mentioned in Section 2.2, it is impossible to collect exhaustive judgments (i.e., annotating each (q, d) pair) for large

document collections. Instead, assessors are usually provided with a *pool* of documents. In a nutshell, evaluation is carried out for the top-k documents (for each query) returned by each track participant. Afterward, results from the annotation process are used to evaluate systems that participated in the track, or build a test collection that can be used for years. The advent of neural ranking models based on PLM has motivated the creation of specific TREC tracks (aka TREC Deep Learning), starting from 2019, with the goal to study Information Retrieval in a large training data regime [Craswell et al. 2021a, 2022, 2020, 2021b].

Pooling TREC-style pooling is not without limitations. There potentially exist relevant documents that have been unjudged (i.e., not retrieved by participating systems) and thus considered as non-relevant. For instance, "old" test collections have been built from systems relying on keyword matching. However, new semantic methods (e.g., based on PLM, see Section 2.6) are able to retrieve relevant documents which do *not* contain query terms and thus have likely not been judged. More generally, there is a bias towards systems that participated in the evaluation campaign – making it vital for IR researchers working on new techniques to participate in such challenges actively. These issues have recently been brought up-to-date [Thakur et al. 2021; L. Xiong et al. 2021]. Some informative metrics, such as the hole@k – which computes the fraction of retrieved documents by a system without labels – can be used to quantify this problem for a given system. However, TREC conferences still provide the most robust testbed for IR evaluation due to the huge effort put into the annotation process. Other datasets used by the IR/NLP communities (sparsely judged, automatically built, etc.) exhibit such biases in a more systematic way.

2.3.2 Standard Test Collections

We introduce in the following common test collections used in this thesis, for which statistics can be found in Table 2.1.

dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{Q}_{\text{train}}\rvert$	$\lvert\mathcal{Q}_{\text{test}}\rvert$	A/T
MS MARCO passages	8,841,823	502,939	6,980	1.1
TREC DL 2019 (p)	8,841,823	-	43	215.3
TREC DL 2020 (p)	8,841,823	-	54	210.9
MS MARCO documents	3,213,835	367,013	5,193	1
MS MARCO passages v2	138,364,198	277,144	53	204.3
MS MARCO documents v2	11,959,635	322,196	57	229.1
Robust04	528,155	-	249	1245.6
TREC-COVID	171,332	-	50	1326.7

Table 2.1: Collection statistics. For MS MARCO passages, we consider the *dev* set as our "official" test set. A/T denotes the average number of annotated documents per topic.

2.3.2.1 MS MARCO

MS MARCO was released in 2016 as a Question Answering dataset based on Bing's anonymized search query logs [Bajaj et al. 2016]. It was later adapted for the ranking task and has largely contributed to the success of neural retrieval based on PLM, by providing the IR community with the first open *large-scale* dataset to train ranking models. It contains around $8.8M$ passages, $500k$ natural language training queries with corresponding *qrels*[12], 6980 dev queries, as well as 6837 test queries with hidden relevance judgments[13]. Each query has approximately one relevant passage – the one used to compose the final answer, from the top-10 passages extracted from documents retrieved by Bing. The annotations are inherently incomplete, as there might be relevant documents that have not been used to formulate the answer, and are thus considered as non-relevant. Such annotation type is usually referred to as "shallow" or "sparse" contrary to the TREC-style labeling introduced in Section 2.3.1, where the annotation is generally "deep". While it introduces some issues for both training and evaluating models (for instance, the abundance of false negatives as shown in [Qu et al. 2021]), the large-scale nature of the collection – in terms of the number of annotated queries – makes it the perfect testbed to train large neural ranking models. A document collection was later built from MS MARCO by crawling source documents that contained passages in the passage task. As the crawl occurred at a different time, the document collection is thus incomplete. It contains around $3.2M$ documents, $300k$ training queries, and 5193 dev queries. The relevance mapping is based on the occurrence of a positive passage in the document. Starting in 2019, TREC launched the Deep Learning tracks [Craswell et al. 2021a, 2022, 2020, 2021b] based on MS MARCO, to study the behavior of ranking models in large training data regimes, and partly alleviate the evaluation issues caused by shallow judgments. Based on the approach described in Section 2.3.1, a deep evaluation for a small set of

[12] There are actually more training queries (around $800k$). Still, we only account for the ones with relevance judgments.

[13] Which constitute the official leaderboard: https://microsoft.github.io/MSMARCO-Passage-Ranking-Submissions/leaderboard/

Queries
definition of tangent
what is wifi vs bluetooth
do goldfish grow
tracheids are part of _____.
who is robert gray

Table 2.2: Sample of queries taken from the MS MARCO Deep Learning 2019 test set.

queries (43 and 54 for the 2019 and 2020 passage tracks, respectively – examples can be found in Table 2.2) is provided by NIST assessors. For the 2021 track, TREC introduced a newer version of MS MARCO – called MS MARCO v2 – whose goal was to provide a larger and cleaner collection, while unifying the passage and document datasets by introducing a passage-document mapping. It contains $138M$ passages and $12M$ documents (around 16 and 4 times more, respectively). It, however, failed at producing a reliable test collection [E. M. Voorhees et al. 2022a].

2.3.2.2 BEIR Benchmark

The ability of neural ranking models to *generalize* to new terms or domains is a vital aspect of their actual performance, and has been the focus of various works in the last two years. BEIR [Thakur et al. 2021] – for Benchmarking-IR – is a heterogeneous benchmark whose purpose is to evaluate neural IR models in a *zero-shot* – or out-of-domain (OOD) – setting. The benchmark consists of a test suite of 18 datasets, each containing documents, queries, and corresponding *qrels*. They are used to evaluate models in zero-shot, i.e., without any sort of training based on those datasets – for instance, bio-medical retrieval on TREC-COVID [E. Voorhees et al. 2021] or entity retrieval on DBPedia [Hasibi et al. 2017]. The selected datasets were chosen based on three factors: diversity in tasks, domains, and difficulty. This makes BEIR really challenging as, unlike classical evaluation settings where the collection usually stays unchanged, both queries and documents are "new". Detailed statistics of the datasets can be found in Table 2.3, as well as in [Thakur et al. 2021]. We further provide additional details on two specific collections that appear in the benchmark: Robust04 and TREC-COVID. The former is still considered one of the most reliable ad-hoc collections. We rely on TREC-COVID to carry out analyses in Chapter 4.

dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{Q}_{\text{train}}\rvert$	$\lvert\mathcal{Q}_{\text{test}}\rvert$
TREC-COVID	171,332	-	50
BioASQ	14,914,602	32,916	500
NFCorpus	3,633	110,575	323
NQ	2,681,468	132,803	3,452
HotpotQA	5,233,329	170,000	7,405
FiQA-2018	57,638	14,166	648
Signal-1M (RT)	2,866,316	-	97
TREC-NEWS	594,977	-	57
Robust04	528,155	-	249
ArguAna	8,674	-	1406
Touché-2020	382,545	-	49
CQADupStack	457,199	-	13,145
Quora	522,931	-	10,000
DBPedia	4,635,922	-	400
SCIDOCS	25,657	-	1,000
FEVER	5,416,568	140,085	6,666
Cimate-FEVER	5,416,593	-	1,535
SciFact	5,183	920	300

Table 2.3: Collection statistics for the 18 datasets in the BEIR benchmark [Thakur et al. 2021].

Robust04 Robust04 [E. Voorhees 2004] can be considered as a rather small test collection, containing around $528k$ documents – a mix of news articles and government documents – for 249 queries focused on poorly-performing topics (examples can be found in Table 2.4). The collection contains a large number of relevance judgments – more than $1.2k$ per query – pooled from a wide range of approaches, including manual runs. Therefore, Robust04 withstood the test of time and remains one of the most used ad-hoc collections. It also still provides a reliable evaluation of neural models, despite the fact they did not participate in the evaluation campaign [E. M. Voorhees et al. 2022b]. Contrary to MS MARCO passages, Robust04 documents are quite long, requiring specific adjustments when working with Pre-trained Language Models, which are usually limited by their input length (see Section 2.6). Because of its relatively low number of queries, Robust04 is, however, not suited for training large neural ranking models.

Queries
wind power location
Winnie Mandela scandal
oceanographic vessels
Health and Computer Terminals
space station moon

Table 2.4: Sample of queries taken from the Robust04 collection.

TREC-COVID TREC-COVID [E. Voorhees et al. 2021] is a community initiative aiming to build a test collection around COVID-19 literature, which captures the information needs of biomedical researchers during a pandemic. More specifically, it is based on the CORD-19 dataset [L. L. Wang et al. 2020], an "evolving" collection capturing the growth of the COVID literature over time. The track thus includes

Queries
coronavirus remdesivir
animal models of COVID-19
coronavirus and ACE inhibitors
dexamethasone coronavirus
violence during pandemic

Table 2.5: Sample of queries taken from the TREC-COVID collection.

several rounds, and the BEIR benchmark relies on the final cumulative relevance judgments, alongside the July 16, 2020 version of the CORD-19 dataset. It thus contains $171k$ documents for 50 queries – for which examples can be found in Table 2.5.

> **Remark**
>
> As of today, MS MARCO remains the main open large-scale dataset available to train neural rankers. Afterward, models are usually zero-shot transferred to other datasets, such as BEIR, for evaluation purposes. We follow a similar rationale in this thesis – see Chapter 3 and 4.

2.4 Representing Text for Information Retrieval

In the following, we introduce techniques to *represent* text in a way that can be meaningful for IR methods and machine learning models. Representing the content of a collection is a key step for many information access tasks, such as document Classification, Clustering, and Information Retrieval. Retrieval models are therefore tied to the underlying representations of documents and queries. In the early days of IR, such representations were derived in an unsupervised manner, based on the statistical properties of natural language. The rise of machine learning applied to NLP and IR has given birth to techniques that can represent text as dense vectors, and that can be trained from large amounts of unsupervised data. Such methods can encode semantic relationships between terms, helping retrieval models to go beyond exact term matching. They span from static representations such as word2vec to the recent developments in Pre-trained Language Models.

In Section 2.4.1, we first introduce a general framework for text ranking that relies on representations of queries and documents. We then proceed to describe the various ways we can represent text in such a framework, from traditional (Section 2.4.3) to contextualized representations (Section 2.4.4). Those representations are the basis of retrieval methods introduced in Section 2.5 and Section 2.6.

2.4.1 A General Representation Framework for IR

Although based on radically different paradigms, many retrieval models can be viewed under the same common framework. Following [J. Guo et al. 2020; Tonellotto 2022], a ranking function can be abstracted as:

$$s(q, d) = f(\phi(q), \psi(d), \eta(q, d)) \tag{2.10}$$

where ϕ and ψ are the functions mapping queries and documents to their representation spaces, while η extracts features from their joint

representation – or interaction. f represents any function deriving relevance from such features[14]. In the remainder of this document, we are mostly interested in representation-based approaches that discard the interaction component η, and can be written as follows:

$$s(q, d) = f(\phi(q), \psi(d)) \tag{2.11}$$

where the representation functions – usually called encoders – map queries and documents in the same vector space, and f usually corresponds to a similarity such as dot product, i.e., $s(q, d) = \phi(q)^T \psi(d)$. In this setting, approaches only differ in the way they *represent* queries and documents, from unsupervised to supervised approaches. We usually refer to as bi-encoders, dual encoders, or Siamese networks [Bromley et al. 1993] the specific supervised case where $\phi_\theta = \psi_\theta$ (see Sections 2.6.3.1 and 2.6.4.3). Such a formulation of the ranking problem was recently brought up-to-date in the conceptual framework for a representational approach to IR introduced in [J. Lin 2022].

[14] Note that these functions can be parametrized, e.g. ϕ=ϕ_θ.

Remark

Models following Eq. 2.11 allow to pre-compute document representations offline, making them usually extremely attractive for Information Retrieval – for which efficiency is a critical aspect. In the meantime, the joint encoding η in Eq. 2.10 is query-dependent – resulting in approaches that are usually more effective due to the fine-grained interactions between queries and documents. They can, however, only be applied in a re-ranking setting (see Section 2.5.2).

2.4.2 Tokenization

At the core of any Information Retrieval model lies the notion of representation *unit*. Such units result from the tokenization process, which decomposes an input sequence of characters into atomic units – referred to as tokens. In traditional approaches, tokens correspond to actual words resulting from various processing steps like normalization[15] or filtering[16]. On the other side of the spectrum, text can be decomposed into characters or n-grams (i.e., n-words sequence). Lying in between, current Pre-trained Language Models operate on sequences that are tokenized at the *subword* level based on unsupervised algorithms like WordPiece [Y. Wu et al. 2016] or Byte-Pair Encoding [Sennrich et al. 2016]. Such approaches reduce the vocabulary space by splitting rare words into sub-units.

The set of unique terms – either words, subwords, or else – in the collection $\mathcal{C}$ constitutes the vocabulary V of *terms*. Word vocabularies are usually large according to Heaps's law [Heaps 1978] (e.g., $|V| =$

[15] Normalization includes – but not exclusive – capitalization, removing accents, and techniques such as lemmatization or stemming [Porter 1980].

[16] For instance, the most frequent words.

2.6M for MS MARCO). On the contrary, subword vocabularies are fixed and relatively small (e.g., $|V| = 30k$ for WordPiece). We show in Table 2.6 the results of a "traditional" and a WordPiece tokenization, on two Robust queries.

q="*wind power location*"
`word` → *wind*, *power*, *locat*
`subword` → *wind*, *power*, *location*
q="*oceanographic vessels*"
`word` → *oceanograph*, *vessel*
`subword` → *ocean*, *##ographic*, *vessels*

Table 2.6: Result of the tokenization for two queries from Robust04 (Table 2.4). Word-level tokenization is obtained using the default Anserini [P. Yang et al. 2017] analyzer with Porter stemming. Subword-level tokenization is obtained with HugginFace's [T. Wolf et al. 2020] WordPiece tokenizer. Note how "*oceanographic*" is decomposed into two subwords, by prepending "*##*" to suffixes.

2.4.3 Traditional Approaches

We refer to traditional approaches for representing text as the techniques that predate BERT and the rise of Pre-trained Language Models. We introduce methods to represent words and documents that can further be applied to represent queries and derive ranking models (Section 2.5).

2.4.3.1 Bag-of-Words Representations

In the Vector Space Model (VSM) [Salton et al. 1975], documents are represented as vectors in the vocabulary vector space $\mathbb{R}^{|V|}$, where each dimension corresponds to a term[17] of V. It thus becomes possible to compute similarities between pieces of text, for instance, by relying on cosine similarity or dot product:

$$sim(d_i, d_j) = d_i^T d_j \tag{2.12}$$

As word order is not taken into account, such representations are called Bag-of-Words (BoW). Documents (or queries) only contain a handful of terms, and the vocabulary is usually large[18], resulting in vectors that are extremely *sparse*, i.e., containing only a few positive entries.

So far, we haven't defined *what* contain such vectors. From a very general standpoint, we can represent a document as $d = (w(t, d_i))_{t \in V}$, where w represents a weighting function for term t in document d[19]. There is a vast literature in IR around weighting schemes that account for statistical properties of terms (Section 2.5.1.1), or that can be derived from probabilistic interpretations of the ranking problem (Section 2.5.1.2). One simple yet effective way to account for the relative

[17] In the remainder, we use term, word or token interchangeably to refer to the units that constitute the vocabulary.

[18] For instance, the MS MARCO passage collection contains 2,660,824 (unique) terms in its vocabulary, based on the processing used in Table 2.6.

[19] Here, the weighting function is the document encoder ψ from Eq. 2.10.

importance of a given term in a document consists in counting how many times it appears (term frequency, or tf):

$$d = (tf(t,d))_{t \in V} \tag{2.13}$$

However, it does not account for the importance of a term (or its specificity), that can be derived as a function of the inverse number of documents a term appears in (called document frequency, or df), as the Inverse Document Frequency (IDF) [Sparck Jones 1972]:

$$IDF(t) = \log \frac{|\mathcal{C}|}{df(t)} \tag{2.14}$$

Intuitively, terms that only appear in a few documents are very discriminative and thus should have a higher weight. The so-called tf-IDF combines both ideas, such that documents are represented as:

$$d = (tf(t,d) \times IDF(t))_{t \in V} \tag{2.15}$$

This approach has a long-standing history in Information Retrieval, and models like BM25, relying on a similar formulation, are still extremely effective (Section 2.5.1.2). Note that the VSM is not limited to IR, as it can also serve as the basis for Classification or Clustering algorithms. As discussed in Section 2.5.1, lexical approaches to IR view queries as vectors in the same space, such that scoring functions $s(q,d)$ can simply be seen as (sparse) dot products. Also note that these representations are tied to the considered vocabulary units, and can seemingly be extended n-grams or subwords.

The sparsity of BoW vectors allows efficient storing and manipulation using inverted indexes. An inverted index is a data structure that stores, for each term in the vocabulary V, the list of documents containing that term (called posting list), alongside statistics such as term frequency or position (as shown in Figure 2.2). [Zobel and Moffat 2006] provide a thorough overview of indexing techniques, and further details on the use of inverted indexes for efficient retrieval are given in Section 2.5.1.4.

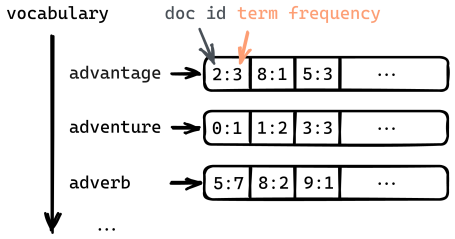


Figure 2.2: Simple illustration of an inverted index. Each term t of the vocabulary V stores the list of document *ids* containing t, alongside term statistics such as tf.

2.4.3.2 Distributed Representations

In IR, queries tend to use a different vocabulary than those used in relevant documents, leading to the so-called vocabulary mismatch problem [Furnas et al. 1987] that cannot be dealt with lexical representations[20]. Indeed, as terms correspond to independent dimensions in the vector space, it is impossible to model relationships between close or distant words in a continuous manner. For instance, the terms "*coronavirus*" and "*COVID-19*" in Table 2.5 are semantically close, but their representations – an index in $\{1, ..., |V|\}$ – are different. Similarly, semantically close documents containing different terms might have different representations.

[20] For instance, user enter queries that insufficiently cover their information need.

To alleviate this problem, different techniques have been developed to obtain word or document representations in a low-dimensional space $\mathbb{R}^d$ that encodes semantic relationships[21], and in which two word vectors will have a high *similarity* if they tend to appear together. These approaches are indeed based on the Distributional Hypothesis [Harris 1954]: *"words that occur in similar context convey similar meanings"*. This hypothesis can be exploited by leveraging large collections of text to infer or learn word representations in an unsupervised manner.

[21] And where $d \ll |V|$.

One of the earliest works to embed words in such a space is Latent Semantic Analysis (LSA) [Deerwester 1988] – which actually comes from the IR community[22]. It produces a set of latent topics by decomposing the term-document matrix of size $|V| \times |\mathcal{C}|$ – which contains for each document of the collection the tf-IDF value for terms in the vocabulary – with Singular Value Decomposition (SVD). This low-rank approximation factorizes term co-occurrences within documents of the collection. Documents, as well as terms, can thus be represented as dense vectors, where each dimension corresponds to a latent topic.

[22] Under the name Latent Semantic Indexing (LSI). Some extensions of LSA include Probabilistic LSA [Hofmann 1999] and Latent Dirichlet Allocation [Blei et al. 2003].

Various word embedding techniques have been further developed and have led to various achievements in NLP. It includes techniques like word2vec [Mikolov et al. 2013], which learns word representations from their context in a contrastive manner, or GloVe [Pennington et al. 2014], which directly factorizes the Pointwise Mutual Information matrix between words and contexts[23]. To obtain document (or query) level representations, various methods have been introduced to aggregate word embeddings, from the doc2vec extension [Le and Mikolov 2014] and BoW pooling [Arora et al. 2017], to deep learning approaches relying, for instance, on Convolutional Neural Networks (CNNs) or Long Short-Term Memories (LSTMs)[24]. As discussed in Section 2.5.4, interactions between queries and documents based on word2vec embeddings (η in Eq. 2.10) have been extensively used in neural IR models.

[23] The context is usually a fixed-sized window around the word.

[24] We don't cover such approaches here – we just inform the reader of their existence.

2.4.4 Pre-trained Language Models

Embedding methods like word2vec have powered various state-of-the-art models in NLP and IR. However, such representations are global and do not consider the context in which words appear[25]. Additionally, each downstream task usually required training a specific architecture from scratch to learn the meaningful relationships between words and/or sentences. This led to the development of generic architectures that can acquire general contextualized knowledge about language through unsupervised pre-training, and which only need to be *fine-tuned* for transfer – dubbed Pre-trained Language Models.

[25] For instance, the word "*bank*" would have the same word2vec embedding in "*the river bank*" or the "*the bank account*" – issue known as polysemy.

Current PLM rely on two main ingredients: *i*) the transformer architecture [Vaswani et al. 2017] (Section 2.4.4.1) and *ii*) self-supervised pre-training (Section 2.4.4.2). In the past two years, the rapid pace of progress in NLP has led to a large variety of models, from simple

encoders to Sequence-to-Sequence (seq2seq) models[26]. For conciseness, we give an overview of BERT (for Bidirectional Encoder Representations from Transformers, [Devlin et al. 2019]), an encoder model widely adopted in the IR community.

[26] For instance, there are more than $175k$ models available on the HuggingFace model page (April 12, 2023): https://huggingface.co/models.

2.4.4.1 Transformer Encoder

Transformers are powerful general-purpose deep learning models that rely on self-attention and constitute, as of today, the state of the art for most NLP tasks and beyond, including, for instance, Computer Vision [Dosovitskiy et al. 2021] or even Reinforcement Learning [L. Chen et al. 2021]. BERT's architecture is similar to the transformer *encoder* introduced in [Vaswani et al. 2017], whose objective is to compute contextualized representations $(h_i)_{i=1:n}$ for all the tokens in an input sequence $(x_i)_{i=1:n}$. A transformer encoder is composed of a stack of L layers based on i) a Multi-Head Self-Attention (MSA) layer, which allows each position i to attend to every position of the input and ii) a position-wise Feed-Forward Network (FFN).

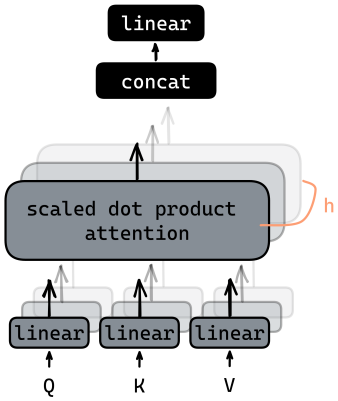


Figure 2.3: Illustration of MSA (Eq 2.18). Q, K, and V are all taken as the output of the previous layer, i.e., the representations of the sequence x at layer $l-1$. In practice, $d_k = d_v = \frac{d}{h}$.

Attention An attention mechanism can be abstracted as mapping a query and a set of key-value pairs to an output, through a weighted sum of the values, where the weight for each value is based on a similarity between the query and corresponding keys. More specifically, BERT is based on scaled dot-product attention, where queries and keys have dimension d_k, and values dimension d_v. The attention output is given by:

$$A(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.16}$$

BERT relies on *self-attention*: queries, keys, and values come from the output of the previous layer, and are linearly mapped by the parameter matrices W^Q, W^K and W^V, of size $d \times d_k$, $d \times d_k$ and $d \times d_v$ respectively[27]. To let the model attend to information from different representation subspaces at different positions, each attention layer actually relies on h attention mechanisms that operate in parallel, referred to as Multi-Head Self-Attention – shown in Figure 2.3:

$$MSA(Q,K,V) = \texttt{concat}\left(\text{head}_1||...||\text{head}_h\right)W^O \tag{2.17}$$

$$\text{head}_j = A(QW_j^Q, KW_j^K, VW_j^V) \tag{2.18}$$

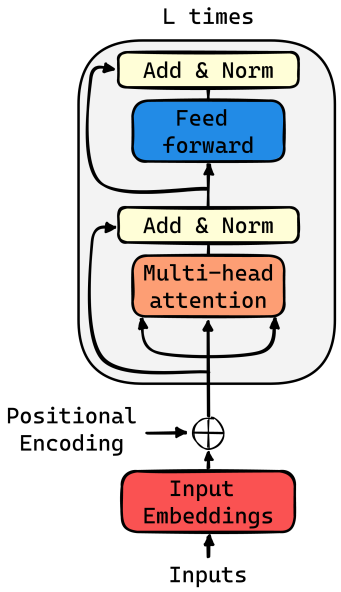


Figure 2.4: Transformer encoder.

FFN Then, a two-layer position-wise MLP is applied to each position i independently. To ease model training, a residual connection [K. He et al. 2016] followed by a layer normalization [J. L. Ba et al. 2016] are placed between each of these two sublayers.

Finally, to take into account the position of tokens in the sequence, positional embeddings are added to the input – which are learned in

[27] d denotes the "model" dimension, i.e., the dimension of embeddings at each layer. In the case of BERT (base), $d = 768$.

the case of BERT. Figure 2.4 summarizes the complete transformer architecture. Two model variants have been introduced in [Devlin et al. 2019], a "base" version and a larger one. Table 2.7 lists their main specifics, including the distilled version introduced in [Sanh et al. 2019].

model	L	A	H	#params
`bert-large-uncased`	24	16	1024	$340M$
`bert-base-uncased`	12	12	768	$110M$
`distilbert-base-uncased`	6	12	768	$66M$

Table 2.7: BERT architectures. L corresponds to the number of layers, while A denotes the number of MSA layers and H the dimension of the output embeddings at each layer. The number of heads, for each MSA, is set to $h = 8$.

2.4.4.2 Pre-Training

Contrary to similar works at that time[28], BERT is a bi-directional encoder that cannot be trained with left-to-right (or right-to-left) Language Modeling objectives. The main innovation of BERT is the Masked Language Modeling (MLM) pre-training task. Inspired by the Cloze Task [Taylor 1953], it relies on a fairly simple idea: given an input sequence, randomly *mask* some fraction of the input tokens, that the model must predict given their context. More specifically, the final representation $h_{\texttt{[MASK]}}$ of a masked token is first mapped to the vocabulary with the MLM head as follows:

$$z = E \times \text{transform}(h_{\texttt{[MASK]}}) + b \tag{2.19}$$

where E of size $|V| \times d$ denotes the BERT input token embedding matrix, b is a token-level bias, and transform(.) is a linear layer with GeLU activation [Hendrycks and Gimpel 2016] and LayerNorm. Then, a softmax is applied to obtain a distribution over the vocabulary, i.e., $p = \text{softmax}(z)$. The model is finally trained with a Cross-Entropy loss to predict the actual masked tokens, as illustrated in Figure 2.5. In practice, 15% of the input tokens are masked, i.e., replaced with a special `[MASK]` token. To mitigate the train-test mismatch, only 80% of these sampled tokens are actually masked; in 10% of the time, they are replaced by random tokens, and the 10% remaining are kept unchanged. Note that BERT has been trained with an additional Next Sentence Prediction loss, which was later shown to have a limited impact [Y. Liu et al. 2019]. We, therefore, do not detail this task. Finally, note that BERT is tokenizing input sequences with WordPiece (Section 2.4.2): besides effectively dealing with certain languages or the issue of unknown terms, subword-level tokenization is also motivated by practical considerations. As transformers include a learnable embedding matrix E, the vocabulary size has to be limited to control the number of parameters[29].

[28] For instance ULMFiT [Howard and Ruder 2018] or GPT [Radford et al. 2018].

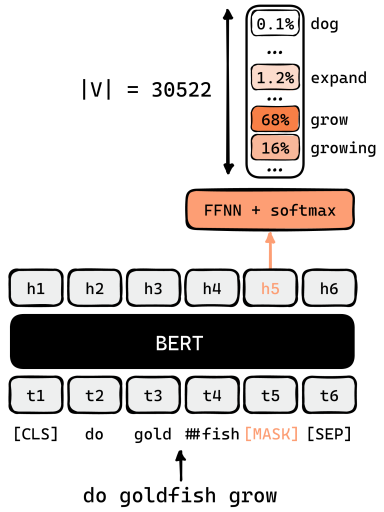


Figure 2.5: MLM prediction illustrated on a test query from TREC DL 2019. Here, the goal is to predict "*grow*" given the context. It implicitly performs *expansion*, by giving more weights to "close" words ("*growing*" in the example).

[29] For instance, the embedding matrix contains $23M$, i.e., 21% of the total number of parameters, for BERT (base).

Remark

Usually, the MLM head, whose role is to project the representations of `[MASK]` tokens to the vocabulary, is only used during pre-training and discarded afterward. We show in Chapter 3 that this layer can effectively be employed to learn query and document expansion.

2.4.4.3 Fine-Tuning

The pre-training paradigm of Pre-trained Language Models has considerably impacted how we use neural models in NLP and IR. By relying on massive amounts of unsupervised text data that provide self-supervision, models can acquire general knowledge about language. Thus, they only need to be *fine-tuned* with a small amount of labeled data on downstream tasks – thus fighting the data scarcity for most NLP tasks which only have access to a limited amount of training data[30]. The fine-tuning step requires little to no architectural changes, and generally simply relies on contextual word vectors (h_i). Additionally, a special "beginning of sentence" token is usually prepended to input sequences (dubbed `[CLS]` for "classification" in the case of BERT), that can generically be fine-tuned for classification tasks (see Section 2.6.1.1).

In the IR case, we are generally interested in learning, from relevance judgments data, a scoring model $s(q, d)$ between a query and document (see Section 2.6.5.1). While pre-training is usually costly[31], and reserved to organizations with sufficient resources, model weights are most of the time released to the community. They can serve as a starting point for almost any NLP task.

[30] Note this is not exactly the case in IR, as MS MARCO provides more than $500k$ training queries.

[31] The training of Google's PaLM-540B [Chowdhery et al. 2022] required "*6144 TPU v4 chips for 1200 hours and 3072 TPU v4 chips for 336 hours*"!

Remark

BERT provides contextualized representations for every term in the input sequences, but many tasks require sequence-level representations. For instance, in the representation-based approach to IR, we aim to map queries and documents in a common vector space such that $s(q, d)$ can be expressed as a dot product.

2.4.4.4 The Landscape of PLM

By building on previous successful contextualized embedding models such as ELMO [Peters et al. 2018] or ULMFiT [Howard and Ruder 2018], BERT has established itself as one of the most popular Pre-trained Language Models. Other encoder models include DistilBERT [Sanh et al. 2019], ERNIE [Sun et al. 2019], RoBERTa [Y. Liu et al. 2019], DeBERTa [P. He et al. 2020], or ELECTRA [Clark et al. 2020]. Because of the quadratic time/space complexity of attention[32], standard PLM

[32] As well as the presence of learned positional embeddings in models like BERT.

are inherently limited in the length of the sequences they can process. For instance, BERT can only encode inputs up to 512 tokens. Therefore, efficient transformer architectures have also been proposed, aiming to tackle long-range dependencies with dedicated attention structures relying, for instance, on sparse patterns [Beltagy et al. 2020] or hashing techniques [Kitaev et al. 2020]. Please refer to [Tay et al. 2022a] for an in-depth overview of efficient transformers.

Additionally, recent progress has also been made with encoder-decoder (or decoder-only) models, with models like BART [Lewis et al. 2020], T5 [Raffel et al. 2020] or GPT-3 [Brown et al. 2020]. Such approaches have demonstrated impressive few- and zero-shot performance, and represent promising research directions in IR[33].

[33] Models like ChatGPT (https://openai.com/blog/chatgpt/) are already thought of as the new generation of "generative search engines".

2.5 Traditional Information Retrieval Models

In Section 2.4, we have discussed various ways to represent text. In the following, we introduce the main Information Retrieval approaches based on such query and document representations. Recall that, for a given query q, the retrieval task is defined as retrieving relevant document(s) from the collection $\mathcal{C}$. More specifically, IR models assign a score $s(q, d)$ to each document in $\mathcal{C}$ – or a subset of it. Documents are usually ranked in decreasing order of $s(q, d)$, and we can evaluate the quality of ranking using evaluation metrics introduced in Section 2.2. We first describe in Section 2.5.1 the traditional approaches to IR based on lexical cues. We then briefly describe in Section 2.5.3 the Learning-to-Rank framework that formulates ranking as a supervised machine learning problem over handcrafted relevance features. We finally review in Section 2.5.4 the pre-BERT approaches to neural IR. In the following part (Section 2.6), we thoroughly describe the new wave of neural IR models based on Pre-trained Language Models.

2.5.1 Lexical Approaches

The central idea behind lexical (or term-based) approaches is that *relevant documents should contain important query words*. These approaches thus rely on the so-called exact matching (or lexical matching) of terms. While this hypothesis suffers from several limitations – the most concerning being the vocabulary mismatch problem – a large portion of the time, it holds true. Thus, keyword-based models have been – and still are – the core component of most search systems. Until very recently, it was actually quite difficult for neural rankers to outperform such simple approaches [J. Lin 2019; W. Yang et al. 2019a][34].

[34] This trend changed with PLM. However, recent works focusing on the generalization capabilities of neural rankers showcased how brittle they can be, compared to lexical models like BM25 [Thakur et al. 2021].

Regardless of the different modeling paradigms, lexical approaches to IR can all be viewed under the lens of the representation framework (Eq. 2.11). Thus, relevance $s(q, d)$ can be expressed as a *sparse* dot product between query and document vectors, and what differentiates

the approaches is the way to obtain such representations (i.e., ϕ and ψ). An alternative (but equivalent) formulation is commonly employed:

$$s(q,d) = \phi(q)^T \psi(d) = \sum_{t \in q \cap d} w^{(q)}(t,q) w^{(d)}(t,d) \tag{2.20}$$

where $w^{(q)}(t,q)$ and $w^{(d)}(t,q)$ represent the weights for term t in the query and document respectively.

2.5.1.1 Scoring in the Vector Space Model

In Section 2.4.3.1, we have discussed the notion of documents represented in the vocabulary vector space. The VSM scoring approach is simply based on representing queries as vectors in the same space, such that similarity measures can be used to score documents. From a general standpoint, we usually express the relevance score as a dot product: $s(q,d) = \phi(q)^T \psi(d)$[35]. If ϕ is chosen as a binary indicator of the presence of the term in the query (i.e., a one-hot encoding of the query), and ψ as the document tf-IDF weighting introduced in Section 2.4.3.1, we come up with the standard tf-IDF scoring [Salton et al. 1975; Salton and C. S. Yang 1973]:

$$\text{tf-IDF}(q,d) = \sum_{t \in q \cap d} IDF(t) \times tf(t,d) \tag{2.21}$$

Many alternatives have been developed to assign proper weights to terms in the VSM framework, including various scaling or normalization components[36].

[35] It is standard to represent similarities in the VSM framework as cosine similarity to account for the varying length of documents and queries. However, such normalization can be included in the representation itself, and we more generally consider dot product.

[36] Some of such alternatives can be found in [Manning et al. 2008], Section 6.4.3.

2.5.1.2 Probabilistic Approaches to IR

The probabilistic approaches to IR give a probabilistic interpretation of relevance by reasoning under uncertainty[37]. They are based on the Probability Ranking Principle [S. E. Robertson 1997], which states that documents should be ranked in decreasing order of their estimated probability of being relevant to the query $\mathcal{P}(R = 1|d,q)$.

While these approaches conceptually differ from the VSM, they still aim to find appropriate term weights, which are, in this case, derived from probabilistic theory. Thus, even though there is no explicit notion of *representation*, the relevance score $s(q,d)$ can still be written as in Eq. 2.11. One such model building on term frequency and document length statistics is BM25 [S. Robertson and Zaragoza 2009; Stephen E. Robertson et al. 1994]. It is derived from a generative model of documents, where term frequencies are assumed to follow a 2-Poisson distribution – it is expressed as:

$$BM25(q,d) = \sum_{t \in q \cap d} \text{IDF}(t) \left(\frac{tf(t,d)(k_1+1)}{tf(t,d) + k_1(1 - b + b\frac{|d|}{L})} \right) \tag{2.22}$$

[37] For instance, user queries only poorly reflect the true underlying information need.

where k_1 and b are tunable parameters that can be adjusted given appropriate relevance annotation[38]. As of today, BM25 remains one of the most standard and effective baselines, and is still widely used as a comparison point in the neural IR literature.

[38] They are usually set to $k_1 \in [1.2, 2]$ and $b = 0.75$.

Building on a different perspective, Language Modeling approaches to IR [Ponte and Croft 1998] view retrieval as a generative process: the relevance of a document to a query is tied to how it is actually possible to *generate* the query from the document[39]. More specifically, the goal is not to explicitly model $\mathcal{P}(R = 1|d, q)$, but rather build a probabilistic Language Model for each document that can be used to rank documents based on query likelihood:

[39] Think about the mental process of writing a query: users tend to select words that might appear in relevant documents.

$$\mathcal{P}(d|q) = \frac{\mathcal{P}(q|d)\mathcal{P}(d)}{\mathcal{P}(q)}$$

We thus only need to estimate $\mathcal{P}(q|d)$, by estimating a Language Model for each document in $\mathcal{C}$, for instance, relying on Maximum Likelihood Estimation [Fisher 1922]. Other probabilistic models include informational approaches such as the Divergence From Randomness framework [Amati and Van Rijsbergen 2002], which aims to infer term weights by measuring the divergence between a term distribution produced by a random process (within the collection) and the actual term distribution (within the document).

2.5.1.3 Fighting the Vocabulary Mismatch

Lexical approaches are inherently tied to exactly matching query terms – or their normalized forms – in documents, and thus suffer from the vocabulary mismatch, where different words may refer to the same concept. Various methods have been developed to tackle such challenges, aiming to bridge the vocabulary gap. It includes *i*) expansion, *ii*) translation models and *iii*) models based on latent spaces.

Query expansion resorts to producing "new" refined queries that better capture the various aspects and formulations of the information need[40]. Global expansion can, for instance, rely on word relationships, such as synonymy, to enrich query representations [E. M. Voorhees 1994]. The most popular techniques, however, involve relevance feedback. It relies on the hypothesis that if we have access to a subset of relevant and non-relevant documents for a given query q, we can better estimate term weights in traditional probabilistic models. In practice, an initial retrieval step is performed, and relevant documents are used to identify terms – with appropriate weights – that can be used to form a newly expanded query, that is further fed to the retrieval system. The Rocchio algorithm [Rocchio 1971] implements such a mechanism in the VSM, relying on relevant documents obtained through user feedback. As requiring user interaction is problematic, automatic methods have further been developed, assuming top documents from the system point

[40] We also inform the reader that document expansion techniques exist, with a similar goal. They have, however, been far less popular in IR until the recent development of sparse methods based on PLM.

of view to be relevant – leading to the so-called Pseudo-Relevance Feedback (PRF) models [Lavrenko and Croft 2001]. Query expansion based on PRF is a popular technique in the IR community[41], and generally works on average; however, it can suffer from query drift, hurting the retrieval quality for some queries.

[41] The RM3 model [Jaleel et al. 2004] is one the most used expansion baseline to compare to neural IR models.

> **Remark**
>
> Document and query expansion are actually part of the PLM-based IR toolbox. More particularly, it constitutes the central idea of our main contribution (see Chapter 3).

Aside from (P)RF, other techniques also aim to bridge the vocabulary gap. [Berger and Lafferty 1999] depart from traditional expansion techniques and propose a new probabilistic approach based on the translation of documents into queries, allowing to take into account "soft" matching signals in the form of translation probabilities learned with a variant of the EM algorithm [Dempster et al. 1977] from a synthetic set of paired queries and documents[42]. LSI [Deerwester 1988] builds on the Singular Value Decomposition of the term-document matrix to represent documents in the latent topic space. Similarly to the VSM, we can project queries into the same space and derive relevance from similarity measures[43]. LSI however tends to be less effective compared to standard models like BM25 [Atreya and Elkan 2011].

[42] [Boytsov and Kolter 2021] adapted with success such model with BERT.

[43] In the case of LSI, we can think of ϕ and ψ of Eq. 2.11 as *unsupervised* encoders that map queries and documents into a dense latent space. It is thus conceptually similar to dense approaches based on PLM (Section 2.6.3) – although no learning is involved here.

2.5.1.4 Scoring with an Inverted Index

As already mentioned in Section 2.4.3.1, BoW approaches can be efficiently handled by an inverted index, which stores posting lists for each term of the vocabulary V. This allows for fast query inference, as it is only required to traverse the posting lists corresponding to the actual query terms (as illustrated in Figure 2.6) – instead of considering the whole collection $\mathcal{C}$. Techniques like *document-at-a-time* can be used, provided that the posting lists are sorted according to document *ids* [Tonellotto et al. 2018; Zobel and Moffat 2006]. Many efficient query processing algorithms such as MaxScore [Turtle and Flood 1995] or WAND [Broder et al. 2003] have further been developed, to additionally skip documents – by quickly identifying those that cannot possibly be in the top-k – and improve efficiency.

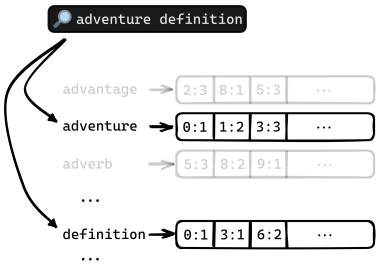


Figure 2.6: Scoring with an inverted index – a high-level overview. Only the documents containing query terms (here, "*adventure*" and "*definition*") must be scored.

2.5.2 The Multi-Stage Ranking Pipeline

We have described lexical approaches which, based on efficient inverted indexes and query processing algorithms, can perform retrieval on large collections of documents. Before introducing more complex ranking models relying on machine learning, we first present the multi-stage ranking pipeline that has become standard practice in modern search

Remark

Dense models based on PLM combined with approximate scoring methods (see Section 2.6.3) hold many promises for future IR systems. However, inverted indexes have been providing search functionalities in distributed settings at Web-scale for decades. They are optimized in various aspects such as space or latency – making sparse methods based on PLM particularly appealing (see Section 2.6.4 and Chapter 3).

systems. This *retrieve-rerank* paradigm is a well-studied setting that addresses the practical limitations faced when applying computationally intensive models for the task. It is indeed obviously impossible to perform online inference for every document in $\mathcal{C}$, and this for every query, especially when the input of ranking models is dependent on the query (η in Eq. 2.10).

Multi-step pipelines decompose the ranking process into two stages: *i*) an efficient model such as BM25 performs first-stage retrieval (or candidate generation), to select a pool of candidate documents – for instance, top-1000, *ii*) one (or several) models are further applied to re-rank this subset of documents. The overall idea is illustrated in Figure 2.7. Because of the scale of the problem, the retrieval step is based on models that can *pre-compute* document representations.

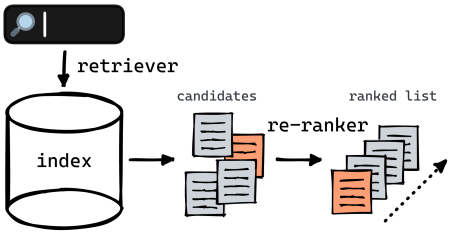


Figure 2.7: The multi-stage ranking pipeline in IR. We depict here a simple two-stage scenario, but multiple intermediate steps can be part of the pipeline. Here, the re-ranker puts the relevant document (in orange) on top.

Methods used for the first stage should have a high Recall, while re-ranking models must be oriented towards Precision. The two are somewhat complementary, as gains in Recall might lead to gains in early Precision. It also perfectly illustrates the efficiency-effectiveness[44] trade-off faced by search systems: it is usually possible to improve the first stage of a ranking pipeline, by sacrificing efficiency. Such aspects are thoroughly discussed in Chapter 3 (Section 3.5.4).

[44] We use effectiveness to refer to model performance in terms of IR metrics, while efficiency refers to measures such as query latency.

Remark

While deferred to BoW approaches for a long time, the first stage of ranking pipelines has recently been the focus of many works based on Pre-trained Language Models (see Section 2.6.2). It also constitutes the central task in Chapter 3.

2.5.3 Learning-to-Rank

In Section 2.5.1, we have introduced several unsupervised retrieval models. Learning-to-Rank (LTR) is a branch of Information Retrieval that applies machine learning to the task [T.-Y. Liu 2011]. Such models, from decision trees to neural networks, are based on carefully designed hand-crafted relevance features, which usually include statistical

text features that either characterize queries or documents[45] (term frequency, length, etc.), or their interactions (such as the various relevance scores previously introduced) [Qin et al. 2016]. The massive adoption of search engines has also contributed to the popularity of Learning-to-Rank for two reasons: *i*) user behavior data can provide additional relevance features; *ii*) large click logs can be used as a source of cheap supervision to train models [Joachims 2002].

In LTR, query-document pairs are represented as vectors of features $x \in \mathbb{R}^d$, with d usually large, i.e., up to a few hundred. Models follow Eq. 2.10, by taking into account query- and document-level features (ϕ and ψ respectively) as well as interaction features (η). There are three main families of approaches, which are not differentiated by the models they build upon, but rather the type of loss that is optimized: *i*) pointwise methods convert the problem into classification or regression, *ii*) pairwise methods learn preferences from pairs of documents, *iii*) listwise methods seek to directly optimize ranking metrics, usually by deriving a differentiable approximation of the true loss. Popular techniques include RankSVM [Joachims 2002], RankNet [C. Burges et al. 2005] or LambdaMART [C. J. C. Burges 2010].

[45] Note that documents can actually be composed of several fields, such as title, anchor text, etc., for which features can be computed independently. Also, note that for Web documents, features like PageRank [Page et al. 1999] can be taken into account.

Remark

Learning-to-Rank strictly refers to machine learning models based on hand-crafted relevance features, even though neural IR models are also *learning how to rank*. We, however, still employ the denomination of the three types of LTR techniques for neural IR models when referring to training losses.

2.5.4 Neural Information Retrieval

Motivated by the success of Deep Learning in Computer Vision and Natural Language Processing, the Information Retrieval community started to develop early 2010s neural IR models that can better address the vocabulary mismatch – which cannot be solved with LTR techniques. Contrary to the latter, neural IR models directly operate on raw text to infer relevance between queries and documents, alleviating the need for hand-crafted features – which are known to be difficult and time-consuming to obtain.

Neural IR therefore builds on techniques to *represent* text introduced in Section 2.4, as well as Deep Learning models trained on relevance judgments. The specificities of the IR task – such as modeling *relevance* or matching short queries to long documents – made the progress of neural rankers far less evident when compared to other NLP tasks, thus driving the design of dedicated architectures and techniques. For instance, because of the central role of lexical matching in relevance

estimation, various models have explicitly encoded this mechanism into their architecture.

We had to wait for the "transformer revolution" to witness agnostic architectures that are able to directly learn such behavior from the data. Neural IR is indeed split into two distinct time-frames: the pre-BERT era[46], and the period that followed the release of PLM like BERT. In this Section, we briefly review the central ideas behind the first wave of neural IR models[47]. Please refer to [J. Guo et al. 2020; Mitra and Craswell 2018; Onal et al. 2018] for more in-depth overviews.

[Clinchant and Perronnin 2013] were the first to use word embeddings in IR, relying on LSA and Fisher Vectors [Perronnin and Dance 2007] to represent queries and documents in the embedding space, such that they can be compared with cosine similarity. In a seminal work, [Huang et al. 2013] introduce DSSM, which similarly learns, from click data, dense low-dimensional representations for queries and document titles using MLPs. The advent of neural IR models was, for the most part, due to the wider adoption of word embedding techniques like word2vec, which provided off-the-shelf term representations encoding semantics in the embedding space. For instance, [Mitra et al. 2016; Nalisnick et al. 2016] propose the Dual Embedding Space Model (DESM) that relies on pre-trained word2vec to represent and match query terms to document representations. SNRM [Zamani et al. 2018] follows a radically different approach by learning high-dimensional sparse representations[48] for queries and documents that can directly be stored in an inverted index to perform efficient retrieval. Many other works followed this central idea of learning vector representations for queries and documents – the so-called *representation-based* approach in IR (Figure 2.8), introduced in Eq. 2.11. While appealing for various reasons – especially the fact that they can be used quite efficiently by pre-computing document vectors – representation-based techniques struggle to model precise relevance signals such as lexical match.

[46] Up to 2018/2019, BERT being released in October 2018.

[47] Models based on PLM are extensively developed in Section 2.6.

[48] In such a space, each dimension corresponds to a latent term. Representations are made sparse through ℓ_1 regularization.

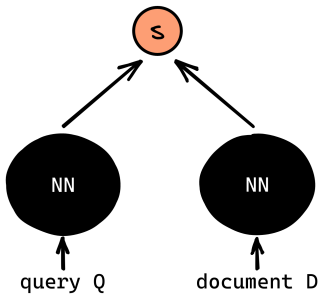


Figure 2.8: Representation-based approaches in IR. Queries and documents are represented as vectors, and the relevance $s(q, d)$ is given by a similarity measure in the embedding space.

Remark

SNRM was one of the first works to conceptually depart from concurrent approaches at that time, by *explicitly* trying to address the first-stage ranking task in multi-step pipelines – with the goal of replacing lexical approaches with a learned neural retriever. [Gillick et al. 2018; L. Wu et al. 2018] also addressed this, but in a dense setting. While having a limited impact at that time, it opened the path for this research direction, which has flourished in the transformer era (Section 2.6.2).

To capture fine-grained relevance patterns, approaches have shifted from representing queries and documents as single vectors to directly modeling their word-level interactions – the so-called *interaction-based* approaches (Eq. 2.10, Figure 2.9). In most cases, these models oper-

ate on top of an interaction matrix M, which collects query-document term-level similarities – generally, cosine similarities based on word2vec embeddings. Each row (resp. column) corresponds to a query (resp. document) token as follows:

$$M = sim(q_i, d_j) = \cos(w(q_i), w(d_j)) \tag{2.23}$$

This matrix serves as the input of interaction models, which are differentiated in the way they aggregate such term-level similarities into a relevance score $s(q, d)$. Note that, because of this joint encoding, these models can only be applied in a re-ranking setting – similarly to LTR approaches – where an initial ranker such as BM25 would retrieve the top candidates for a given query. Models like (Co-)PACRR [Hui et al. 2017, 2018] or DUET [Mitra and Craswell 2019; Mitra et al. 2017] build upon CNNs to capture position information (e.g., bi-gram matching). DRMM [J. Guo et al. 2016] was the first model to significantly outperform traditional IR systems by relying on row-level histogram pooling to summarize uni-gram matching signals, allowing to explicitly model different IR properties such as exact matching. Building on this success, K-NRM [Dai et al. 2018; C. Xiong et al. 2017] replaces the non-differentiable histograms with Gaussian kernels. It thus becomes possible to backpropagate until the embedding layer, which can be learned in an end-to-end manner, providing new word associations that are learned from click data[49]. Despite all the promises brought by Deep Learning, pre-BERT neural IR models have, however, suffered from limited success [J. Lin 2019; W. Yang et al. 2019a].

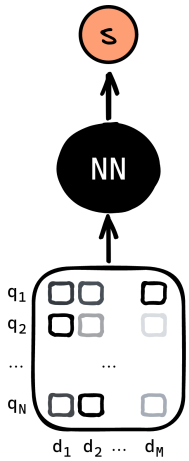


Figure 2.9: Interaction-based approaches in IR. The interaction matrix represents η in Eq. 2.10.

[49] K-NRM also constituted a major milestone, as it showed how pre-trained embeddings like word2vec – which are not suited for IR [Zamani and Croft 2017] – could be adapted for the task in an end-to-end manner.

Remark

While these models were designed before the new wave of transformer-based ranking architectures, the distinction between *interaction* and *representation* approaches still holds in the BERT era (see Section 2.6).

2.6 Transformers for Neural Information Retrieval

The release of large Pre-trained Language Models like BERT has shaken up Natural Language Processing, and later on Information Retrieval. These models have shown a strong ability to adapt to the retrieval task by simple fine-tuning. Early 2019, [R. F. Nogueira and Cho 2019] completely shifted the state of the art on the MS MARCO passage ranking task, paving the way for a new generation of PLM-based neural ranking models.

These models were initially considered as re-rankers in a standard multi-stage ranking pipeline, where candidate generation is conducted

with lexical models like BM25[50]. While BoW approaches remain strong baselines, they suffer from the long-standing vocabulary mismatch problem[51]. Re-ranking approaches are therefore inherently limited by this *retrieve-rerank* paradigm.

Thus, motivated by the large improvements brought by PLM to re-ranking, a new research direction emerged in the IR community, aiming to directly tackle *retrieval* and substitute proven lexical approaches by learned rankers based on transformers. Such a task requires coping with the strict efficiency requirements of modern search engines (extremely large collections, up to thousands of queries per second, etc.), for which lexical approaches combined with optimized inverted indexes and query processing techniques have withstood the test of time. It thus requires adapting ranking architectures to retrieve from large corpora in sublinear time.

We first describe effective (but not efficient) re-ranking models in Section 2.6.1, before covering recent works that target first-stage retrieval in Section 2.6.2. Please refer to [J. Guo et al. 2022; J. Lin et al. 2020; Tonellotto 2022] for additional resources on PLM-based Information Retrieval.

[50] That is, the usual setting introduced in Section 2.5.2 for LTR and (pre-BERT) neural IR models.

[51] Despite various attempts to bridge this gap such as query expansion introduced in Section 2.5.1.3.

Remark

Despite the variety of available Pre-trained Language Models, BERT has been widely adopted by the IR community and is the basis of recent developments in this field. Most of the models we introduce in the following are thus based on BERT (or DistilBERT), but similar results could be obtained with other approaches. Note that seq2seq models currently represent exciting research directions in Information Retrieval.

2.6.1 Re-Ranking

Section 2.6.1.1 presents re-ranking models based on encoder-only models such as BERT. In Section 2.6.1.2, we additionally introduce approaches based on Sequence-to-Sequence models.

2.6.1.1 Encoder Models

Similarly to other NLP classification tasks, [R. F. Nogueira and Cho 2019] first adapted BERT for passage retrieval by converting the ranking task into a binary classification problem based on the `[CLS]` token. We describe in the following this simple idea, that has set the stage for many subsequent works. First, each query-document pair (q, d) is concatenated, using the special `[SEP]` token, and the input is constructed as follows:

$$\texttt{[CLS]}, q_1, ..., q_N, \texttt{[SEP]}, d_1, ..., d_M, \texttt{[SEP]} \tag{2.24}$$

where [CLS] corresponds to BERT's special classification token. This input is fed to the model, and the output [CLS] representation goes through a linear layer $(W, b) \in (\mathbb{R}^d, \mathbb{R})$ followed by a sigmoid, mapping it to a relevance probability used to rank documents:

$$s_i = \mathcal{P}(R = 1|d_i, q) = \sigma(W h_{\texttt{[CLS]}} + b) \tag{2.25}$$

The model is finally trained in a pointwise fashion, using binary Cross-Entropy:

$$\mathcal{L} = -\sum_{j \in J_{\text{pos}}} \log(s_j) - \sum_{j \in J_{\text{neg}}} \log(1 - s_j) \tag{2.26}$$

where J_{pos} is the set of indices of relevant passages, and J_{neg} the set of indices of non-relevant passages in the top-k documents returned by BM25. This approach – illustrated in Figure 2.10 – is commonly referred to as *cross-encoder*[52], and is reminiscent of the interaction based models introduced in Section 2.5.4. Indeed, as queries and documents are jointly encoded, each query token can attend to each document token (and vice-versa), in an all-to-all interaction fashion. The final [CLS] representation can thus be seen as the joint representation of (q, d), which encodes (contextualized) fine-grained relevance information.

Various works have extended this idea. DuoBERT [R. Nogueira et al. 2019a] further improves re-ranking effectiveness by considering pairwise passage interactions, in a multi-stage ranking scenario. [L. Gao et al. 2020; MacAvaney et al. 2020a] address efficiency aspects of the approach by delaying query-document interactions, to pre-compute (parts of) the document representations. [MacAvaney et al. 2020b] additionally propose a representation-based approach that performs passage expansion and term-weighting, enabling fully pre-computing document representations prior to retrieval. Steering away from the trend of pre-trained models, [Hofstätter et al. 2020b,c] introduce the transformer kernels, which are lightweight models specifically designed for re-ranking, and that can be trained from scratch.

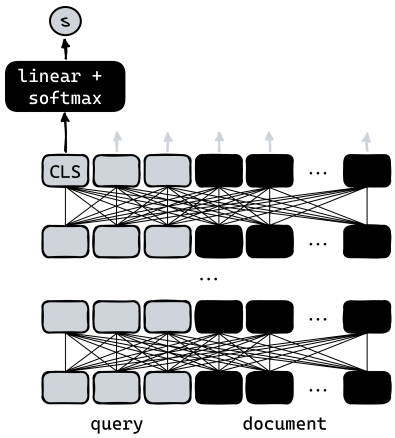


Figure 2.10: Cross-encoder model. The PLM is initialized with BERT, and the classification head is randomly initialized. The whole network is updated during fine-tuning.

[52] Even though we can find in the literature other names like monoBERT [R. F. Nogueira and Cho 2019] or vanilla BERT [MacAvaney et al. 2019a].

From Passages to Documents So far, one caveat has been omitted: [R. F. Nogueira and Cho 2019] tackled the *passage* ranking task of MS MARCO, where input passages are rather short. However, ad-hoc retrieval is usually concerned with (re-)ranking full *documents*, which can be much longer[53]. Due to the quadratic time/space complexity of attention w.r.t. input size, there is a maximum sequence length that can be processed by models[54]. To overcome this limitation, a variety of approaches have adapted BERT to the document ranking task. BERT-MaxP [Dai and Callan 2019b] simply splits documents into (overlapping) passages, and estimates the relevance of each passage independently, before resorting to max-score pooling. Birch [Akkalyoncu Yilmaz et al. 2019a,b; W. Yang et al. 2019b] operates in a similar manner,

[53] For instance, Robust04 documents contain around 540 words in average *vs* 56 for MS MARCO passages.

[54] BERT furthermore relies on learned position embeddings, inherently limiting its sequence length to $\max_l = 512$.

by combining top-k scores with cross-validated weights. [MacAvaney et al. 2019a] concurrently introduce CEDR, which incorporates contextualized embeddings into pre-BERT neural architectures like DRMM. Input documents are also split into chunks that are independently processed by BERT, before explicitly modeling term interactions. Contrary to the above approaches, CEDR directly models document-level relevance, making it fully differentiable for the task. Inspired by this idea, PARADE [C. Li et al. 2020] further simplifies CEDR, by proposing to learn passage-level representations – contrary to passage-level scores in MaxP – that are further combined with either max, attention or transformer-based pooling. The result is finally mapped to a ranking score with an MLP.

Moving away from adapting constrained architectures like BERT, other approaches propose to directly use the *X-formers* architectures tailored to improve computational and memory aspects of traditional transformers, which in turn allows dealing with longer sequences (Section 2.4.4.4). [Boytsov et al. 2022] carry a thorough evaluation of two specialized transformer architectures against previous approaches such as PARADE, highlighting the limitations of current datasets to accurately benchmark long-document models.

> ## Remark
>
> We have introduced various methods to deal with long-document *re-ranking*. Such approaches could, in theory, also be applied to long-document *retrieval*. As of today, they remain largely unexplored. We would also like to emphasize that, as most works on neural IR, throughout this thesis, we are mostly interested in passage ranking, so we put aside such limitations. However, we sometimes have to deal with long documents (for instance, when evaluating models in a zero-shot setting on the BEIR benchmark, see Chapter 3). In such cases, we rely on the surprisingly effective FirstP baseline [Boytsov et al. 2022; Dai and Callan 2019b], which is based on the hypothesis that the first paragraph of a document faithfully represents its content.

2.6.1.2 Encoder-Decoder Models

Departing from the encoder-only paradigm, subsequent works have explored ways to adapt encoder-decoder (or seq2seq) architectures for the ranking problem. For these models, every task is converted into a text generation task, by appropriately creating templates (or prompts), that explicitly tell the model *what to do*. [R. Nogueira et al. 2020] first proposed monoT5, which adapts the T5 model [Raffel et al. 2020] as a re-ranker. The input text sequence is created as follows (replacing q

and d by their actual content):

$$\texttt{Query:}q\ \texttt{Document:}d\ \texttt{Relevant:} \tag{2.27}$$

and the model is fine-tuned to produce (i.e., generate) the word "*true*" or "*false*" depending on whether the document is relevant or not. At inference time, a softmax is applied on the logits of these two tokens – at the first decoding step – and the documents are re-ranked according to the probability of the "*true*" token. [Pradeep et al. 2021] propose duoT5, which extends the approach to the pairwise setting, while [Nogueira dos Santos et al. 2020; Sachan et al. 2022a] draw the connection between seq2seq generation and query likelihood models, by considering the likelihood of a query generation conditioned on the passage $\mathcal{P}(q|d)$[55]. [Muennighoff 2022] similarly proposes to use GPT-3 [Brown et al. 2020] as a cross-encoder, without any fine-tuning. Recently, [H. Zhuang et al. 2022] propose a method to fine-tune seq2seq models with pairwise or listwise ranking losses.

[55] [S. Zhuang and Zuccon 2021c] made the same type of connection for BERT.

> **Remark**
>
> As of today, cross-encoders based on encoder-only like BERT or seq2seq models like T5 constitute the state of the art in document re-ranking. They outperform pre-BERT approaches by a large margin, at the expense of higher inference costs. They also demonstrate a strong ability to generalize, when transferring in a zero-shot setting [G. M. Rosa et al. 2022; Thakur et al. 2021].

2.6.2 From Re-Ranking to Retrieval

Re-ranking models discussed in Section 2.6.1 can only be used to score a handful of documents – typically no more than 1000. Indeed, the joint encoding of queries and documents – through concatenation – prevents pre-computing document representations, making it impossible to score *each* document in the collection given a query. [L. Gao et al. 2020; MacAvaney et al. 2020a] propose to postpone query-document interactions in BERT, but only partially address the issue, and merely derive more efficient re-rankers.

Thus, such approaches are usually bound to re-order top documents retrieved by an efficient retriever in a multi-stage ranking scenario (Figure 2.7, Section 2.5.2) – where lexical approaches like BM25 are used for the first stage[56]. Despite their high effectiveness, interaction-focused models are tied to the performance of the retriever (its Recall), which remains a bottleneck of the whole pipeline. Indeed, lexical models are known to suffer from the vocabulary gap problem, making it difficult to retrieve documents that do not contain some important query terms like synonyms.

[56] Please note that Learning-to-Rank, as well as pre-BERT neural IR models obviously face the same issue.

Building on this observation – and motivated by the large improvements brought by PLM to re-ranking overall – the IR community has started to develop *efficient* neural models based on PLM to replace traditional lexical approaches. The notion of *efficiency* is key here, as large-scale IR systems need to operate on extremely large collections of items, and might serve thousands of queries per second[57]. Term-based approaches have been relying on decades of work around optimized inverted indexes and query processing techniques [Tonellotto et al. 2018; Zobel and Moffat 2006], and still remain the backbone of modern-day search engines. In a seminal work preceding Pre-trained Language Models, [Boytsov et al. 2016] propose to replace inverted indexes by approximate k-Nearest Neighbors algorithms for the ranking task, setting the stage for new retrieval pipelines based on dense vector embeddings[58].

[57] Think about any Web search engine with millions of users per day.

Models belonging to this new generation of learned *retrievers* follow the representation-based paradigm introduced in Section 2.4.1 – Eq. 2.11. In such a setting, queries and documents are represented *independently*, allowing computing in an offline manner document representations, that can further be stored in a dedicated index structure. At inference time, only one model forward pass on the (usually short) query is required, and the search process further relies on optimized search structures – which depend on the type of representations considered, i.e., *dense* or *sparse*.

[58] While the idea seemed appealing, we had to wait until BERT to witness such approaches flourishing in the IR community.

In the first case, queries and documents are represented as dense – or continuous – low-dimensional vectors, and retrieval relies on efficient Approximate Nearest Neighbors (ANN) algorithms. In the latter case, traditional lexical approaches are brought up-to-date in various manners, and the overall approach can be seen as learning sparse representations that can further rely on the proven efficiency of traditional inverted indexes. We describe in Section 2.6.3 the general framework for dense representation learning in IR and briefly introduce the dedicated search techniques in Section 2.6.3.2. In Section 2.6.4, we introduce the various methods grouped under the "sparse" umbrella, which can be seen as methods aiming to – implicitly or explicitly – learn sparse representations.

To motivate our statements, and illustrate the large gap between traditional methods like BM25 and learned retrievers, we show in Table 2.8 the results on MS MARCO of a simple dense bi-encoder (Section 2.6.3.1) which *significantly* outperforms BM25 (↑+12.8 MRR@10).

2.6.3 Dense Approaches

In this Section, we describe the basic design of dense bi-encoders based on PLM, as well as the dedicated indexes and algorithms used to efficiently retrieve dense vectors. We finally introduce the "multiple-representations" approaches, culminating with the ColBERT model [Khattab and Zaharia 2020].

Table 2.8: Comparing BM25 and a simple dense bi-encoder on MS MARCO passage retrieval (dev set).

model	MS MARCO dev	
	MRR@10	R@1000
BM25	18.4	85.3
Dense bi-encoder	31.2	94.1

Remark

First-stage approaches are first thought of as drop-in replacements of term-based models in standard multi-stage pipelines to improve Recall. However, we might also imagine that, if they perform well enough, we could remove the need for subsequent re-rankers. That's why we do not consider re-rankers in this thesis, and solely focus on first-stage ranking. While dense approaches have received increasing attention in both IR and NLP communities, we are, however, mostly interested in sparse methods (see Chapter 3).

2.6.3.1 Dense Bi-encoders

The overall design of dense bi-encoders strictly follows Eq. 2.11, where we seek to obtain independent query and document latent representations that are used to compute similarity with dot product or cosine similarity. In the dense case, representations belong to a continuous low-dimensional space $\mathbb{R}^d$, where d is small (e.g., $d < 1000$)[59]. While this idea predates BERT with models like DSSM [Huang et al. 2013], the appearance of PLM as the backbone of such approaches, combined with more advanced LTR techniques, filled the gap of previous attempts which lacked representation power.

Sentence-BERT [Reimers and Gurevych 2019] is one the first works introducing dense bi-encoders based on BERT[60], applied to sentence similarity tasks. It provides a good overview of their basic design. Given the BERT output token embeddings $(h)_i = [h_{\texttt{[CLS]}}, h_1, ..., h_{\texttt{[SEP]}}]$, we obtain a single vector representation for the input text (query or document) by means of a simple pooling strategy – either using $f = h_{\texttt{[CLS]}}$, or by relying on the average $f = \frac{1}{N}\sum_i h_i$. In practice, we can use different encoders for queries and documents (e.g., [Karpukhin et al. 2020]); however, it has become standard to use a single encoder for both sides, as it was shown to provide additional robustness [Izacard et al. 2022; Reimers and Gurevych 2019; L. Xiong et al. 2021]. Thus, in this simplified setting, given the PLM encoder f_θ (for instance based on BERT), and the input query q and document d, the relevance score is given by:

$$s(q, d) = f_\theta(q)^T f_\theta(d) \tag{2.28}$$

[59] As opposed to the *sparse* approaches which represent queries and documents in large vocabulary-sized spaces $\mathbb{R}^{|V|}, |V| >> 10k$, with a sparse structure – see Section 2.6.4.

[60] Although many concurrent works appeared in a short period of time between 2019 and 2020. Please refer to [J. Lin et al. 2020], Section 5.4, for a thorough discussion on the historical development of BERT-based bi-encoders.

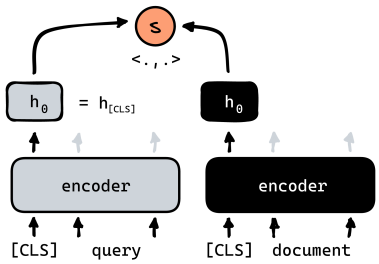


Figure 2.11: Overview of the dense bi-encoder. Here, representations are obtained with [CLS].

where representations can be normalized or not, depending on the similarity we want to use (dot product or cosine). Figure 2.11 illustrates the overall process. Building on Sentence-BERT, many subsequent works adapted bi-encoders for the ranking task. Usually, models follow the same structural design but rely on different training strategies like DPR [Karpukhin et al. 2020] or ANCE [L. Xiong et al. 2021]. The main training strategies include *i*) distillation, *ii*) hard-negative mining and *iii*) pre-training, and are further detailed in Section 2.6.5.1. In Table 2.9 we list several dense models alongside their main characteristics.

model	D	NS	PT
DPR [Karpukhin et al. 2020]	✖	✖	✖
ANCE [L. Xiong et al. 2021]	✖	✔	✖
TAS-B [Hofstätter et al. 2021]	✔	✖	✖
RocketQA [Qu et al. 2021]	✔	✔	✖
RocketQAv2 [Ren et al. 2021b]	✔	✔	✖
TCT-ColBERT [S.-C. Lin et al. 2020]	✔	✖	✖
TCT-ColBERTv2 [S.-C. Lin et al. 2021]	✔	✔	✖
CoCondenser [L. Gao and Callan 2022]	✖	✔	✔
Contriever [Izacard et al. 2022]	✖	✖	✔
AR2 [H. Zhang et al. 2022]	✖	✔	✔
ColBERT [Khattab and Zaharia 2020]	✖	✖	✖
ColBERTv2 [Santhanam et al. 2022b]	✔	✔	✖

Table 2.9: Small cartography of dense approaches. `D`, `NS` and `PT` refer respectively to Distillation, (Hard-)Negative sampling, and Pre-training. Many more approaches have been proposed in the last two years – but overall rely on the same building blocks. Also, note that each model relies on additional specifics and implementation details.

Remark

The bi- (or dual-) encoder terminology is not restricted to dense approaches per se. In Chapter 3, we introduce our main contribution SPLADE, which bridges the gap between dense and sparse approaches – being the first end-to-end sparse bi-encoder model.

2.6.3.2 ANN Search

Retrieving from large collections of dense vectors requires dedicated indexing structures and efficient similarity search algorithms. Many of such ideas have been developed by the Computer Vision (CV) community, motivated by the need to search over large databases of image representations. With the advent of dense retrieval based on PLM for text retrieval, the IR community has also recently turned its attention

to such techniques. In the dense setting, retrieval is formulated as a nearest neighbors search problem, which consists in finding the top-k most similar documents to the query, based on the similarity defined by the model – usually dot product in the case of dense bi-encoders. This is the exact same problem we aim to solve with inverted indexes storing sparse BoW representations – but the objects of interest are very different in nature, and thus benefit from specific techniques. Overall, we need *i*) a search algorithm, *ii*) a corresponding index structure. More formally, given the formulation in Equation 2.28, let $f_\theta(q)$ denote the query embedding, and $\mathcal{C}_f = \{f_\theta(d_1), ..., f_\theta(d_{|\mathcal{C}|})\}$ the set of document representations – with $|\mathcal{C}|$ usually large. The retrieval problem is formulated as a Maximum Inner Product Search (MIPS)[61]:

$$i* = \arg \max_{i \in \{1,...,|\mathcal{C}|\}} f_\theta(q)^T f_\theta(d_i) \tag{2.29}$$

[61] MIPS can equivalently be formulated as a minimization problem with Euclidean distance.

MIPS can be trivially solved with exhaustive search (or brute force approach) on a flat index, which simply stores all the document embeddings: compute $f_\theta(q)^T f_\theta(d_i) \forall i$, and store the top-$k$ results in a heap data structure. While this approach remains feasible on moderate-sized collections[62], they become too costly for large-scale scenarios – both in terms of storage and query latency. Thus, more efficient methods have been developed, relying on *approximate* versions of the search problem – dubbed Approximate Nearest Neighbors.

[62] Including MS MARCO.

Local Sensitive Hashing (LSH) [Datar et al. 2004; Indyk and Motwani 1998] is based on hash functions that map similar documents – in terms of dot product – in the same "bucket" with high probability, thus partitioning the input space. At inference time, one needs to perform (exact) search on the subset of documents that belong to the hash tables the query is hashed to. Quantization follows a different approach and partitions the space according to the data. Early methods rely on a k-means clustering of the embeddings to extract k centroids $C = (c_1, ..., c_k)$ – the codebook – that are used to partition the points. The quantizer is the function mapping a (document) vector to its closest centroid:

$$Q(d) = \arg \min_{c_i \in C} ||f_\theta(d) - c_i|| \tag{2.30}$$

It then becomes possible to build an IVF (Inverted File) index, where each dimension of the codebook stores the representations of vectors assigned by the quantizer [Sivic and Zisserman 2003]. Interestingly, this design is inspired by traditional inverted indexes in text IR: here, each cluster c_i can be seen as a "virtual" word of a latent vocabulary. The large body of work around Product Quantization (PQ) [Ge et al. 2014; Jégou et al. 2011; Matsui et al. 2018] builds on this general concept, but further corrects the drawbacks of IVF indexes – which usually requires

very large codebooks. Finally, Approximate methods based on proximity graphs like HNSW (Hierarchical Navigable Small Worlds) [Malkov and Yashunin 2020] have recently gained popularity. They are considered as the current state of the art for approximate k-NN.

Remark

Many open-source implementations of ANN algorithms are available. META's FAISS library [Johnson et al. 2019][†] provides most of the state-of-the-art approaches and has become a common ground for IR researchers working on dense retrieval. Libraries like Vespa combines text and vector search at scale, making them appealing for production-ready systems[‡].

[†] https://github.com/facebookresearch/fais

[‡] https://vespa.ai/

2.6.3.3 ColBERT

In Section 2.6.3.1, we have introduced dense models representing queries and documents as single vectors in the embedding space. One notable limitation, however, is that various complex meanings can be embedded into long documents. Similarly, search queries may have multiple intents [Sanderson 2008], which might be difficult to encode in a single low-dimensional representation. Directly tackling this issue, [Humeau et al. 2020] introduce the poly-encoder, which represents each document through multiple vectors that correspond to different *views* of the input, relying on learned codes and attention mechanism. Queries are still "single-view", and are used at inference time to compute a single document representation – using an attention mechanism based on the query vector. The similarity score is finally based on the dot product between query and document representations. ME-BERT [Luan et al. 2021] follows a similar idea but simply represents a document by its first m BERT embeddings $f_\theta(d) = [h_0, h_1, ..., h_m]$ – the relevance score corresponding to the maximum dot product between the query and document vectors.

ColBERT [Khattab and Zaharia 2020; Santhanam et al. 2022b] takes the approach to its extreme by representing every token of the input (query or document) as a dense vector. By postponing the interaction of query-document tokens to the very end, it thus becomes possible to pre-compute document representations offline, while still modeling fine-grained relevance information – bridging the gap between interaction- and representation-based approaches. More formally, given the output BERT embeddings $f_\theta(q) = (h_q)_i$ and $f_\theta(d) = (h_d)_j$ for respectively the query q and document d, ColBERT computes relevance as:

$$s(q,d) = \sum_{i \in q} \text{MaxSim}(q_i, d) = \sum_{i \in q} \max_{j \in d} \cos(h_{q_i}, h_{d_j}) \tag{2.31}$$

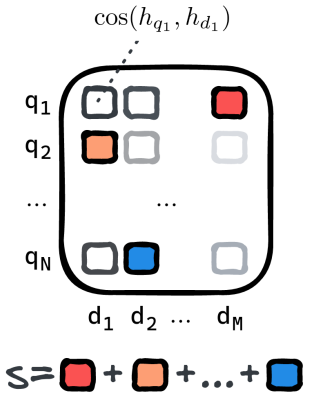


Figure 2.12: ColBERT architecture. After computing the interaction matrix, we simply sum the maximum values taken over columns, for each row.

Such mechanism is reminiscent of the interaction approaches in pre-BERT models (Section 2.5.4), as ColBERT can be seen as operating on top of the interaction matrix containing cosine distances between BERT embeddings – as shown in Figure 2.12. In practice, the same encoder f_θ is used for both queries and documents, but special tokens `[Q]` and `[D]` differentiate them. Queries are also artificially augmented by appending `[MASK]` tokens – up to a maximum length of 32 – to include a soft, differentiable query expansion mechanism. Thus, inputs are formatted as follows:

$$q \rightarrow \texttt{[Q]}, \texttt{[CLS]}, q_1, ..., q_N, \texttt{[SEP]}, \texttt{[MASK]}, ..., \texttt{[MASK]}, l = 32$$
$$d \rightarrow \texttt{[D]}, \texttt{[CLS]}, d_1, ..., d_M, \texttt{[SEP]}$$

Finally, without loss of generality, the model also incorporates a linear layer $(W, b) \in (\mathbb{R}^{d \times d'}, \mathbb{R}^{d'}), d' < d$, whose role is to project token embeddings into a lower dimensional space, thus reducing the index memory requirements. Indeed, ColBERT can either be used as a cheap re-ranker, but more interestingly, as an end-to-end retriever: its pruning-friendly MaxSim operator is specifically tailored to leverage vector similarity techniques introduced in Section 2.6.3.2. To this end, embeddings for all *terms* in the collection are stored in an index supporting approximate similarity search[63]. Note that the index size scales with the number of tokens in the collection – contrary to the number of documents for dense bi-encoders. Then, end-to-end ColBERT operates in two steps: *i*) retrieve from the index $k' < k$ tokens for each of the N_q query terms, producing a pool of $K < N_q \times k'$ unique document; *ii*) use Equation 2.31 to rank these documents[64].

Overall, this late-interaction[65] approach closes the gap between re-rankers and dense bi-encoders, offering a compelling trade-off between effectiveness and efficiency. As of today, ColBERTv2 [Santhanam et al. 2022b] is still one of the most effective IR models and has inspired various following works. [Tonellotto and Macdonald 2021] reduce ColBERT latency by pruning the number of query tokens to retrieve from, based on simple collection statistics. [Macdonald and Tonellotto 2021] further limit the number of documents to fully score in step *ii)*, relying on selection strategies based on the ANN scores of the first stage. [Hofstätter et al. 2022; Lassance et al. 2022] propose token pruning mechanisms to reduce the index memory footprint further. ColBERT-PRF [X. Wang et al. 2021] takes advantage of ColBERT token-level design to model dense PRF by selecting new embeddings to add to the query based on the IDF of the underlying tokens. [Qian et al. 2022] recently propose to learn sparsified pairwise alignments in ColBERT – improving both the effectiveness and efficiency of the approach by pruning a large number of unimportant query and document vectors. [J. Lee et al. 2023] further introduce XTR (for ConteXtualized Token Retriever) based on a novel

[63] Keeping a pointer to their document *ids*.

[64] A dedicated efficient engine for late-interaction – dubbed PLAID – has been proposed in [Santhanam et al. 2022a].

[65] To distinguish between dense bi-encoders and late-interaction approaches, we also find the single-*vs* multiple-representations terminology [Macdonald et al. 2021].

objective function that encourages the ColBERT model to retrieve the most important document tokens first – improving both effectiveness and efficiency of the approach.

> **Remark**
>
> Despite its simple design, ColBERT remains, as of today, one of the most effective IR models. Additionally, its term-level scoring mechanism bears similarities with traditional lexical approaches, allowing its analysis under the IR lens. This is the central idea of one of our contributions in Chapter 4.

2.6.4 Sparse Methods

While dense approaches have witnessed an increasing interest in the IR and NLP communities, they rely on real-time query embedding as well as Approximate Nearest Neighbors techniques, usually increasing query latency compared to methods based on inverted indexes and efficient query processing algorithms[66]. Therefore, another research direction simultaneously emerged in the field, dedicated to bringing traditional IR approaches up-to-date.

These methods falling under the "sparse umbrella" tackle the problem through different views, leading to various approaches and architectures. The overall idea is to augment traditional term-based approaches by either learning *term-weighting* functions, *expansion* methods, or a combination of both – relying on BERT (or other PLM) and the proven efficiency of inverted indexes. We can thus think of these methods as aiming to implicitly or explicitly learn sparse representations of queries and documents that can efficiently be compared by sparse dot product – following, once again, Eq. 2.11. While initially motivated by efficiency concerns, other advantages are brought by sparse modeling. For instance, scoring functions become more explainable, as we can explicitly interpret each dimension of the representation – which corresponds to terms from the vocabulary – contrary to the embedding space of dense approaches. Moreover, such design explicitly enforces lexical matching that is known to be critical in IR[67]. They can also seemingly be integrated into standard software stacks, which usually only require mono-CPU environments – compared to dense methods based on ANN that might require GPUs and multi-threading.

Following prior categorizations, we first introduce in Section 2.6.4.1 text-based expansion models. Section 2.6.4.2 is dedicated to approaches aiming to learn term weights. Section 2.6.4.3 finally introduces the general framework of sparse representation learning that can integrate both aspects for end-to-end learning.

[66] Note that this has to be taken very cautiously. Comparisons between different types of systems are rather difficult, as they depend on different hardware (GPU *vs* CPU) and software design choices (e.g., multi- *vs* mono-CPU). In any case, in large industry-scale systems, traditional approaches have withstood the test of time and remain the core component of modern search systems. We further discuss such aspects in Chapter 3, Section 3.5.5.

[67] We further develop these aspects in Chapter 4.

2.6.4.1 Text-Based Document and Query Expansion

Expansion techniques in Information Retrieval date back decades and have originally been developed to tackle the vocabulary mismatch problem, where query terms can be lexically different from those used in relevant documents[68]. The idea behind expansion is to enrich (or expand) queries and/or documents to better align them in terms of the vocabulary they use. Traditionally, such approaches have been relying on techniques like PRF (Section 2.5.1.3), but have been significantly impacted by Pre-trained Language Models.

doc2query [R. Nogueira et al. 2019b] is the first of such approaches, and is based on a simple idea: for any given document in the collection $\mathcal{C}$, use a seq2seq model to *generate* queries for which that document might be relevant. The model is trained on pairs of queries and relevant documents (q, d^+) from MS MARCO, and is used to generate N queries for each document using top-k random sampling [Fan et al. 2018], which are simply concatenated to original documents. Then, the newly "expanded collection" $\mathcal{C}'$ can be indexed and queried using standard term-based approaches like BM25. The original doc2query [R. Nogueira et al. 2019b] relies on a seq2seq transformer model trained from scratch. doc2query-T5 [R. Nogueira and J. Lin 2019] further extends the approach, using T5 [Raffel et al. 2020] as a base model. This expansion approach has two main effects (which are illustrated in Figure 2.13): it either *i*) adds new terms to documents (expansion terms), that are likely to bridge the gap with semantically relevant queries, *ii*) boosts important terms from the document, by increasing their frequency. Document expansion is quite advantageous, as query generation can be done offline, thus transferring all the workload prior to indexing and inference – the expanded terms having almost no cost on query latency. In this current form, it remains however quite heavy, as each document needs several forward passes – up to 80 – from a large transformer model like T5[69]. Despite its simplicity, doc2query-T5 offers a strong baseline, that is effective in both in- and out-of-domain settings. It also serves as the basis of several subsequent works aiming to learn term weighting (Section 2.6.4.2). By modeling query likelihood $\mathcal{P}(q|d)$ based on BERT, the TILDE model [S. Zhuang and Zuccon 2021c] has also been used as a light document expansion model in [S. Zhuang and Zuccon 2021b].

From a different perspective, other works have been tackling query expansion using PLM. For instance, CEQE [Naseri et al. 2021] extends the traditional relevance feedback model from [Lavrenko and Croft 2001] by incorporating BERT contextual representations to better select expansion terms. [Claveau 2021] conditions query expansion based on GPT-2 [Radford et al. 2019] with constrained generation – using the query as the seed for generation. Similarly to doc2query, the large number of generations encapsulates both aspects of expansion and im-

[68] Note that expansion can either refer to document or query expansion. While traditionally not as popular as query expansion, document expansion based on PLM has recently proven to be very effective.

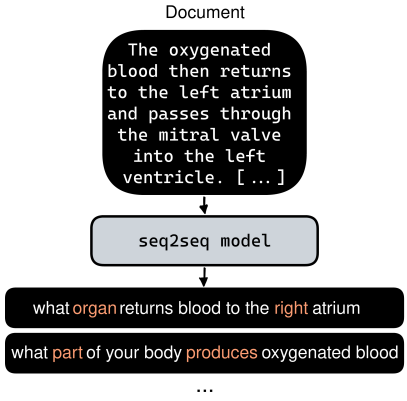


Figure 2.13: doc2query illustration. The passage is taken from MS MARCO. Orange terms are *expansion* terms (i.e., not in the original document), while others enforce term importance. We generated the example queries using the T5 HuggingFace weights at `castorini/doc2query-t5-base-msmarco`.

[69] However, expanded collections for MS MARCO have been released at `https://github.com/castorini/docTTTTTquery`, making it easier for IR researchers to experiment with the approach. Additionally, [Gospodinov et al. 2023] have recently shown how it is possible to filter irrelevant queries to either increase effectiveness or reduce index size.

plicit term weighting. While "deep" query expansion approaches have shown gains over traditional techniques like RM3, they tend to be less effective compared to term weighting models introduced in the next Section.

Remark

Recent works have shown how dense models can additionally benefit from PRF techniques [H. Li et al. 2022a, 2023; X. Wang et al. 2021; H. Yu et al. 2021], highlighting the complementary aspects of semantic matching and expansion techniques in fighting the vocabulary mismatch.

2.6.4.2 Term-Weighting

As discussed in Section 2.6.4.1, expansion can implicitly boost important terms by increasing their frequency in queries or documents. This is, however, quite inefficient, as the number of generated queries has to be large for the approach to be effective – up to 80 generated queries for doc2query-T5. Additionally, this mechanism is not learned, and is merely a byproduct of the large number of generations – it, therefore, suffers from its inability to estimate term importance precisely. Such limitations thus motivated approaches that can better estimate *contextualized* term importance in a principled way, to replace traditional weighting schemes estimated from term statistics – which are effective yet imperfect proxies for term importance.

[Dai and Callan 2019a, 2020b] first initiated such works with the DeepCT model[70]. DeepCT aims to map BERT contextualized passage term embeddings $(h_q)_i = [h_0, h_1, ..., h_N]$, with a linear projection $(W, b) \in (\mathbb{R}^d, \mathbb{R})$, to scalar values that can be interpreted as term importance[71]. Instead of explicitly optimizing ranking metrics, DeepCT is trained with a per-token regression loss, aiming to assign each token to an "ideal" term importance in the form of Query Term Recall $\in [0, 1]$, estimated from query statistics. It relies on the assumption that important terms should appear in queries for which the given document is relevant. Once the model has been trained, the collection can be indexed by re-scaling and quantizing (for instance to integer values between 0 and 100) document term weights, such that they can be stored in a standard inverted index[72]. Finally, retrieval on the augmented index is performed with BM25 – with no additional online inference needed, as queries are kept unchanged. This is a major advantage in terms of efficiency, compared to dense models, for instance, which require online query encoding[73]. A high-level overview of the model is shown in Figure 2.14. HDCT [Dai and Callan 2020a] later extended DeepCT to tackle long document ranking.

[70] Although we inform the reader that prior to PLM, some works have attempted to learn term discrimination, with limited success [Frej et al. 2020; J. Lee et al. 2020].

[71] Note that Dai and Callan also introduce a query weighting mechanism for "long" queries, but the document model is the most popular one.

[72] In practice, a new "fake" collection is created, by repeating terms according to their quantized term frequency. This collection is further processed and indexed in the standard way.

[73] Some works, however, have shown how standard query processing techniques relying on dynamic pruning are not suited to deal with such approaches, for which the distribution of weights differs from term frequencies [Mackenzie et al. 2020, 2021], see Chapter 3.

DeepCT does not include any expansion mechanism and is thus merely operating as a term re-weigher. Consequently, it does not address the vocabulary mismatch and suffers from limited Recall gains. DeepImpact [Mallia et al. 2021] combines the best of both worlds, by re-weighting terms from documents expanded with doc2query-T5. The core idea remains the same, but some mechanisms slightly differ: *i*) term scoring relies on a two-layer MLP instead of a linear layer, *ii*) the first occurrence of each unique term is provided to the scorer – instead of considering the max score for DeepCT, *iii*) the model is trained with a pairwise ranking loss, *iv*) and finally, the ranking score $s(q, d)$ is simply obtained by summing the importance of query terms appearing in documents – thus not relying on, e.g., BM25. DeepImpact quantizes weights into 8-bit integers – called impact scores – that are directly stored in an inverted index[74].

Similarly, TILDEv2 [S. Zhuang and Zuccon 2021b] learns term weighting on top of expanded documents. However, the model uses a single-layer MLP to compute scores, and represents queries and documents in BERT WordPiece vocabulary, instead of the full vocabulary of terms. This has the effect of reducing the index to a fixed small number $|V| = 30k$ of longer posting lists. Finally, authors take advantage of their previously introduced TILDE re-ranking model [S. Zhuang and Zuccon 2021c] that can cheaply provide expansion terms for each document at almost no effectiveness cost when compared to doc2query-T5 expansion. Note, however, that TILDEv2 is solely evaluated as an efficient re-ranker.

Taking a different perspective, COIL-tok [L. Gao et al. 2021a] rather learns a low-dimensional dense vector (instead of a weight) for each query and document term, providing effective contextualized term matching by means of dot product, at the expense of increased storage and retrieval cost[75]. UniCOIL [J. Lin and Xueguang Ma 2021] further extends COIL by *i*) relying on doc2query-T5 expanded documents, *ii*) projecting term representations as single scores instead of vectors. The approach is thus similar to DeepImpact, but additionally includes query term weighting, which largely boosts the overall effectiveness, while requiring additional online computation.

2.6.4.3 Sparse Representation Learning

Previously introduced models, which learn expansion and/or term weighting functions to perform sparse matching, can still be viewed under the lens of Eq. 2.11, where representation learning is somewhat implicit. For instance, DeepImpact can be seen as estimating $s(q, d)$ as a sparse dot product between a binary sparse query vector, and the learned sparse document representation. However, these approaches do not learn expansion and term weighting in a joint manner, and are, to this end, sub-optimal. For instance, DeepImpact relies on the fixed doc2query

[74] For that, *Mallia et al.* rely on the PISA search engine: https://github.com/pisa-engine/pisa.

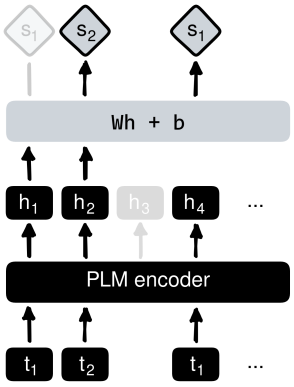


Figure 2.14: DeepCT operates in the term space by simply considering the first subword prediction for split words (as shown for t_2 here). When a word appears multiple times through the input, its maximum "importance" is kept, as shown for t_1. Up to some minor variations, most of the term-weighting approaches follow a similar design.

[75] Note that the full COIL model relies on the matching between query and document dense `[CLS]` vectors (after a downsizing projection), and can be thus considered as an "hybrid" method.

expansion, which has furthermore not explicitly been trained for the ranking task.

In the following, we give a light overview of the general framework of sparse representation learning, further developed in Chapter 3. Such approaches aim to learn sparse high-dimensional query and document vectors explicitly, which can accommodate both term weighting and expansion mechanisms, and that can be compared with dot product. We refer to this type of model as sparse bi-encoders, mirroring the dense bi-encoder terminology. [MacAvaney et al. 2020c] initiated this line of work with EPIC, which performs passage expansion by relying on the MLM head introduced in Section 2.4.4.2. More specifically, it estimates the importance of each term of the vocabulary *implied* by each term of the document, and each token is thus represented as a vector in $\mathbb{R}^{|V|}$. The overall idea is illustrated in Figure 2.15. Because of the MLM pre-training objective, such representations are already pre-conditioned, to some extent, for expansion: for a given input term, its highest MLM logits would usually correspond to the term itself, alongside synonyms or "related" terms.

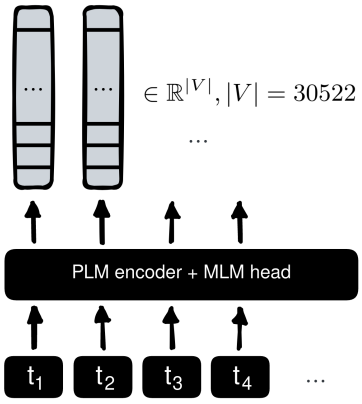


Figure 2.15: General overview of high-dimensional token representations relying on the MLM head. Such representations are then aggregated to obtain query- or document-level vectors.

Different aggregation mechanisms have been proposed to obtain document- or query-level vectors. Moreover, for obvious efficiency reasons, it is required for representations to be sparse, which is also achieved by various means. Although EPIC has been designed as an efficient re-ranking technique – since there is no control over the sparsity of representations – it set the ground for vocabulary-based representation learning. [Bai et al. 2020] builds on this idea and introduce SparTerm, the first sparse bi-encoder designed to tackle first-stage ranking efficiently. However, the model cannot be learned end-to-end, resulting in limited success. By building on the Query Term Independence Assumption [Mitra et al. 2019], SPARTA [Zhao et al. 2021] takes a different perspective and introduces a matching function similar to ColBERT, where query tokens are not contextualized. It thus becomes possible to compute, prior to indexing, term matching scores $y_v = \max_{j \in d} \cos(e_v, h_{d_j})$, and this for every term v in the BERT vocabulary – resulting to a formulation close to MLM[76].

Note that the MLM head is not *a priori* necessary to represent text in high dimensional space. It, however, provides a powerful inductive bias, where each dimension corresponds to an actual vocabulary token. For instance, [Lassance et al. 2021a] propose an autoencoder approach that maps frozen dense vector representations to latent topics in an unsupervised fashion, which are sparse by design and can, therefore, efficiently be indexed[77].

All in all, we list in Table 2.11 existing sparse approaches with their main characteristics.

[76] SPARTA incidentally bridged the gap between interaction-based models like ColBERT and sparse models based on MLM.

[77] We can easily make the parallel with SNRM [Zamani et al. 2018] introduced in Section 2.5.4, whose high-dimensional projection is learned from scratch.

Remark

Inspired by such ideas, we have developed SPLADE [Formal et al. 2021a, 2022a, 2021c], the first model designed to efficiently tackle first-stage ranking by jointly learning expansion and term-weighting. We give a detailed overview of the approach in Chapter 3. However, to motivate our statements, we show in Table 2.10 how SPLADE outperforms both previous sparse approaches as well as dense bi-encoders.

Table 2.10: Comparing SPLADE to dense bi-encoders and previous sparse approaches on MS MARCO passage retrieval (dev set).

model	MS MARCO dev	
	MRR@10	R@1000
▶ Sparse		
BM25	18.4	85.3
DeepCT	24.3	91.3
doc2query-T5	27.7	94.7
SPLADE ♢	34.5	96.5
▶ Dense		
Bi-encoder	31.2	94.1

model	QE	DE	TW	e2e
doc2query-T5 [R. Nogueira and J. Lin 2019]	✗	✓	✗	✗
DeepCT [Dai and Callan 2020b]	✗	✗	✓	✗
DeepImpact [Mallia et al. 2021]	✗	✓	✓	✗
TILDEv2 [S. Zhuang and Zuccon 2021b]	✗	✓	✓	✗
COIL-tok [L. Gao et al. 2021a]	✗	✗	✓	✓
UniCOIL [J. Lin and Xueguang Ma 2021]	✓	✓	✓	✗
SPARTA [Zhao et al. 2021]	✗	✗	✓	✗
SparTerm [Bai et al. 2020]	✗	✓	✓	✗
SPLADE-doc [Formal et al. 2021a] ♢	✗	✓	✓	✓
SPLADE [Formal et al. 2021c] ♢	✓	✓	✓	✓
SpaDE [E. Choi et al. 2022]	✗	✓	✓	✓
Efficient-SPLADE [Lassance and Clinchant 2022]	✓	✓	✓	✓

Table 2.11: A cartography of existing sparse approaches for first-stage ranking. `QE`, `DE`, `TW`, and `e2e` refer respectively to `Query Expansion`, `Document Expansion`, explicit `Term Weighting` and `end-to-end`, which are the main characteristics differentiating models. We notice that the end-to-end aspect of SPLADE allows us to seemingly apply the same techniques introduced for dense approaches such as distillation (Table 2.9). Also, note that each model relies on additional specifics and implementation details. A similar description is proposed in [Thong Nguyen et al. 2023].

2.6.5 Training Bi-encoders

We now proceed to describe how ranking models based on PLM are usually trained. Section 2.6.5.1 describes the fine-tuning step. In Section 2.6.5.2, we further present recent works tackling pre-training of PLM in the context of Information Retrieval.

2.6.5.1 Fine-Tuning

Up to now, we haven't precisely discussed how PLM-based models are trained – or more accurately *fine-tuned* – for the ranking task[78]. In the following, we focus on bi-encoders – whether dense or sparse. Let our model be parametrized by θ — the set of parameters to learn. From a high-level perspective, we aim to learn a suitable embedding function that maps queries and relevant documents together, while pushing away irrelevant ones.

In IR, we usually have access to training samples, obtained from instance from the logs of a search engine, in the form of *i*) a query q, *ii*) a corresponding relevant document d^+, *iii*) a set of m negative documents $\{d_1^-, ..., d_m^-\}$. To optimize ranking models from this data, many losses have been proposed in the Learning-to-Rank literature, such as Hinge Loss or ListNet [T.-Y. Liu 2011]. However, there recently has been an increasing interest towards contrastive approaches to learn effective query and document representations[79]. It aims to minimize the negative log-likelihood of the positive passage against the set of m negative documents:

$$\mathcal{L} = -\log(\mathcal{P}_\theta(q|d^+)) \tag{2.32}$$

where:

$$\mathcal{P}_\theta(q|d^+) = \frac{\exp(s_\theta(q, d^+))}{\exp(s_\theta(q, d^+)) + \sum_{i=1,...,m} \exp(s_\theta(q, d_i^-))} \tag{2.33}$$

In an ideal setting, all the negative documents in the collection $\mathcal{C}$ would be taken into account. However, this is computationally infeasible, as the collection contains millions – or more – documents for which it would be required to compute $s_\theta(q, d)$. Hence, m is practically set to a small number, such that we approximate the true distribution over the collection – following Noise Contrastive Estimation [Gutmann and Hyvärinen 2010] literature.

The way negatives are selected to build training examples is thus critical and has recently been the focus of several works in neural IR[80]. One obvious way consists of sampling random negatives from the collection $\mathcal{C}$. In-Batch Negatives (IBN) is another option that considers positive documents from other queries in the current batch as negative documents. Both types of approaches, however, suffer from slow

[78] Recall from Section 2.4.4.2 that PLM are pre-trained with an unsupervised objective such as MLM, to be later fine-tuned for downstream tasks.

[79] That has also been extensively studied in Computer Vision, e.g., [Ting Chen et al. 2022].

[80] Note that this problem is rather new, as it is directly linked to training first-stage models. Re-rankers do not suffer from such issues, as samples are simply built from the candidate generator.

training convergence [L. Xiong et al. 2021], as they end up providing non-informative samples[81].

[81] Such negatives can be seen as too "easy" for the model. Consequently, they usually provide near-zero loss values, and thus near-zero gradients.

Thus, several approaches have focused on providing more informative – or *hard* – negatives to train neural IR models. They are usually based on a retrieval model that can identify negative documents close enough to the relevant ones. As such, [Karpukhin et al. 2020] propose to use BM25 as a way to select negatives – combined with local IBN. This procedure remains the "standard" way to train neural retrievers. Other works have proposed to rely on the models themselves to select even more informative negatives, typically in a multi-step or dynamic training strategy [Lindgren et al. 2021; Qu et al. 2021; Ren et al. 2021b; L. Xiong et al. 2021; Zhan et al. 2021, 2020]. For instance, ANCE [L. Xiong et al. 2021] periodically indexes at training time the collection with the current model. Such an index is used to dynamically retrieve negative documents for subsequent training queries. The Learning To Retrieve (LTRe) approach freezes the document index and further trains a query encoder to learn how to retrieve relevant documents from the entire corpus.

Other developments have focused on distillation strategies [Hinton et al. 2015]. In Natural Language Processing and Computer Vision, distillation is traditionally employed to distill a large model into a smaller one with the same overall architecture. However, in Information Retrieval, the objective is to distill an effective – but costly – model like a cross-encoder into an efficient one, such as a bi-encoder. Hence, [Hofstätter et al. 2020a] first proposed the marginMSE loss, which minimizes, with Mean Squared Error, the positive-negative margins between a cross-encoder teacher T and various student models S:

$$\mathcal{L} = ((s_T(q, d^+) - s_T(q, d^-)) - (s_\theta(q, d^+) - s_\theta(q, d^-))^2 \tag{2.34}$$

Similarly, [S.-C. Lin et al. 2021] minimize the Kullback–Leibler divergence between $\mathcal{P}_T(q|d^+)$ and $\mathcal{P}_\theta(q|d^+)$, for a light ColBERT teacher, enabling the use of dynamic In-Batch Negatives. Finally, [Hofstätter et al. 2021] combine both types of distillation. Overall, cross-architecture distillation has become quite standard and is a simple and effective way to boost the performance of bi-encoders.

Remark

The hard-negative sampling and distillation techniques introduced in this Section have initially been designed to boost the effectiveness of *dense* bi-encoders. They are further developed in Chapter 3, where we show how such techniques can seemingly be applied to *sparse* bi-encoders, leading to state-of-the-art results for SPLADE.

2.6.5.2 Pre-Training

Masked Language Modeling pre-training is not entirely aligned with the Information Retrieval task, and is, to this end, sub-optimal. The learned embedding space is not suited for retrieval – or to compute similarities more generally. Recent works have proposed to align this space to the task better, by performing an additional intermediate *unsupervised* pre-training step[82] – thus providing a more effective initialization for subsequent fine-tuning

In the context of re-ranking, [Xinyu Ma et al. 2021a,b] propose Pre-training with Representative wOrd Prediction (PROP). Given an input document, two pseudo queries are generated, and query likelihood is used to build pairwise preferences to further pre-train the model alongside MLM[83]. In a similar pairwise setting, [Jia Chen et al. 2022] incorporate IR axioms – formal descriptions of what makes a good ranker – into pre-training, by generating samples according to such axioms. [Z. Ma et al. 2021] propose the HARP framework, in which pre-training objectives take advantage of dependencies between hyperlinks and anchor texts in Wikipedia pages. [Y. Guo et al. 2022] further devise four tasks that rely on the hierarchical structure of Web documents.

Another line of work has studied the role of pre-training in the context of first-stage ranking, i.e., applied to dense bi-encoder models, and can roughly be categorized into *i*) contrastive and *ii*) information bottleneck techniques.

[Chang et al. 2020] early studied the role of several pre-training tasks such as the Inverse Cloze Task – initially proposed by [K. Lee et al. 2019] – showing how they can help a transformer-based bi-encoder trained from scratch to outperform BM25. CoCondenser [L. Gao and Callan 2022] builds on the Condenser architecture [L. Gao and Callan 2021] designed to enforce models to condense information into dense vectors. CoCondenser is pre-trained with a new unsupervised objective: given a document, sample a pair of spans, and contrastively train the model such that the `[CLS]` representations of spans from the same document are close, while the ones from different documents are far apart. [Izacard et al. 2022] carefully study the limits of unsupervised contrastive pre-training, and propose the Contriever model, which outperforms BM25 on several zero-shot evaluation settings. [Ram et al. 2022b] build on similar ideas by taking advantage of recurring spans in documents, in the context of open-domain Question Answering.

From a different perspective, other methods have proposed to pre-train query and document representations through an Information Bottleneck, in an AutoEncoder fashion [Sachan et al. 2022b; Shen et al. 2022a; K. Wang et al. 2021; Xiao and Z. Liu 2022; Xiao et al. 2022]. The general idea consists in better pre-conditioning the network to rely on its `[CLS]` representation to match two pieces of text.

[82] Thus, almost all these approaches start from a pre-trained model, and further devise additional pre-training objectives. We refer to such intermediate steps as *middle-training*.

[83] The two methods differ in the way they generate the set of words, either from a multinomial uni-gram Language Model or a contextual Language Model based on BERT.

Remark

Note that these pre-training techniques have either been applied to re-ranking or dense retrieval settings. They remain largely unexplored in the context of sparse representation learning. In Chapter 3, we do not explicitly devise pre-training tasks for sparse models. However, we rely on middle-trained dense checkpoints, which can *surprisingly* improve the performance of sparse retrievers.

2.6.6 Active Research Directions

This section introduces several active research directions in the neural IR space. We don't cover such topics in this thesis.

Weak labeling Obtaining high-quality, large-scale annotated retrieval datasets is notoriously hard. It has motivated the study of cheap weak labeling techniques early on [Dehghani et al. 2017; MacAvaney et al. 2019b]. Various works have recently shown how generative Language Models can be used to generate synthetic training data in the context of Information Retrieval [Bonifacio et al. 2022; Dai et al. 2022; J. Ma et al. 2021; Saad-Falcon et al. 2023; Sachan et al. 2022a; K. Wang et al. 2022]. For instance, GPL [K. Wang et al. 2022] combines a doc2query-T5 query generator with pseudo labeling from a cross-encoder, to adapt dense retrievers on new domains with marginMSE loss. InPars [Bonifacio et al. 2022] relies on GPT-3 to generate synthetic queries from input documents, that are used to train a monoT5 re-ranker with Cross-Entropy loss – leading to impressive unsupervised performance. [Boytsov et al. 2023; Jeronymo et al. 2023] recently extend and update the approach. PROMPTAGATOR [Dai et al. 2022] follows a similar line of work by relying on the FLAN model [Wei et al. 2022] as a few-shot query generator.

Differentiable Search In a position paper, [Metzler et al. 2021] present a research agenda for the IR community, based on the recent progress of generative Language Models. They propose to rethink the standard retrieve-rerank pipeline (Section 2.5.2), which could be replaced by a single generative step. The model would thus encode all the information contained in the corpus *in its parameters*, and directly answer the information needs. They identify key advantages (such as getting rid of the document index) and practical limitations (such as scaling or trustworthiness). Building on this idea, [Tay et al. 2022b] introduce the first technical implementation of generative search, with the Differentiable Search Index (DSI). The IR task is cast as a text-to-text problem, where the model maps (i.e., generates) string queries to relevant document *ids*. Several following works have improved over

such idea [Bevilacqua et al. 2022; Jiangui Chen et al. 2022; H. Lee et al. 2022a,b; Mehta et al. 2022; Y. Wang et al. 2022; Zhou et al. 2022; S. Zhuang et al. 2022]. However, due to the difficulty of the task, and the issues of scaling such approaches, the evaluation scenarios remain as of today quite limited.

Retrieval-Enhanced Machine Learning Building on quite the opposite idea, there has been a recent trend departing from ever-larger models storing knowledge in their parameters, to models having an explicit memory – thus offloading memorization to storage. For instance, models such as REALM [Guu et al. 2020] early integrated a dense bi-encoder model in the pre-training of BERT, where retrieved documents are used as additional context. [Zamani et al. 2022] recently introduced the generic Retrieval-Enhanced Machine Learning (REML) framework, placing IR models in a broader scope, where they could provide access to more abstractly-represented knowledge. Such a research agenda highlights exciting opportunities for the IR community in the larger scope of Machine Learning optimization.

2.7 Conclusion

The advent of Pre-trained Language Models has irreversibly impacted Information Retrieval. Re-ranking approaches based on cross-encoders (Section 2.6.1) have shifted the state of the art on standard benchmarks (e.g., MS MARCO, Section 2.3.2.1) beyond what the community thought would be possible. Motivated by these improvements, new *retrieval* models have been designed as a direct alternative to traditional lexical approaches like BM25 in multi-stage pipelines to bridge the vocabulary gap (Section 2.6.2). Such approaches can roughly be categorized into *dense* or *sparse* depending on how they represent queries and documents. Due to their fundamental differences, both rely on different index structures as well as search algorithms to cope with the strict efficiency requirements of the retrieval step (Sections 2.6.3.2 and 2.5.1.4 respectively).

In Chapter 3, we contribute to this research direction and introduce a new sparse bi-encoder model – dubbed SPLADE for SParse Lexical AnD Expansion. SPLADE is highly *effective*, *efficient*, in addition to being *robust*. Additionally, its learned representations can be *interpreted*.

While neural ranking models based on PLM achieve impressive results in *in-domain* evaluation scenarios such as MS MARCO, various studies have recently pointed out their limitations when it comes to *generalization*, compared to traditional lexical approaches which are inherently *zero-shot* [Sciavolino et al. 2021a; Thakur et al. 2021; Zhan et al. 2022b]. We further discuss such aspects for SPLADE in Chap-

ter 3, and study more generally the ability of ranking models to perform lexical match *off-the-shelf* in Chapter 4.

Chapter 3

Sparse Representation Learning for Information Retrieval

Outline

In this Chapter, we present the main contribution of this thesis. Our model – dubbed SPLADE, for SParse Lexical AnD Expansion – is a *sparse* bi-encoder that learns query and document representations grounded in the Pre-trained Language Models vocabulary. SPLADE is *effective*, *efficient*, and its representations can be *interpreted*. It relies on an explicit sparsity regularization of representations which allows controlling the effectiveness-efficiency trade-off in an end-to-end fashion. SPLADE achieves state-of-the-art results in both in- and out-of-domain settings when combined with training techniques such as distillation or hard-negative sampling – making it an appealing candidate to replace traditional term-based approaches such as BM25 in modern search engines.

This Chapter is built on our four contributions:

- [Formal et al. 2021c] *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking.*
- [Formal et al. 2021a] *SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.*
- [Lassance et al. 2021b] *Naver Labs Europe (SPLADE) @ TREC Deep Learning 2021.*
- [Formal et al. 2022a] *From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective.*

3.1 Introduction

The recent developments in Pre-trained Language Models have given birth to a new generation of neural IR systems. Models like BERT [Devlin et al. 2019] have irreversibly impacted Information Retrieval – way beyond what the community thought would be possible a few years back. Initially employed as re-rankers in a standard Learning-to-Rank framework, the real paradigm shift, however, occurred with the emergence of a new generation of neural models tackling the *retrieval* step in multi-stage ranking pipelines – holding the promise to replace traditional IR approaches in modern search engines.

For such a task, models based on dense representations combined with Approximate Nearest Neighbors algorithms, have proven to be both highly effective and efficient (Section 2.6.3). These approaches tend to move away from sparse signals and the long-standing vocabulary mismatch by directly modeling relevance as distances in the continuous semantic space. In the meantime, several works have been dedicated to bringing lexical models up to date by taking advantage of PLM in various manners. Such sparse approaches, for instance, learn contextualized term weights, query or document expansion, or use both mechanisms jointly (Section 2.6.4). They inherit the good properties of Bag-of-Words models, such as the exact matching of terms or some level of interpretability. They additionally benefit from the proven efficiency of inverted indexes and can seamlessly be integrated into standard IR software stacks – enabling GPU-free inference in some cases.

Motivated by these advantages, we propose in this Chapter an original *sparse* bi-encoder model – dubbed SPLADE for SParse Lexical AnD Expansion – that unifies expansion and term weighting. We, therefore, develop the ideas introduced in Section 2.6.4.3, from which we took inspiration to design SPLADE. We describe in Section 3.2 how the MLM head of Pre-trained Language Models can be used to *represent* tokens in a sequence as vectors in the vocabulary space. In Section 3.3, we present models that aggregate such token-level representations into query and document vectors. We address the main pitfalls of such approaches with SPLADE in Section 3.3.4. The remainder of the Chapter is dedicated to the various experiments and analyses we conduct, resulting in state-of-the-art results in in- and out-of-domain settings (Sections 3.7 and 3.8 respectively).

3.2 Representing Tokens in the Vocabulary Space

In sparse representation learning for IR, term importance is predicted in BERT's WordPiece vocabulary – where $|V| = 30522$ – based on the logits of the Masked Language Modeling head[84]. More precisely, let us consider an input query or document sequence after WordPiece tokenization $t = (t_0, t_1, ..., t_N) = (\texttt{[CLS]}, t_1, ..., t_{N-1}, \texttt{[SEP]})$. BERT

[84] Note that most approaches, including SPLADE, are originally based on BERT; we show in Section 3.5.6 how it is possible to use various PLM models, as long as they provide a pre-trained MLM head.

outputs contextualized embeddings $(h_0, h_1, ..., h_N)$ for each token[85] in t. The MLM head projects each embedding h_i back to the vocabulary space $\mathbb{R}^{|V|}$. To ease reading, we re-introduce a slightly modified formulation of the MLM head (Eq. 2.19, Section 2.4.4.2). More specifically, the importance (logit) w_{iv} of the token v (of the vocabulary) for a token i (of the input sequence) is given by:

$$w_{iv} = \text{transform}(h_i)^T E_v + b_v, \quad v \in \{1, ..., |V|\} \tag{3.1}$$

where E_v denotes the BERT input embedding for token v, b_v is a token-level bias, and transform(.) is a linear layer with GeLU activation and LayerNorm. The matrix projecting contextualized representations h_i back to the vocabulary is *tied* to the input embedding matrix $E_{|V| \times d}$. While nothing *a priori* prevents having a separate output projection matrix W, tying the input and output embedding matrices pre-dates BERT and has shown to be both an effective and efficient design choice – by reducing the number of training parameters [Press and L. Wolf 2017] – thus becoming standard for Pre-trained Language Models.

As illustrated in Figure 3.1, the weights w_i correspond to an unnormalized probability distribution over the vocabulary, for token t_i, where each output dimension $v \in V$ is actually associated with the token it represents – therefore providing an *interpretable* representation basis for IR models. During pre-training, a softmax is applied to the term importance of `[MASK]` tokens, which are then used to train the models with Cross-Entropy to predict the actual masked tokens (Section 2.4.4.2). The MLM head is usually discarded during fine-tuning – only the contextualized embeddings h_i are used for downstream tasks. However, it provides a well-calibrated initialization point for representing queries and documents in the vocabulary space by implicitly learning term *expansion* with the MLM task.

We illustrate this phenomenon for two pre-trained models in Table 3.1. As the task requires predicting masked tokens from their context, expansion thus naturally occurs. For instance, synonyms of a given word are usually associated with high logits, as they could likely occur in the same context – for instance "*cancer*"→"*tumor*" in Table 3.1. In Information Retrieval, we can see this as propagating importance to similar tokens.

[85] Note that we consider tokens to be tokenized units of the WordPiece vocabulary.

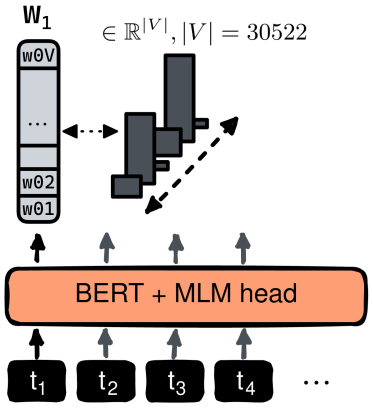


Figure 3.1: Illustration of the MLM head. Each token t_i is mapped to an unnormalized distribution over the vocabulary (the logits) – shown for token t_0 here.

`>> distilbert-base-uncased` [Sanh et al. 2019]	
prostate	→ (*prostate*, *.*, *the*)
cancer	→ (*cancer*, *tumor*, *##mour*)
detection	→ (*detection*, *identification*, *screening*)
treatment	→ (*.*, *;*, *treatment*)
`>> google/electra-base-generator` [Clark et al. 2020]	
prostate	→ (*prostate*, *##iate*, *mal*)
cancer	→ (*cancer*, *cancers*, *tumor*)
detection	→ (*detection*, *detecting*, *screening*)
treatment	→ (*treatment*, *therapy*, *treatments*)
`>> naver/splade-cocondenser-ensembledistil` [Formal et al. 2021c]	
prostate	→ (*prostate*, *bp*, *ur*)
cancer	→ (*cancer*, *tumor*, *disease*)
detection	→ (*detection*, *detect*, *test*)
treatment	→ (*treatment*, *therapy*, *treated*)

Table 3.1: MLM prediction for the Robust04 query "*prostate cancer detection treatment*". For each token, we show the top-3 predicted tokens, i.e., with the highest MLM logits. Note that these models use the WordPiece vocabulary (Section 2.4.2). Also, note that only the generator part of ELECTRA has been pre-trained with MLM. We also include for comparison the prediction for one of our SPLADE model.

We thus see that such vocabulary-based representations offer two nice properties, namely *i*) *expansion* and *ii*) *term-weighting*, as tokens "close" to an input token t_i have higher values (logits) than non-related ones.

Remark

This approach conceptually differs from term-weighting models such as DeepImpact [Mallia et al. 2021], which learn scoring functions mapping contextualized embeddings to a score in $\mathbb{R}$. In SPLADE and related models, each embedding is mapped to a vocabulary-sized representation in $\mathbb{R}^{|V|}$.

3.3 From Token to Sequence-Level Representations

In Section 3.2, we have shown how we can *represent* tokens as vectors in the vocabulary space, therefore explicitly modeling expansion and term weighting. Different approaches have been proposed to derive query and document views from the token representations by aggregating importance vectors w_i for tokens t_i in the input sequence. Importantly, representations are located in a high-dimensional space $\mathbb{R}^{|V|}$, with $V >> 10k$. To perform efficient retrieval from large collections with inverted indexes, such representations must be *sparse*. Thus, methods also differ in the way they enforce such sparsity.

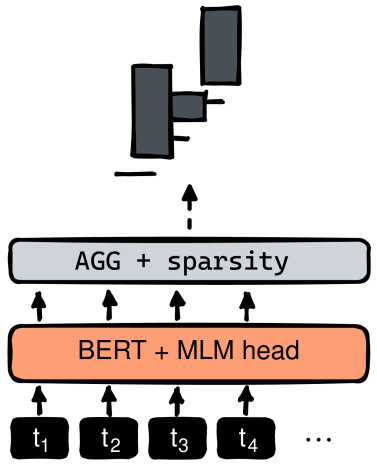


Figure 3.2: Sparse representation learning based on the MLM head. Ranking models rely on *i)* an aggregation mechanism, *ii)* a method to make representations sparse.

The overall scheme of sparse representation learning based on the output of the MLM head is represented in Figure 3.2. It follows the representation paradigm from Eq. 2.11, where we aim to learn encoders[86] ϕ and ψ that represent queries and documents as sparse vectors w, and where relevance is expressed as :

$$s(q,d) = \phi(q)^T \psi(d) \tag{3.2}$$

[86] While in the dense case, query and document encoders are usually the same, we show in Section 3.6 that for sparse approaches, query-specific encoders can lead to substantial efficiency gains. Also, note that such encoders are usually parametrized, i.e., $\phi = \phi_\theta$, but we omit θ for simplicity.

Remark

Depending on the approach, query and document encoders can be different or symmetric. We note $w^{(q)}$ (resp. $w^{(d)}$) vectors for queries (resp. documents). We also indicate with subscripts selection operations: w_{iv} corresponds to the v-th token of the i-th position of the input sequence, while w_v simply corresponds to the v-th dimension of a query or document vector.

3.3.1 EPIC

EPIC [MacAvaney et al. 2020b] (for Expansion via Prediction of Importance with Contextualization) is a lightweight re-ranking module based on vocabulary-grounded query and document representations. It is the first approach proposing to perform passage expansion based on a mechanism similar to Eq. 3.1. In contrast, queries are treated similarly to term-weighting approaches like DeepCT [Dai and Callan 2019a, 2020b], i.e., by mapping each query token t_i to a single score $w^{(q)}(t_i)$ as follows:

$$w^{(q)}(t_i) = \log\left(1 + \text{softplus}\left(\theta_1^T h_{q_i}\right)\right) \tag{3.3}$$

where $\theta_1 \in \mathbb{R}^d$. The query is represented as a sparse vector $\phi(q)$ in $\mathbb{R}^{|V|}$ by setting to 0 all the other dimensions (tokens) in the vocabulary – it thus does not perform expansion. The softplus function prevents terms from having a negative importance while enforcing a saturation effect through the logarithm[87].

[87] We show in Section 3.5.4 how saturating term importance is actually quite helpful for SPLADE.

Contrary to queries, documents are represented as *dense* vectors in the vocabulary space $\mathbb{R}^{|V|}$ by means of three components. More precisely, document terms t_j are first projected to the vocabulary V, i.e., $w_j = \theta_2 h_{d_j}, \forall j$. Note that while θ_2 is initialized to the pre-trained MLM projection matrix (or equivalently, the input embedding E), it is, however, no *tied* to it. Such projection is akin to what is done in Eq. 3.1 – although not strictly equivalent. In addition, term importance is taken into account, similarly to Eq. 3.3:

$$w^{(d)}(t_j) = \log\left(1 + \text{softplus}\left(\theta_3^T h_{d_j}\right)\right)$$

Finally, a global feature for the passage is computed, based on the representation of the `[CLS]` token: $c(d) = \sigma(\theta_4^T h_{d_{\texttt{[CLS]}}})$, where σ is the sigmoid function. The importance of each token v of the vocabulary V is finally given by:

$$w_v = c(d) \max_{j \in d} \left(w^{(d)}(t_j) w_{jv} \right), \quad v \in \{1, ..., |V|\} \tag{3.4}$$

While queries are *sparse* by design, Eq. 3.4 implies that documents are *dense* vectors in the vocabulary space. MacAvaney et al. indeed focus on the re-ranking task, for which exhaustive computation of large dot products is feasible – i.e., without requiring fast query processing over inverted indexes. The approach is efficient, in the sense that document representations can be computed offline – at the expense of a large index ($d \in \mathbb{R}^{|V|}$). The authors further study the role of pruning, by observing that a large proportion of the values in document vectors are actually low. Documents representations can be reduced up to 1000 non-zero values, by simple top-k pooling (after training), without any loss of effectiveness[88].

[88] Note that 1000 is still quite large – $10e^8$ weights for $1M$ documents – but further reducing k (e.g., $k = 100$) leads to performance drops.

> **Remark**
>
> Even though EPIC is not strictly relying on the MLM head to represent queries and documents, and only tackles re-ranking, it still has set the basis for grounding query and document representations in the PLM vocabulary space in neural Information Retrieval.

3.3.2 SPARTA

In the context of Open-Domain Question Answering, SPARTA [Zhao et al. 2021] (for Sparse Transformer Matching) addresses the problem from a different angle, by introducing a matching function similar to ColBERT, where query tokens are *not* contextualized. More specifically, given the *input* BERT embeddings $\mathcal{E}(q) = (e_q)_i$, and the output BERT embeddings $(h_d)_j$ for respectively the query q and document d, the relevance score is computed as:

$$y_i = \max_{j \in d} \cos(e_{q_i}, h_{d_j}) \tag{3.5}$$

$$s(q, d) = \sum_{i \in q} \log(\text{ReLU}(y_i + b) + 1) \tag{3.6}$$

where b is a trainable bias. Because query token embeddings are non-contextual, it thus becomes possible to compute, prior to indexing, term matching scores $y_v = \max_{j \in d} \cos(e_v, h_{d_j})$, and this for every term v in the WordPiece vocabulary. This results in viewing documents as

vectors w in the vocabulary space[89], with a formulation *implicitly* close to MLM. The ReLU function, coupled with b, enforces sparsity such that documents only have a handful of non-zero term matching scores y_v. In practice, however, there is no explicit control of the sparsity, and SPARTA representations are quite large – up to several thousand terms. Zhao et al. further study the effect of a post-processing top-k pooling strategy – similar to EPIC – and manage to reduce vectors up to a few hundred dimensions without significant drops in effectiveness.

[89] Queries in the meantime can be seen as sparse binary BoW vectors.

Remark

SPARTA incidentally bridged the gap between interaction-based models like ColBERT and sparse bi-encoders based on the MLM head. Even if the latter aim to represent queries and documents as single vectors, they are, to some extent, computing interactions with a *fixed* vocabulary basis.

3.3.3 SparTerm

SparTerm [Bai et al. 2020] (for Term-based Sparse Representations) is the first sparse bi-encoder designed to explicitly tackle first-stage ranking. It relies on the term-level, vocabulary-grounded representations from the MLM head expressed in Eq. 3.1. The final query or document vector is then obtained by summing importance predictors over the input sequence t, after applying ReLU to ensure that term weights are positive[90]:

$$w_v = g_v \times \sum_{i \in t} \text{ReLU}(w_{iv}), \quad v \in \{1, ..., |V|\} \tag{3.7}$$

where g_v is a binary mask (or gating controller) that controls which terms to include in the representation, by turning off certain dimensions v of the vocabulary V. Vectors can be made sparse, thus allowing efficient retrieval from an inverted index. Bai et al. introduce two masking mechanisms:

- *lexical-only* is a BoW masking, i.e., $g_v = 1$ if token v appears in t, and 0 otherwise;
- *expansion-aware* is a lexical-expansion-aware binary gating mechanism based on BERT, which *learns* to select the terms to appear in the final representation. To preserve the original input, it is forced to 1 if the token v appears in t.

The binary nature of the gating controller g prevents training the whole model end-to-end. Thus, the controller is first trained on four

[90] Because the encoder is symmetric, i.e., $\phi = \psi$, we use i to denote the position in the input sequence, whether it's a query or a document.

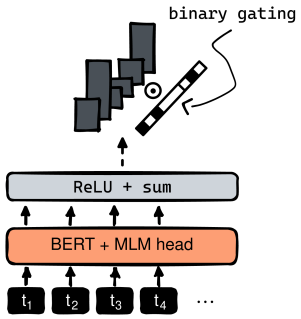


Figure 3.3: High-level overview of SparTerm. The importance predictor is combined with a binary mask that controls which terms to include. Note that in the *expansion-aware* case, the binary controller is based on BERT, and learned on auxiliary tasks *before* fine-tuning.

auxiliary tasks to learn how to expand. It is then fixed, while fine-tuning the term-weighting model with negative log-likelihood, on samples built from BM25 (Section 2.6.5.1). A threshold value is chosen for the binarizer, such that representations are sparse enough.

> **Remark**
>
> SparTerm expansion-aware gating is somewhat intricate. The model cannot be trained end-to-end, therefore preventing it from learning the optimal pruning strategy for the ranking task. Moreover, the two lexical and expansion-aware strategies do perform almost equally well, questioning the actual benefits of expansion (see Section 3.5.2, Table 3.3). Finally, including *all* input tokens is also questionable (e.g., stopwords).

3.3.4 SPLADE

SPLADE (for SParse Lexical AnD Expansion) was born from these preliminary works. We propose slight yet essential changes to the SparTerm model, that dramatically improve its effectiveness, efficiency, as well as its "practicality", making SPLADE the first sparse model to outperform dense approaches[91].

[91] In equivalent training settings (see Section 3.7.2.2).

3.3.4.1 Saturated Term Weighting

We first modify the importance estimation from Eq. 3.7, by introducing a log-saturation effect which prevents some terms to dominate[92]:

$$w_v = \sum_{i \in t} \log\left(1 + \text{ReLU}(w_{iv})\right), \quad v \in \{1, ..., |V|\} \tag{3.8}$$

[92] We also remove the gating controller – we detail in the next Section how we can achieve sparsity.

It is inspired by sublinear term frequency scaling in traditional IR models[93], which hypothesizes that relevance should not be linear w.r.t. to tf. Such ideas were formally developed in axiomatic approaches to IR [Fang et al. 2004], and models based on $\log(tf)$ are usually more effective than their unscaled counterparts. This type of saturation has proven effective in various neural IR models, including ones based on BERT, such as EPIC (Section 3.3.1) or SPARTA (Section 3.3.2).

[93] For instance, described in Section 6.4.1 in [Manning et al. 2008].

In early experiments and analyses, we noticed that BERT's pre-trained MLM layer used at initialization tends to output a rather large amount of positive values (i.e., several thousand). Because importance estimation is unbounded (ReLU), the model can adjust the margin in the ranking loss solely by increasing values for dimensions corresponding to important terms. Thus, nothing is inherently pushing for sparsity. Adding a saturation function such as log enforces the model to turn off noisy dimensions very early during training, such that it focuses on

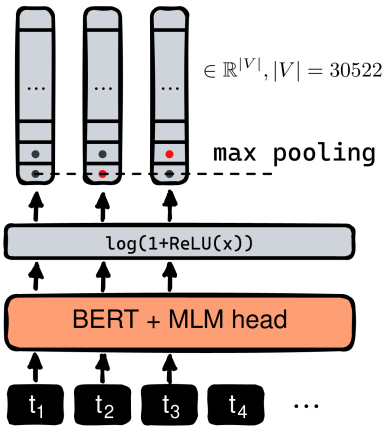


Figure 3.4: High-level overview of SPLADE-max. Red dots correspond to the max values over the sequence.

important expansion components – resulting in already sparse representations (see Section 3.5.2, Table 3.3). Finally, we noticed experimentally that replacing the sum aggregation by a max – illustrated in Figure 3.4 – actually results in much more effective models (see Section 3.5.3)[94]:

$$w_v = \max_{i \in t} \log\left(1 + \text{ReLU}(w_{iv})\right), \quad v \in \{1, ..., |V|\} \tag{3.9}$$

With this formulation, the model selects, for each dimension v, the most salient *contribution* among input tokens.

SPLADE has an overly simple design – besides the transformer architecture – and can be written in a few lines of Python code, as shown in Listing 3.1:

```python
import torch
from transformers import AutoModelForMaskedLM


class Splade(torch.nn.Module):

    def __init__(self, model_type_or_dir):
        super().__init__()
        self.transformer = AutoModelForMaskedLM.from_pretrained(
    model_type_or_dir)

    def forward(self, **kwargs):
        out = self.transformer(**kwargs)["logits"]
        values, _ = torch.max(torch.log(1 + torch.relu(out)) *
    kwargs["attention_mask"].unsqueeze(-1), dim=1)
        return values
```

Listing 3.1: PyTorch implementation of SPLADE-max.

3.3.4.2 Learning Sparse Representations

The idea of learning sparse representations for first-stage retrieval dates back to SNRM [Zamani et al. 2018] – in the pre-BERT era of neural Information Retrieval. SNRM is based on an encoder ϕ that maps queries and documents into a high-dimensional latent space. At training time, an ℓ_1 regularization is applied to query and document representations, and the model is trained by jointly optimizing ranking and regularization losses:

$$\mathcal{L} = \mathcal{L}_{\text{rank}}(q, d^+, d^-) + \lambda \left(\ell_1(\phi(q)) + \ell_1(\phi(d^+)) + \ell_1(\phi(d^-))\right) \tag{3.10}$$

where $\ell_1(w) = \sum_v |w_v|$ is a smooth, differentiable proxy of the $\ell_0 = \sum_v |w_v|^0$ loss – the true non-differentiable objective to minimize, which corresponds to the number of non-zero elements in w. The strength of the regularization is given by the hyperparameter λ. After training, SNRM's representations are sparse enough to be stored in an inverted index, and retrieval from the collection $\mathcal{C}$ is performed with standard

[94] The sum formulation was part of our first publication [Formal et al. 2021c], but was quickly replaced by the max starting from [Formal et al. 2021a].

query processing techniques. However, the effectiveness of the approach remained quite limited at that time.

In the context of learning sparse image representations in Computer Vision, [Paria et al. 2020] pointed out that minimizing the ℓ_1 norm of high-dimensional representations does not result in the most efficient index, as nothing ensures that posting lists are evenly distributed. This is even more true for standard text-based indexes, due to the Zipfian nature of the term frequency distribution. To obtain a well-balanced index, Paria et al. introduce the FLOPS regularizer, a smooth relaxation of the average number of floating-point operations necessary to compute the dot product between two vectors – hence directly related to the retrieval time. More precisely, let us consider the activation probability p_v for token (or dimension) $v \in V$, and its empirical estimation $\overline{p}_v$ over a batch $\mathcal{B}$ of b documents[95]:

$$\overline{p}_v = \frac{1}{b} \sum_{i=1}^{b} \mathbb{1}[w_v^{(d_i)} \neq 0] \tag{3.11}$$

where $\mathbb{1}[.]$ denotes the indicator function. The mean-FLOPS-per-row, i.e., the expected number of floating-point operations when computing the score between two random document vectors, is given by[96]:

$$\mathcal{F} = \sum_{v \in V} p_v^2 \tag{3.12}$$

It is estimated over a batch by $\overline{\mathcal{F}} = \sum_{v \in V} \overline{p}_v^2$. For a fixed amount of sparsity $\sum_{v \in V} p_v = Vp$, such a quantity is minimized when $p_v = p, \forall v$, i.e., when distributions are *uniform* across posting lists. Intuitively, its goal is to balance each dimension (or token) in the representation space to be equally utilized. In other words, we want to *spread* the representations in the embedding space as much as possible. As $\overline{\mathcal{F}}$ is not continuous, and cannot be optimized with backpropagation, Paria et al. further introduce a smooth relaxation of the (empirical) activation probabilities as $\bar{a}_v = \frac{1}{b} \sum_{i=1}^{b} |w_v^{(d_i)}|$ – which corresponds to the ℓ_1 norm of the activation (scaled by $1/b$)[97]. This gives the following relaxation, estimated over a batch:

$$\tilde{\mathcal{F}} = \ell_{\texttt{FLOPS}} = \sum_{v \in V} \bar{a}_v^2 = \sum_{v \in V} \left(\frac{1}{b} \sum_{i=1}^{b} w_v^{(d_i)} \right)^2 \tag{3.13}$$

As illustrated in Figure 3.5, this differs from the ℓ_1 regularization used in SNRM, where the $\bar{a}_v$ are not squared – as the ℓ_1 estimated over a

[95] Equivalently on the query side.

[96] We refer to [Paria et al. 2020] for the rationale and the complete derivation.

[97] $\bar{a}_v = \frac{1}{b} \sum_{i=1}^{b} w_v^{(d_i)}$ in our case, as SPLADE activations are positive.

batch can be written as:

$$\ell_1 = \frac{1}{b}\sum_{i=1}^{b}\sum_{v\in V} w_v^{(d_i)} \tag{3.14}$$

$$= \sum_{v\in V}\frac{1}{b}\sum_{i=1}^{b} w_v^{(d_i)} = \sum_{v\in V}\bar{a}_v \tag{3.15}$$

Contrary to ℓ_1, $\ell_{\texttt{FLOPS}}$ tends to sparsify the denser dimensions at a faster rate. Paria et al. furthermore show that, in the case of normalized representations $w/||w||$, $\tilde{\mathcal{F}}$ is minimized when vectors are orthogonal – such a loss tends to make the representations fully occupy the space in a uniform manner. It bears a resemblance to a thread of work in CV dedicated to learning image representations that are uniformly distributed over the unit sphere [C. Guo et al. 2019; Sablayrolles et al. 2019][98]. In practice, the non-linear positive activation is crucial to learn sparse vectors: the bias in Eq. 3.1 directly acts as a knob that can turn off specific dimensions.

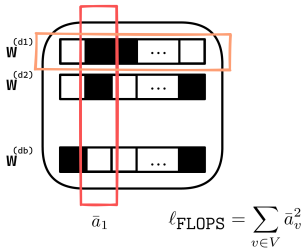


Figure 3.5: Illustration of the FLOPS regularization over a batch $\mathcal{B}$ of samples. The FLOPS operates on columns of the batch of representations (red) – contrary to ℓ_1 that operates on rows (orange).

[98] Before the release of [Paria et al. 2020], we did some preliminary experiments combining ℓ_1 with entropy-based regularization – but $\ell_{\texttt{FLOPS}}$ proved to be simpler and more effective in the end.

Remark

The FLOPS measure in [Paria et al. 2020] estimates the average number of floating point operations needed to compute the score (i.e., the dot product) between a query and a document vector – not to be confused with the FLOPS measure commonly used in Deep Learning, and which corresponds to the total number of floating-point operations performed by a model forward pass.

3.3.4.3 Summing-Up

With SPLADE, we combine the best of both worlds for end-to-end training of sparse, expansion-aware representations of documents and queries. Thus, we discard the binary gating in SparTerm, and instead learn our log-saturated model (Eq. 3.9), by jointly optimizing ranking and regularization losses:

$$\mathcal{L} = \mathcal{L}_{\texttt{rank}} + \lambda_q\mathcal{L}^q_{\texttt{reg}} + \lambda_d\mathcal{L}^d_{\texttt{reg}} \tag{3.16}$$

where $\mathcal{L}_{\texttt{reg}}$ is a sparse regularization (such as ℓ_1 or $\ell_{\texttt{FLOPS}}$). We use two distinct regularization weights (λ_d and λ_q) for queries and documents, allowing us to put more pressure on the query sparsity – a critical aspect for fast retrieval. Following standard practices, we train the model with negative log-likelihood (Section 2.6.5.1). More specifically, given b samples in a batch $\mathcal{B}$, containing i) a query q, ii) a corresponding relevant document d^+, iii) a set of m negative documents $\{d_1^-, ..., d_m^-\}$[99], the ranking loss is given by:

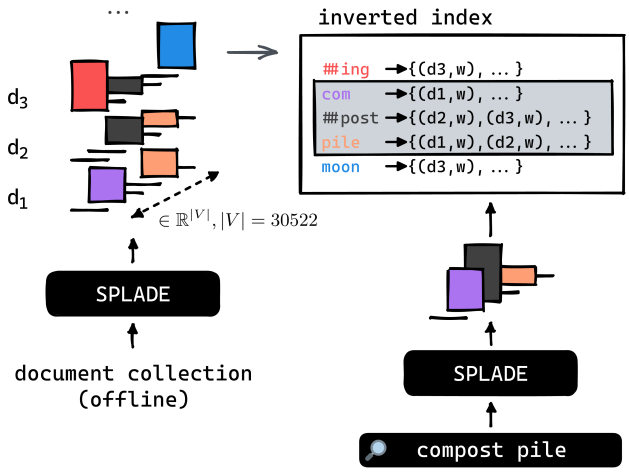
[99] We detail in Section 3.4 how such negatives are usually obtained, and show in Section 3.7 how a careful selection of negatives can improve effectiveness.

$$\mathcal{L}_{\texttt{rank}} = \frac{1}{b}\sum_{\mathcal{B}} -\log(\mathcal{P}(q|d^+)) \tag{3.17}$$

$$= \frac{\exp(s(q,d^+))}{\exp(s(q,d^+)) + \sum_{i=1,\dots,m}\exp(s(q,d_i^-))} \tag{3.18}$$

where the ranking score is given by the dot product between SPLADE representations $s(q,d) = \phi(q)^T\phi(d) = w^{(q)^T}w^{(d)}$. In contrast to SparTerm, SPLADE is trained end-to-end and is remarkably simple. It relies on inverted indexes for fast inference – similar to BM25 – as illustrated in Figure 3.6. Additionally, it avoids resorting to approximate search[100] – contrary to Approximate Nearest Neighbors methods for dense bi-encoders, whose impact on IR metrics has not been fully evaluated yet[101]. Most of the ANN methods for dense retrieval are *unsupervised* and applied after model training. A few works have, however, recently tackled end-to-end optimization of dense encoders and quantization methods. For instance, [Yamada et al. 2021] integrate a learning-to-hash module into DPR [Karpukhin et al. 2020] training to represent passages with compact binary codes rather than continuous vectors. Similarly, [Zhan et al. 2022a] jointly trains the dense encoder and the Product Quantization.

Figure 3.6: Indexing and inference workflow with SPLADE. Sparse document representations can be pre-computed offline and stored in an inverted index. At inference time, the query is fed to the model, and posting lists corresponding to predicted terms are traversed – only shown here for input terms ("*com*", "*##post*", "*pile*") and not expansion.



[100] Although efficient approximate query processing algorithms can further be applied.

[101] Some studies like [Izacard et al. 2021] however showed how dimension reduction or quantization techniques could be used with minimal performance loss on two Question Answering datasets.

3.4 Experimental Setting

In this Section, we describe the experimental setting we follow for training SPLADE (and its extensions) and other baselines such as dense bi-encoders.

Remark

SPLADE is the first sparse bi-encoder designed to learn term-weighting and expansion in an end-to-end manner. It also explicitly optimizes a proxy of efficiency, contrary to most dense approaches that rely on post-processing ANN techniques. In addition, its sparse structure allows it to explicitly account for fine-grained relevance signals such as exact-matching (Sections 3.5.7 and 4.5.3). As shown in Sections 3.5.3 and 3.7, SPLADE is also the first sparse approach to clearly outperform dense models.

Data We train and evaluate our models on the MS MARCO passage ranking dataset introduced in Section 2.3.2.1. We rely on the official MS MARCO training pairs[102], which contain a query q, an annotated positive document d^+, and a (pseudo) negative document d^- sampled from the top-1000 documents retrieved by BM25[103], for query q. We further rely on In-Batch Negatives, a technique widely used to train image retrieval models, that has shown to be effective in learning first-stage rankers [Karpukhin et al. 2020; S.-C. Lin et al. 2021; Qu et al. 2021]. Such a technique artificially augments the number of negatives without adding new documents in the batch, and thus without increasing memory requirements for bi-encoders like SPLADE. Specifically, we augment the number of negatives for a given sample by considering every other positive document in the batch $\mathcal{B}$ as negative. Thus, a training sample becomes $(q, d^+, d^-, d_1^-, ..., d_{|\mathcal{B}-1|}^-)$, and the model can further be optimized with loss 3.17. For evaluation, we use the MS MARCO development set, which contains 6980 queries with shallow annotation. We also evaluate on TREC DL 2019, which contains fine-grained annotations from human assessors for a set of 43 queries. To lighten the analysis, we voluntarily omit results on TREC 2020 – which tend to align with the ones from 2019. We furthermore report significance paired t-tests, using the ranx library [Bassani 2022].

[102] https://microsoft.github.io/msmarco/Datasets.

[103] Note that more negatives could be used. [L. Gao et al. 2021b] show some improvements when training cross-encoders, and we recently noticed performance gains when training neural retrievers.

Training, Indexing and Retrieval Models are trained based on PyTorch [Paszke et al. 2019] and HuggingFace transformers [T. Wolf et al. 2020]. We consider DistilBERT [Sanh et al. 2019] – which contains around $66M$ parameters – to be the default initialization checkpoint for SPLADE models[104]. Section 3.5.6 shows that different architectures can be used as a backbone, as long as they provide the pre-trained MLM head. Models are trained for $150k$ iterations with the ADAM optimizer [Kingma and J. Ba 2015], with a batch size of 124, using a learning rate of $2e^{-5}$ with linear scheduling and a warmup of 6000 steps. We keep the best model checkpoint based on MRR@10 computed on an *approximate* retrieval validation set every $10k$ iteration, similarly to [Hofstätter et al. 2021; Kim et al. 2022]. Indeed, our end goal is to

[104] `distilbert-based-uncased` on HuggingFace. Note that many works (such as SparTerm or EPIC) use BERT as a backbone. As we show in Section 3.5.6, results are usually similar on MS MARCO (in-domain) but tend to degrade with DistilBERT on out-of-domain evaluation (Section 3.8). Nevertheless, models are much faster to train, constituting our base setting.

optimize first-stage retrieval for the collection $\mathcal{C}$. However, indexing the complete collection ($8.8M$ documents) based on the SPLADE encoder $\phi_{(s)}$ at each validation step s is costly, and we need to resort to some approximation. Thus, we first select a set of 1600 queries from the larger set of MS MARCO dev queries – which are not part of the official evaluation dev queries. For each of such queries, we retrieve 100 documents from $\mathcal{C}$ with *i*) BM25 *ii*) and a simple dense bi-encoder model. By considering each unique retrieved document, and adding all the missing relevant ones, we end up with a collection sample $\mathcal{C}_s$ of approximately $300k$ documents, that can be easily indexed in a few minutes at training time, providing us a proxy of the true retrieval performance. We consider a maximum length of 256 for input sequences – larger ones are truncated. This is quite reasonable for MS MARCO passages which are usually shorter – 75 WordPiece tokens in average[105]. To mitigate the contribution of the regularizer at the early stages of training, we follow [Paria et al. 2020] and use a scheduler that quadratically increases $\lambda = (\lambda_q, \lambda_d)$ at each training iteration, until a given step ($50k$ in our case), from which it remains constant[106]. Typical values[107] for λ fall between $1e^{-1}$ and $1e^{-5}$.

Sparse representations allow us to take advantage of inverted indexes and query processing techniques. We use a custom inverted index implementation based on Python arrays and *term-at-a-time* processing, which relies on Numba [Lam et al. 2015] to parallelize retrieval. Contrary to standard software implementations of inverted indexes that usually store compressed integer values for terms such as tf, our index can directly store the float values in SPLADE vectors. We further devise a quantization strategy inspired by DeepCT. At indexing time, we create quantized pseudo-tf for each token (or dimension) v of SPLADE vectors, by simply scaling and rounding: $tf(v) = \texttt{round}(w_v^{(d)} * 100)$. We can thus create a pseudo text collection by repeating, for each document, each term of the WordPiece vocabulary $tf(v)$ times. Afterward, such a collection can be processed, indexed, and retrieved from using standard retrieval software stacks[108]– provided the same treatment is applied to queries. In the end, we have two different ways to perform indexing and retrieval, which, however, lead to the same results – despite the quantization in the latter case.

We conducted experiments with two main regularization losses, ℓ_1 (Eq. 3.14) and $\ell_{\texttt{FLOPS}}$ (Eq. 3.13), which have different effects (see for instance Section 3.5.4). Finally, we rely on 4 Tesla $V100$ GPUs with 32GB memory[109] to train and index models – retrieval can trivially be performed on a single GPU or even on CPU-based environments. Models are trained and evaluated with half-precision (`fp16`) [Micikevicius et al. 2018]. We open-source the code, available at `https://github.com/naver/splade`. We further release model checkpoints on HuggingFace, available at `https://huggingface.co/naver`.

[105] We can actually consider lower sizes, e.g., 128, without hurting the performance. This, however, can have an effect in zero-shot evaluation settings (Section 3.8), where collections might contain longer documents.

[106] Although it remains our default setting, some experiments we recently conducted didn't conclude in a huge effect of the scheduler.

[107] This obviously depends on the range values of $\mathcal{L}_{\texttt{rank}}$. These values are thus given on an indicative basis, for the contrastive loss 3.17.

[108] Such as Anserini [P. Yang et al. 2017], which is built on Lucene. The approach, which requires a few adaptations (such as dot product scoring instead of BM25), is described here: `https://github.com/castorini/anserini/blob/master/docs/regressions-msmarco-passage-distill-splade-max.md`.

[109] However, effective models can also be trained on a single GPU, given appropriate hyperparameters adaptation – we provide such configurations in our repository.

Evaluation We report R@1000 for all datasets, as well as the official metrics MRR@10 and nDCG@10 for MS MARCO dev set and TREC DL, respectively. As SPLADE allows various effectiveness-efficiency trade-offs[110] – depending on the regularization strength λ_q and λ_d – we resort to the FLOPS measure to compare efficiency between different models. It gives an estimation of the average number of floating point operations needed to compute the score (i.e., the dot product) between a query and a document vector. It is defined as the expectation $\mathbb{E}_{q\sim\mathcal{Q},d\sim\mathcal{D}}\left[\sum_{v\in V} p_v^{(q)} p_v^{(d)}\right]$ where p_v is the activation probability for token v in a document d or a query q. It is empirically estimated based on the representations obtained from the complete set of $100k$ MS MARCO dev queries and the full collection $\mathcal{C}$. Overall, the FLOPS gives an indication of *i*) how sparse representations are, *ii*) how balanced the dimensions in query and document representations are. Note that it does not account for the computations needed to obtain the representations (model inference). Other efficiency metrics, closer to practical considerations – such as query latency – could also be used. However, they can be hard to evaluate properly, as different models can rely on different hardware and software stacks (e.g., CPU or GPU, mono-CPU or multi-threaded, etc.). When we compare the efficiency of SPLADE models, the FLOPS is therefore informative enough. We, however, provide in Section 3.5.5 query latency measurements for various SPLADE models.

[110] As already mentioned in Chapter 2, we refer to *effectiveness* (or performance) as IR metrics such as MRR or nDCG. *Efficiency*, in the meantime, refers to how "light" a model can be, for instance, in terms of query latency. See Section 3.5.4.

Remark

The experimental setting introduced here can be seen as the "default" one for training and evaluating SPLADE on MS MARCO. In Section 3.7, we show how SPLADE training can be improved with better sampling strategies or distillation techniques previously shown to be effective for dense bi-encoders. We further extend the evaluation procedure, in Section 3.8 by including zero-shot experiments on the BEIR benchmark.

3.5 Experiments

We present the overall experimental results for SPLADE in Sections 3.5.2 and 3.5.3. When reporting results in Tables, we consider the best performance we can achieve – in terms of MRR@10 on our approximate validation set – *with a FLOPS value inferior to* 3. It thus does not necessarily correspond to the best performance we can obtain, but to a more realistic trade-off between effectiveness and efficiency. We further analyze such trade-offs in Sections 3.5.4 and 3.5.5. We finally provide additional ablations and analyses in respectively Sections 3.5.6 and 3.5.7. Please note that some results reported in this Chapter are

slightly different compared to the ones reported in the corresponding papers – they correspond to different sets of experiments, at different time frames, with slightly different hyperparameters.

> **Remark**
>
> SPLADE results from a long maturation process, spanning months of experimentations and incremental improvements towards the goal of learning effective sparse representations for the IR task. We, therefore, present the results in chronological order, to highlight the research process underlying this thesis.

3.5.1 Overall Results

Compared models Since we are essentially interested in *retrieval* – and not *re-ranking* – we do not consider cross-encoders, and we only compare our approach to first-stage rankers – either dense or sparse[111]. We consider two main families of approaches: *i*) `sparse`, which include models like BM25 [Stephen E. Robertson et al. 1994], doc2query-T5 [R. Nogueira and J. Lin 2019] or DeepImpact [Mallia et al. 2021]; *ii*) `dense` which include models like ANCE [L. Xiong et al. 2021], TAS-B [Hofstätter et al. 2021] or ColBERT [Khattab and Zaharia 2020].

To give an up-to-date picture of the field, we list in Table 3.2 the main approaches that have been proposed in the last three years[112]. We indicate which methods have been proposed after the release of the first SPLADE publication [Formal et al. 2021c] to ensure a fair comparison. We report results from corresponding papers and reproduce some of them. While adding a cross-encoder re-ranker could obviously increase the effectiveness of considered IR systems, we are solely interested in the retrieval step of multi-stage ranking stacks. Thus, gains in Recall are of primary interest in such a setting. However, in the case of a sufficiently effective model, we could imagine bypassing the (costly) re-ranking step – the system would only perform a single *retrieve-and-rank* step. Thus MRR or nDCG at early ranks also constitute our metrics of interest. Note that this way, we also simplify our evaluation process, as the interaction between first-stage and re-ranking models has to be taken into account to properly compare approaches [L. Gao et al. 2021b][113].

Overall, we observe that:

1. *SPLADE and its improvements outperform previous and concurrent sparse retrieval methods by a large margin;*
2. *the results are competitive with state-of-the-art dense retrieval methods.*

We thoroughly discuss the above results in the remainder of this Chapter.

[111] Results reported on the MS MARCO leaderboard (`https://microsoft.github.io/MSMARCO-Passage-Ranking-Submissions/leaderboard/`) – for which top systems rely on a multi-stage ranking pipeline – are thus not entirely comparable to the results presented here.

[112] We acknowledge that some of them might be missing – we tried to identify the most "influential" ones.

[113] In Section 3.7, we, however, rely on cross-encoders for distillation purposes.

Table 3.2: Evaluation on MS MARCO passage retrieval (dev set) and TREC DL 2019. As methods quickly evolved in the field, we denote by ★ models that were not yet released at the time of the first SPLADE publication [Formal et al. 2021c], and by ♣ methods that were published at the same conference (SIGIR'21). For ease of presentation, we multiply values by 100. We use ♢ to refer to various SPLADE models reported in the remainder of this Chapter.

model	MS MARCO dev		TREC DL 2019	
	MRR@10	R@1000	nDCG@10	R@1000
▶ Sparse				
BM25 [Stephen E. Robertson et al. 1994]	18.4	85.3	50.6	74.5
DeepCT [Dai and Callan 2019a]	24.3	91.3	55.1	75.6
doc2query-T5 [R. Nogueira and J. Lin 2019]	27.7	94.7	64.2	82.7
SparTerm [Bai et al. 2020]	27.9	92.5	-	-
UniCOIL (w/ doc2query-T5) [J. Lin and Xueguang Ma 2021] ★	35.2	95.8	70.2	82.9
DeepImpact [Mallia et al. 2021] ♣	32.6	94.8	69.5	-
COIL-tok [L. Gao et al. 2021a] ★	34.1	94.9	66.0	-
COIL-full [L. Gao et al. 2021a] ★	35.5	96.3	70.4	-
SpADE [E. Choi et al. 2022] ★	35.5	96.5	68.2	-
Efficient-SPLADE [Lassance and Clinchant 2022] ★	38.8	98.0	-	-
▶ Dense bi-encoders				
Bi-encoder (ours)	31.2	94.1	63.7	71.1
TCT-ColBERT [S.-C. Lin et al. 2020]	33.5	96.4	67.0	72.0
TCT-ColBERTv2 [S.-C. Lin et al. 2021] ★	35.9	97.0	71.9	76.0
ANCE [L. Xiong et al. 2021]	33.0	95.9	64.8	-
TAS-B [Hofstätter et al. 2021] ♣	34.7	97.8	71.7	84.3
RocketQA [Qu et al. 2021]	37.0	97.9	-	-
RocketQAv2 [Ren et al. 2021b] ★	38.8	98.1	-	-
PAIR [Ren et al. 2021a] ★	37.9	98.2	-	-
ADORE [Zhan et al. 2021] ♣	34.7	-	68.3	-
Condenser [L. Gao and Callan 2021] ★	36.6	97.4	-	-
CoCondenser [L. Gao and Callan 2022] ★	38.2	98.4	-	-
Contriever [Izacard et al. 2022] ★	34.1	97.9	67.6	84.3
AR2 [H. Zhang et al. 2022] ★	39.5	98.6	-	-
▶ Dense multi-vectors				
ColBERT [Khattab and Zaharia 2020]	36.8	96.9	-	-
ColBERTv2 [Santhanam et al. 2022b] ★	39.7	98.4	-	-
▶ Our methods: SPLADE				
♢ SPLADE-sum [Formal et al. 2021c]	32.2	95.5	66.5	81.3
♢ SPLADE [Formal et al. 2021a] ★	34.5	96.5	67.6	81.1
♢ SPLADE-doc [Formal et al. 2021a] ★	32.6	94.0	65.1	71.9
♢ DistilMSE [Formal et al. 2022a] ★	35.8	97.8	72.9	85.9
♢ SelfDistil [Formal et al. 2022a] ★	36.7	98.0	72.6	87.9
♢ EnsembleDistil [Formal et al. 2022a] ★	36.8	97.8	70.7	86.7
♢ CoCondenser-SelfDistil [Formal et al. 2022a] ★	37.6	98.3	73.2	87.7
♢ CoCondenser-EnsembleDistil [Formal et al. 2022a] ★	38.0	98.2	72.7	87.1

Remark

SPLADE evolved in accordance with the field. At the time of its first release, it achieved state-of-the-art results for sparse retrieval. We then adapted the model and the training procedures to keep SPLADE a top competitor in the current landscape of neural IR models.

3.5.2 A First Proposal: SPLADE-sum

Context

This Section summarizes the experimental results from the first SPLADE publication [Formal et al. 2021c], corresponding to SPLADE-sum in Table 3.2.

The design of SPLADE was inspired by SparTerm, and we initially reproduced parts of the results in [Bai et al. 2020]. We thus include a purely lexical SparTerm (i.e., *without* expansion) trained with our pipeline – referred to as SparTerm-lexical. This model is learning query and document term-weighting based on the MLM head, in an end-to-end manner – contrary to models like DeepCT. We also include early experiments, which combined the ideas of SparTerm with the sparse ℓ_1 regularization from SNRM [Zamani et al. 2018]. It corresponds to a "simpler" SparTerm (i.e., without the gating controller), that can also be learned end-to-end. The model is thus equivalent to SPLADE-sum in Eq. 3.8, without the log activation – it is referred to as SparTerm exp. We report detailed results in Table 3.3.

We first observe that our (lexical) SparTerm already outperforms the results of DeepCT as well as the results reported in the original SparTerm paper – including the model using expansion[114]. The model benefits from our training pipeline, which includes – among other things – In-Batch Negatives and larger batch sizes. The sparse expansion mechanism allows bridging the gap with state-of-the-art dense approaches on MS MARCO dev set (e.g., R@1000 close to 0.96 for SparTerm exp (ℓ_1)), at the expense of a larger FLOPS for which representations are not sparse "enough". By adding the log-saturation effect on term importance, SPLADE is able to *i*) increase effectiveness (e.g., around ↑+1 MRR@10), *ii*) greatly increase sparsity – thus reducing FLOPS to similar levels than BoW approaches like doc2query-T5. In addition, we notice the advantage of the $\ell_{\texttt{FLOPS}}$ regularization over ℓ_1 to decrease the computing cost, in terms of the FLOPS measure. We leave to Sections 3.5.4 and 3.5.5 more detailed analyses of efficiency aspects.

[114] Note that when we look at, e.g., MRR@10, improvements become rapidly significant over baselines like BM25, DeepCT or doc2query-T5. In the remainder of this Chapter, we don't report significance tests for those.

Table 3.3: Performance of SPLADE-sum and various additional baselines on MS MARCO (dev) and TREC DL 19. Dense and sparse baselines correspond to state-of-the-art results at the time of publication. We include for sparse models the FLOPS estimation when possible. ^{abcde} denote significant improvements over the corresponding rows, for a paired t-test with p-value=0.01.

model	MS MARCO dev		TREC DL 2019		FLOPS
	MRR@10	R@1000	nDCG@10	R@1000	-
▶ Dense bi-encoders					
Bi-encoder (ours)	31.2	94.1	63.7	71.1	-
ANCE	33.0	95.9	64.8	-	-
TCT-ColBERT	33.5	96.4	67.0	72.0	-
RocketQA	37.0	97.9	-	-	-
▶ Sparse					
BM25	18.4	85.3	50.6	74.5	0.13
DeepCT	24.3	91.3	55.1	75.6	-
doc2query-T5	27.7	94.7	64.2	82.7	0.81
SparTerm	27.9	92.5	-	-	-
▶ Our method: SPLADE					
♢ SPLADE-sum ($\ell_{\texttt{FLOPS}}$) *(a)*	32.2^{cde}	95.5^{c}	66.5^{c}	81.3^{c}	0.73
SPLADE-sum (ℓ_1) *(b)*	32.2^{cde}	95.4^{c}	66.7^{c}	79.2^{c}	0.88
▶ Additional baselines					
SparTerm-lexical *(c)*	29.0	92.3	59.5	77.4	1.84
SparTerm exp (ℓ_1) *(d)*	31.4^{c}	95.9^{c}	66.8^{c}	80.0^{c}	4.62
SparTerm exp ($\ell_{\texttt{FLOPS}}$) *(e)*	31.2^{c}	95.4^{c}	67.1^{c}	81.3^{c}	2.83

Table 3.4: Performance of SPLADE on MS MARCO (dev) and TREC DL 19. DeepImpact [Mallia et al. 2021] and COIL-tok [L. Gao et al. 2021a] represent the most competitive *sparse* baselines at the time of publication. ^{ab} denote significant improvements over the corresponding rows, for a paired t-test with p-value=0.01.

model	MS MARCO dev		TREC DL 2019	
	MRR@10	R@1000	nDCG@10	R@1000
▶ Dense bi-encoders				
Bi-encoder (ours)	31.2	94.1	63.7	71.1
ANCE	33.0	95.9	64.8	-
TCT-ColBERT	33.5	96.4	67.0	72.0
RocketQA	37.0	97.9	-	
▶ Sparse				
DeepImpact	32.6	94.8	69.5	-
COIL-tok	34.1	94.9	66.0	-
♢ SPLADE-sum *(a)*	32.2	95.5	66.5	81.3
♢ SPLADE-max *(b)*	34.5^{a}	96.5^{a}	67.6^{a}	81.1

3.5.3 Towards an Improved Base Model: SPLADE-max

Context

This Section corresponds to (parts of) the SPLADE v2 publication [Formal et al. 2021a], detailing the overall improved model architecture – referred to as SPLADE in Table 3.2.

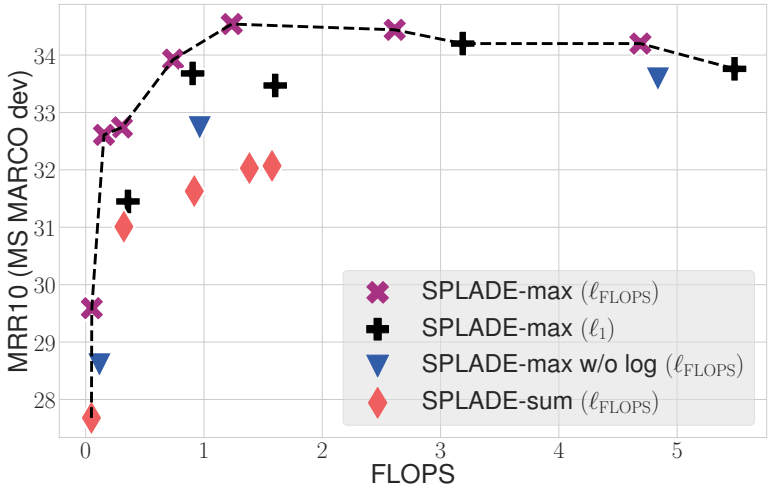
The sum aggregation in SPLADE (Eq. 3.8) was initially borrowed from SparTerm. By drawing some parallels with EPIC and SPARTA, and to some extent ColBERT, we experimented with max-pooling instead (Eq. 3.9). Such a mechanism allows the selection, for every dimension v in the vocabulary space, of the most salient contribution among input tokens. Additionally, the magnitude of the weights becomes more invariant to the input length. This simple modification already leads to significant boosts in effectiveness – around ↑+2.3 MRR@10 and ↑+1 R@1000, see Table 3.4. Improvements are also statistically significant. SPLADE outperforms models like DeepImpact and becomes competitive with approaches like COIL-tok – for a much lower indexing and inference cost. We have also experimented with other simple aggregation strategies, which didn't perform as well. For instance, taking the average instead of the sum – similar to what is done for dense bi-encoders – leads to very poor results. This might be due to the length penalty being too strong, especially given the log activations.

3.5.4 The Effectiveness-Efficiency Trade-Off

The IR community has been thinking about the effectiveness-efficiency tradeoffs for many decades [L. Wang et al. 2010]. It has recently received renewed attraction due to the increasing size of models based on Pre-trained Language Models [Santhanam et al. 2022c]. Complex methods – such as cross-encoders based on BERT – perform extremely well but run orders of magnitude more slowly than term-based approaches. To reduce the time required to present query results to users, one usually has to sacrifice performance.

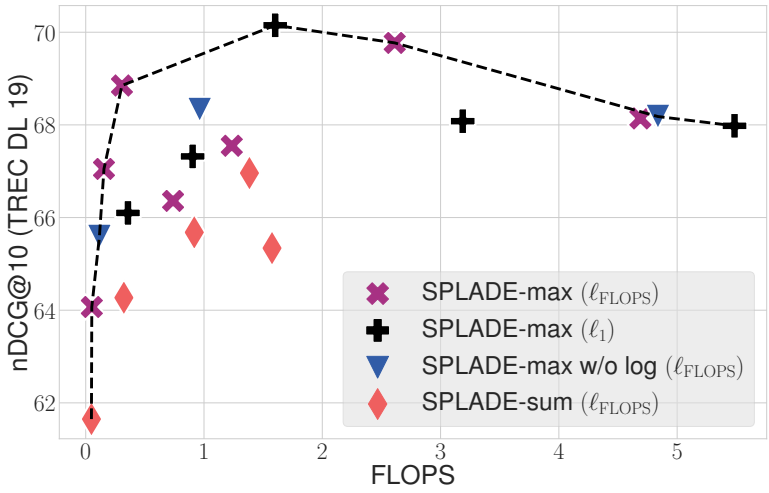
By jointly optimizing both ranking and regularization losses (Eq. 3.16), SPLADE can inherently adjust such a trade-off, by means of the $\lambda = (\lambda_q, \lambda_d)$ hyperparameter. Thus, SPLADE explicitly encompasses efficiency as part of its training – contrary to standard dense bi-encoders, where ANN algorithms are usually applied in an unsupervised manner once the models are trained. Increasing λ generally pushes models to learn sparser representations – therefore leading to more efficient models – usually at the expense of a performance loss. Up to now, we have shown single performance points for different SPLADE models (in Table 3.3 and Table 3.4), that correspond to the best performance we can obtain, and which usually correspond to lower λ (i.e., "denser" models). Figure 3.7 illustrates such a trade-off between effectiveness (MRR@10 on MS MARCO) and efficiency (as measured by FLOPS), when we vary λ_q and λ_d, for different versions of SPLADE. In the remainder of this Chapter, we plot the Pareto front (dotted line) for each trade-off plot.

Figure 3.7: Effectiveness-Efficiency trade-off on MS MARCO dev set (MRR@10). Models on the left correspond to high $\lambda = (\lambda_q, \lambda_d)$.



Overall, MRR@10 tends to increase as the strength of the regularization λ diminishes – leading to higher FLOPS. On the contrary, if λ is high, models must learn compact representations by selecting a handful of terms in the embedding space. If the compression becomes too im-

Figure 3.8: Effectiveness-Efficiency trade-off on TREC DL 2019 (nDCG@10).



portant, some key terms might be dropped, thus hurting performance to some extent. Note that expansion saturates at some point: further decreasing λ (leading to higher FLOPS, not shown here) does not lead to better performance and usually even worsens the results. As such, the most effective model (MRR@10 = 34.5) is obtained for a middle-valued λ (FLOPS=1.23). Interestingly, strongly regularized models still show competitive performance. For instance, a model resulting in an average query (resp. document) length $\ell_0(q) = 11$ (resp. $\ell_0(d) = 33$) reaches 32.6 MRR@10 – results on par with models like DeepImpact, which relies on heavily expanded passages [115]. When comparing different versions of the model, we first observe that SPLADE trained with $\ell_{\texttt{FLOPS}}$ defines the Pareto front in Figure 3.7. The regularization effect brought by $\ell_{\texttt{FLOPS}}$ compared to ℓ_1 is clear: for the same level of efficiency, the performance of the latter is always lower. In addition to optimizing efficiency, $\ell_{\texttt{FLOPS}}$ implicitly adds an entropy-based regularization, which seems to be beneficial to learn effective representations. We also notice the beneficial effect of the log-saturation effect – boosting both effectiveness and efficiency. Finally, it is pretty clear how SPLADE-sum (Section 3.5.2) lags behind its max counterpart.

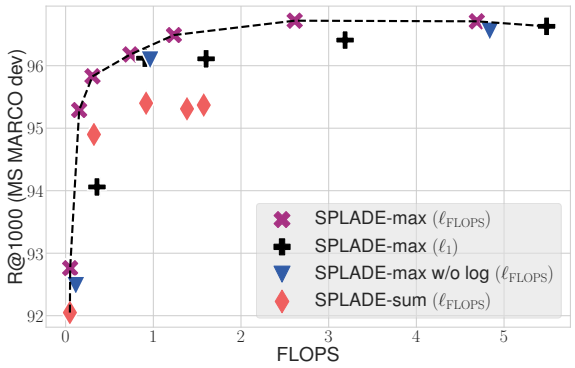
We additionally show on Figure 3.8 the same trade-off on TREC DL 19 for nDCG@10, for which results are more contrasted[116]. Figures 3.9a and 3.9b show the same trade-off, for respectively R@1000 on MS MARCO dev and TREC DL 19. For the former, we observe the same overall trends. On TREC DL 19, results are however again more contrasted – aligned with the observations made on Figure 3.8.

Finally, we plot in Figure 3.10 the sorted number of passages in each posting list, for different models – for the 8.8M MS MARCO passages. We first observe that a SPLADE model trained *without* regularization is still somewhat sparse[117], but has overall much larger posting lists

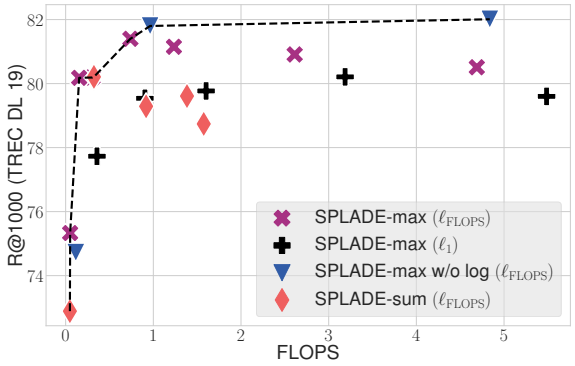
[115] Note that for comparison, $\ell_0(d) = 75$ for original documents tokenized with WordPiece.

[116] The low number of queries (43) in TREC is responsible for relatively unstable results. For instance, as shown across the Chapter, it is usually difficult to obtain significant results on DL 19.

[117] Otherwise, posting lists would have a constant size of $|\mathcal{C}|$.



(a) MS MARCO dev.



(b) TREC DL 19.

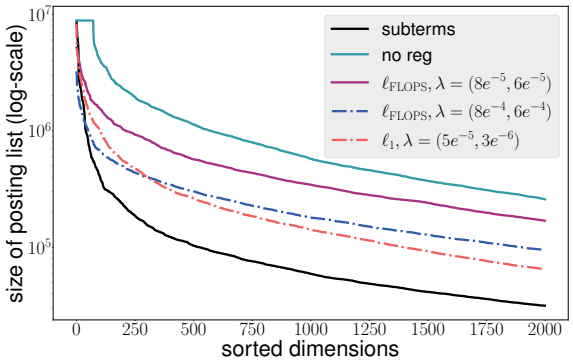
Figure 3.9: Effectiveness-Efficiency trade-off for R@1000.

– increasing index size and latency. On the other hand, we provide the same distribution for original text documents, when tokenized with WordPiece[118] – which has a Zipfian profile. The blue and red dotted lines correspond to two SPLADE models with similar document sparsity $\ell_0(d)$, trained with respectively $\ell_{\texttt{FLOPS}}$ and ℓ_1. We observe that the FLOPS tends to distribute the posting lists more evenly – thus acting as an entropy regularizer. We also see how decreasing λ increases the size of posting lists overall (solid purple line).

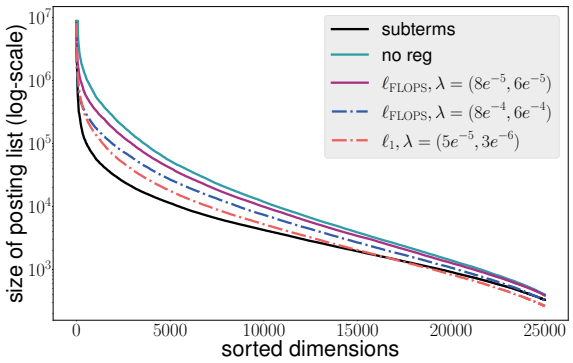
[118] Such an index was built for a BM25 model based on WordPiece tokens – instead of actual terms – for which effectiveness only decreases a bit (MRR@10=17.5 instead of 18.4).

3.5.5 Evaluating Query Latency

Up to now, we have characterized SPLADE efficiency by the sparsity of its learned representations (or its FLOPS). In Information Retrieval, we are usually interested in measuring query latency – the average time needed to serve a query. Such a measure takes into account practical considerations and is directly linked to user satisfaction: large production systems need to provide results under the *millisecond* for millions of users. We first discuss several challenges when it comes to measuring query latency for learned sparse models in Section 3.5.5.1. We then evaluate SPLADE efficiency in Section 3.5.5.2.



(a) $2k$ dimensions.



(b) $25k$ dimensions.

Figure 3.10: Index distribution for subterm-based representations. We see the regularization effect on the distribution: the lower the λ, the "denser" the posting lists. Note that around $2k$ (out of the $30k$ of the WordPiece vocabulary) dimensions are actually not used by the model (i.e., appear in no document representations.) – not shown here. These usually correspond to rare or unseen terms on MS MARCO, as well as other special characters of the BERT tokenizer. For SPLADE models, large posting lists also correspond to frequent terms: "*is*", "*do*" and "*the*", with respectively $3.2M$, $2.9M$ and $2.4M$ documents for the blue line.

3.5.5.1 Challenges in Measuring Query Latency for Learned Sparse Models

The FLOPS regularization introduced in [Paria et al. 2020] (Section 3.3.4.2) was motivated by the implied efficiency of sparse representations coupled with inverted indexes, that usually do not require approximations[119] – contrary to ANN techniques for dense representations. In their setting, retrieval is cast as a sparse matrix multiplication between query and document vectors, that suppose *intra-query* parallelism, i.e., that posting lists can be traversed in parallel in a multi-threaded setting. Thus, the FLOPS measure is directly related to retrieval time. By relying on the same regularization, SPLADE is also optimized for multi-threaded retrieval.

[119] Although, many efficient query processing algorithms do rely on approximations.

However, the standard retrieval software stacks usually only assume *inter-query* parallelism: as such, a single query is processed by a sin-

gle CPU core[120]. In this setting, increasing the number of posting lists to traverse at inference time usually leads to huge efficiency drops, which is particularly hurtful for query expansion methods – including SPLADE. [Lassance and Clinchant 2022] propose several techniques to overcome such issues for SPLADE models. Moreover, document expansion can also impact efficiency, as it increases the size of the posting lists. One subtle additional point is the distribution of weights in the posting themselves. Standard query processing algorithms such as MaxScore [Turtle and Flood 1995] or WAND [Broder et al. 2003] assume a certain weight distribution, derived from traditional term-based scoring functions such as BM25 and the statistical properties of natural language. The new sparse approaches based on PLM can therefore learn term distributions for which standard algorithms are not suited. For instance, [Mallia et al. 2021] show how query latency is significantly higher for DeepImpact, despite the absence of query expansion. [Mackenzie et al. 2021] show how SPLADE generates "wacky" weights, that reduce the opportunities for skipping and early exiting optimizations that lie at the core of standard *document-at-a-time* techniques. Indeed, traditional models like BM25 assume that frequent terms have low importance scores (IDF effect) – which is not necessarily the case for learned sparse models (see Section 3.5.7). As a consequence, a relatively new research branch emerged, with the goal of speeding up learned sparse retrieval. [Mallia et al. 2022] propose to guide the traversal of (learned) posting lists with BM25. [Mackenzie et al. 2022a] recently introduced *postings clipping* – adapting MaxScore and WAND for learned importance schemes[121].

[120] The rationale is that production systems should deal with many queries concurrently – the bottleneck is thus not within a single query.

[121] They, however, do not consider SPLADE as part of the study.

3.5.5.2 Measuring SPLADE Latency

Generally speaking, measuring and comparing the latency of retrieval systems is intricate, as different approaches rely on different assumptions. As such, dense retrieval usually assumes having access to multiple CPUs – or even GPUs – to perform search. In the meantime, sparse retrieval based on inverted indexes is usually based on mono-core implementations (as discussed in Section 3.5.5.1). We propose to measure the efficiency of our systems in two settings. First, we rely on Anserini [P. Yang et al. 2017], an IR toolkit built on the open-source Lucene search library[122]. To index our collection, we follow the quantization approach described in Section 3.4. We keep the Anserini default setting, which relies on BMW [Ding and Suel 2011] (*document-at-a-time*) query evaluation. To measure latency with Anserini, we place ourselves in a mono-CPU setting (by setting the option `-parallelism 1`). Additionally, we propose to evaluate latency based on our custom inverted index, which relies on a multi-threaded Numba *term-at-a-time* query processing – thus being more aligned with SPLADE training objective. Queries are still processed sequentially, but we benefit from

[122] Note that more efficient search libraries – such as PISA [Mallia et al. 2019] – are available to the research community.

intra-query parallelism – which is not supported by Anserini. We conduct experiments on a CPU-based environment, using a server with 64 2GHz Intel(R) Xeon(R) Gold 6338 CPUs and 256GB of RAM. We use 1 and 12 CPU cores to measure latency with Anserini and our index, respectively. We evaluated the mean query latency on the set of 6980 dev queries.

We report in Table 3.5 effectiveness, as well as several efficiency measures, for a set of SPLADE models trained with varying λ – corresponding to the ones in Figure 3.7. Note that we only evaluate retrieval time, which does not account for the time needed to compute the query representation with DistilBERT[123]. *Overall, we see that latency decreases with sparsity (and the FLOPS measure).* More specifically, latency measurements on a mono-CPU environment (with Anserini) are in line with the ones reported in [Mackenzie et al. 2021]. They are especially large for less regularized SPLADE models (e.g., 24954ms for a FLOPS value of 4.7). However, for a sufficiently large λ, we are able to reach efficiency levels similar to BM25 – e.g., 69.3ms for a model with almost 30 MRR@10. When switching to a multi-threaded setting based on Numba parallelism, query latency largely decreases – reaching efficiency levels on par with BM25 (Anserini) for the most effective models (44.1ms for 34.5 MRR@10). We also provide Anserini index sizes which highlight the effect of regularization: the higher the λ, the smaller the memory requirements for the index[124]. While we do not evaluate latency for such a model, it is interesting to note that SPLADE trained without regularization already leads to sparse solutions (e.g., $\ell_0(d) = 326$), due to the log activation – as already observed in Section 3.5.4. Note that generally, both query and document sizes increase with the FLOPS. In theory, one could decrease the FLOPS by largely increasing query sizes and decreasing document sizes, which would be more detrimental for mono-core evaluation settings.

[123] Which can be of the order of 10ms on a single GPU [Hofstätter et al. 2021].

[124] Note that a standard flat index obtained with FAISS [Johnson et al. 2019] for a dense bi-encoder weighs around 22GB for the whole MS MARCO collection.

3.5.6 The Impact of the MLM Head

This Section focuses on the central aspect of the projection layer for SPLADE. Section 3.5.6.1 discusses our earliest attempts to learn effective sparse representations for IR – at a time when dense approaches were on a roll. Section 3.5.6.2 provides experimental results for various Pre-trained Language Models.

3.5.6.1 Background

Prior to SPLADE, we made several attempts to learn sparse bi-encoders for first-stage retrieval. One such idea consisted in adapting SNRM [Zamani et al. 2018] with PLM, by trying to map BERT contextualized embeddings to a high-dimensional sparse latent space. The overall main difference w.r.t. SPLADE is that the projection matrix in Eq. 3.1 was

model	MRR@10	$\ell_0(q)$	$\ell_0(d)$	FLOPS	Latency (ms)		Index Size (GB)
					Anserini	Numba	
BM25	18.4	-	-	0.13	47.8	-	0.6
DeepImpact	32.5	-	-	-	244.1	-	1.4
`no reg`	34.1	90	326	75.47	-	-	-
(λ_q, λ_d)							
$(8e-5, 6e-5)$	34.2	22	166	4.69	2494.3	71.5	3.4
$(3e-4, 1e-4)$	34.4	19	148	2.61	1434.2	58.7	3.2
$(8e-4, 6e-4)$ ♢	34.5	16	90	1.23	828.37	44.1	2.3
$(3e-3, 1e-3)$	33.9	15	74	0.74	486.53	38.0	2.1
$(8e-3, 6e-3)$	32.7	14	44	0.31	286.96	31.2	1.6
$(3e-2, 1e-2)$	32.6	11	33	0.15	119.20	19.5	1.4
$(8e-2, 6e-2)$	29.6	9	18	0.05	69.34	16.1	1.1

Table 3.5: Experimental results on MS MARCO dev set. $\ell_0(q)$ and $\ell_0(d)$ denote the average query and document sizes, estimated respectively from the 6980 dev queries and the full collection $\mathcal{C}$ after training. We measure latency (in *milliseconds*) for BM25 ($47.85ms$ in the same evaluation environment), and report numbers for DeepImpact from [Mackenzie et al. 2021] for comparison – acknowledging the differences between experimental settings.

learned *from scratch*[125]. Dimensions, in that case, are latent and cannot be interpreted as actual terms. Models were difficult to train, and, overall, not so effective. Learning sparse components in such a high dimensional space might be too challenging, whereas formulating the same problem with lexical matching and document expansion (i.e., in the term space) may lead to a simpler optimization problem. Such a setting also motivated our use of the FLOPS regularization (instead of ℓ_1), as models tended to simply learn a smaller dense subspace of the representation space – leading to "falsely" sparse representations[126]. While this issue is greatly mitigated when working in the term space, the implicit entropy regularization from $\ell_{\texttt{FLOPS}}$ turned out to produce more effective representations, as shown in Section 3.5.4. We won't present such results in this thesis – some of them have been lost – but they were anyway far less effective than SPLADE. The pre-conditioning of the projection matrix, based on the MLM head that is tied to the vocabulary, was the main ingredient to make such an approach *i*) effective and *ii*) easy to train.

[125] The model also did not use log activation, among other details.

[126] This problem has been reported by several teams trying to reproduce the original SNRM – including ourselves.

3.5.6.2 Initializing SPLADE

So far, our experiments have been based on a DistilBERT backbone model ($66M$ parameters, $|V| = 30k$). We provide additional experiments in Table 3.6, for which we change the base PLM model, including *i*) BERT ($110M$ parameters, $|V| = 30k$) *ii*) RoBERTa ($125M$ parameters, $|V| = 50k$) *iii*) ELECTRA ($33M$ parameters, $|V| = 30k$).

Table 3.6: Performance of SPLADE on MS MARCO (dev) and TREC DL 19 for various PLM initializations. ^{abcde} denote significant improvements over the corresponding rows, for a paired t-test with p-value=0.01. Note that none of the improvements are significant on TREC DL 19.

model	MS MARCO dev		TREC DL 2019	
	MRR@10	R@1000	nDCG@10	R@1000
♢ SPLADE *(a)*	34.5^{bcd}	96.5	67.6	81.1
SPLADE (RoBERTa) *(b)*	30.9	96.0	67.2	79.4
SPLADE (ELECTRA) *(c)*	33.0^{b}	96.7^{b}	67.1	82.4
SPLADE (BERT-base) *(d)*	33.7^{b}	96.7^{b}	70.0	81.1
MLM FLOPS *(e)*	35.7^{abcd}	97.2^{abcd}	67.9	80.2

We first observe that SPLADE is not able to take advantage of the RoBERTa checkpoint. The models are harder to train, and sometimes even diverge. This might be due to several factors, including the different vocabulary (BPE) or the ill-conditioned distribution of the MLM logits for SPLADE as noted in [Nair et al. 2022]. Also note that RoBERTa is case-sensitive, which might hurt to some extent retrieval performance[127].

[127] We, however, didn't explore further the roots of the problem.

SPLADE based on the ELECTRA generator, with a number of parameters twice lower as the base SPLADE, already outperforms models like DeepImpact. Finally, increasing the number of parameters (by switching from DistilBERT to BERT) doesn't seem to help on MS MARCO – with performance actually lower overall, which might be caused by a slight overfitting[128]. We further show in Figure 3.11 the same type of trade-off shown in earlier Sections – obtained by training models with varying λ.

[128] We show, however, in Section 3.8 that larger checkpoints are usually helpful for zero-shot transfer.

As we aim to learn sparse representations, the distribution of the MLM weights at initialization plays a key role in both the convergence and performance of SPLADE. We have recently experimented with an intermediate unsupervised pre-training step (or middle-training, Section 2.6.5.2) – i.e., before fine-tuning – which trains the PLM with a combination of the Masked Language Modeling task (Section 2.6.5.1) and the FLOPS regularization (Eq. 3.13) on the logits of the MLM head – dubbed MLM-FLOPS. More specifically, the MLM logits first go through the SPLADE activation function (i.e., $\log(1+\text{ReLU}(w_{iv}))$), and are fed to a softmax which defines an MLM loss over *sparse* logits. The FLOPS regularization is applied to sequence-level representations obtained with max-pooling – similar to SPLADE – to achieve sparsity. The total loss is simply given by:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{MLM-SPLADE}} + \lambda \ell_{\texttt{FLOPS}} \tag{3.19}$$

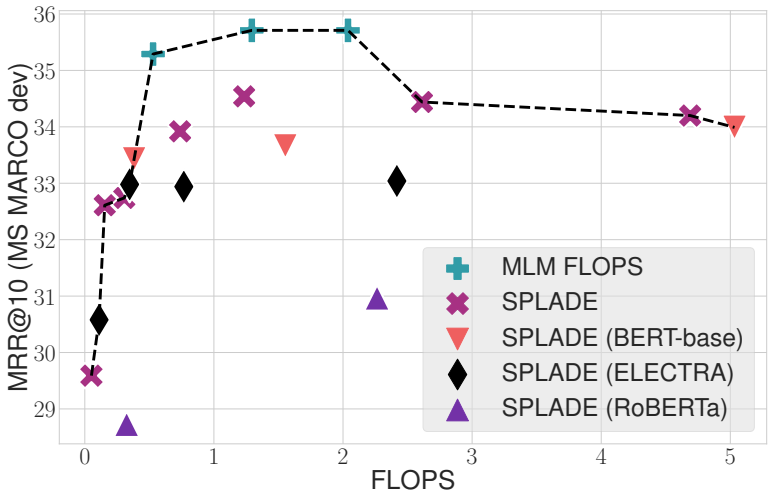
Models are trained for 10 epochs on MS MARCO passages, with $\lambda = 0.001$. Thus, representations are already pre-conditioned to be sparse,

which helps SPLADE converge, leading to significant performance boosts (e.g., ↑+1.2 MRR@10).

Remark

While MLM FLOPS pre-training is beneficial for SPLADE, all the experiments presented in the remainder of this Chapter are actually based on standard checkpoints like DistilBERT – as they correspond to earlier experiments. [Lassance and Clinchant 2022] relied on such a sparse pre-conditioning to improve SPLADE efficiency.

Figure 3.11: Effectiveness-Efficiency trade-off on MS MARCO dev set for different base Pre-trained Language Models.



3.5.7 Interpretable Representations

By design, SPLADE BoW representations are grounded in the vocabulary. This is ensured by the fact that the weights of the output projection of the MLM head are tied to the input embeddings of the PLM. Thus, each dimension of the representation space actually corresponds to a token of the vocabulary. This results in a valuable byproduct: representations become *interpretable*, and we can – to some extent – provide explanations for the observed rankings. For instance, it is possible to identify terms with the highest weights in a document vector – those contribute the most to the relevance of the document to a given query[129]. Such a property can also be useful for failure analysis, as it becomes easier to understand the causes of low-performing queries. Dense representations – for which the learned latent space is more opaque – do not enjoy such direct benefits. We provide in Table 3.7 and Table 3.8

[129] While we haven't explicitly worked on explainability aspects of SPLADE, we acknowledge its benefit for Information Retrieval, as users, as well as political authorities, tend to require more transparency from search and recommendation systems.

Table 3.7: SPLADE BoW prediction for six TREC DL 2019 queries. We highlight in color the predictions for terms in the input (*lexical* part), and strike-through terms which are not in the final presentation, i.e., terms whose weights are zero (*compression*). Other terms correspond to *expansion*. We selected a relatively sparse model ($\ell_0(q) = 11$ and $\ell_0(d) = 33$) – see Table 3.5.

SPLADE BoW predictions
`query` (*id*: 1127893) → "*ben foster footballer net worth*" ► (foster, 2.65), (ben, 1.89), (worth, 1.78), (footballer, 1.39), (wealth, 0.96), (player, 0.63), (football, 0.56), (benjamin, 0.37), (net, 0.24)
`query` (*id*: 478605) → "*port orange ~~what~~ county*" ► `BoW` → (orange, 2.48), (port, 2.24), (county, 1.67), (counties, 0.03)
`query` (*id*: 1124145) → "*truncating meaning*" ► (tr, 2.26), (##cating, 2.24), (##un, 1.89), (##cation, 1.53), (##cate, 1.43), (##cated, 1.09), (meaning, 0.49), (., 0.23), (##cy, 0.13)
`query` (*id*: 646207) → "*~~what~~ ~~does~~ production design entail*" ► (production, 2.46), (design, 2.18), (en, 1.85), (##tail, 1.79), (producer, 1.03), (designer, 0.77), (designed, 0.68), (factory, 0.18), (product, 0.05)
`query` (*id*: 1037798) → "*who ~~is~~ robert gray*" ► (gray, 2.61), (robert, 2.31), (grey, 1.59), (who, 1.04), (strange, 0.8), (he, 0.73), (kent, 0.25), (bobby, 0.09)
`query` (*id*: 104861) → "*cost ~~of~~ interior concrete flooring*" ► (concrete, 2.16), (interior, 2.11), (floor, 1.83), ($, 1.26), (cost, 1.01), (cement, 0.73), (inside, 0.64), (price, 0.62), (internal, 0.14)

examples of predictions for queries and documents, respectively. We observe four main effects:

- **Term-weighting** Tokens in the input are given weights according to their estimated *contextual* importance – usually high for what seems to be "important" tokens. The effect of context can also be observed: for instance, the term "*concrete*" (Table 3.8) is given a high score (2.08) for a document about floor materials, while it has a low score (0.03) in the last document about another topic. We also notice that some apparently non-important tokens actually have high weights (e.g., "*who*" for the query "*who is robert gray*") – Table 3.7. Pre-trained Language Models are therefore not only useful to bridge the vocabulary gap but also better estimate the importance of terms in IR models. Overall, such a mechanism conditions the model to rely on *exact matching* (see Section 4.5.3) – which might be harder to learn for dense bi-encoders.

- **Compression** Some input tokens are discarded – the model predicting a zero weight for this dimension. This effect is due to the sparsity regularization, which identifies irrelevant tokens to

remove. This is usually the case for stopwords, similar to standard practices in term-based approaches – but not only. Thus, we see for the first document in Table 3.8, the birth date ("*May 10, 1755*") is actually deemed non-important by the model. This effect is also stronger on documents, as removing terms from queries – which are already rather short – is riskier and can greatly impact performance.

- **Expansion** Tokens not in the original input are predicted non-zero weights. This usually corresponds to related tokens, such as synonyms, that allow bridging the vocabulary gap between queries and documents. For instance, the tokens "*price*" or "*cement*" are both expanded on the last query in Table 3.7, and the corresponding relevant document in Table 3.8. Note that expansion is not always accurate: we see how tokens like "*grey*" or "*bobby*" are wrongly predicted for the query "*who is robert gray*". This is a side effect of expansion, which happens to predict "wacky" weights [Mackenzie et al. 2021].

- **Stemming** This is a specific side effect of expansion, where linguistic variations of a token are predicted – as a result of the WordPiece-based representation space. Stemming – which has a long-standing history in Information Retrieval – is thus (partially) taken into account in the model, and *learned* at training time. It is shown in the third row of Table 3.7, where various suffixes of "*trun*" are predicted by SPLADE ("*truncating*"→ "*truncation*","*truncate*","*truncated*").

Remark

We see from previous examples that SPLADE has a strong lexical prior, by predicting weights for *important* query or document terms. We show in Chapter 4 that this prior has a positive effect for generalization [Formal et al. 2022b].

3.6 Towards Improving SPLADE Efficiency

We provide in this Section several SPLADE extensions, that aim to improve its *efficiency*.

3.6.1 Models

As already discussed in Section 3.5.5, efficiency is a critical aspect of IR systems, and reducing measures such as query latency can greatly improve the "usability" of models for production scenarios. Minimizing

Table 3.8: SPLADE BoW prediction for three documents. The first two are respectively *relevant* for the two last queries in Table 3.7. Compared to queries, we see how documents are far more *compressed*, due to the regularization constraint. We use the same model as the one for Table 3.7.

SPLADE BoW predictions
`document` (*id*: 8760871) → "*Atlantic Ocean, United States. Robert Gray, (born ~~May 10, 1755~~, Tiverton, ~~R.I.~~ died ~~summer 1806 at~~ sea ~~near eastern U.S.~~ coast) captain ~~of the~~ first ~~U.S.~~ ship ~~to~~ ~~circumna~~vigate ~~the~~ globe ~~and~~ explorer ~~of the~~ Columbia River.*" ► (gray, 2.55), (robert, 2.19), (atlantic, 2.17), (ocean, 1.74), (grey, 1.73), (##vert, 1.59), (globe, 1.53), (first, 1.4), (captain, 1.37), (who, 1.31), (ship, 1.25), (died, 1.2), (explorer, 1.12), (columbia, 1.0), (born, 1.0), (sea, 0.96), (he, 0.77), (##vi, 0.66), (states, 0.65), (death, 0.64), (##rc, 0.63), (discovery, 0.59), (commanded, 0.55), (##gate, 0.45), (served, 0.44), (strange, 0.4), (river, 0.39), (coast, 0.38), (invented, 0.36), (united, 0.34), (us, 0.24), (boat, 0.22), (ti, 0.21), (american, 0.16), (##on, 0.06), (deceased, 0.02)
`document` (*id*: 1017092) → "*~~Some things that may~~ add ~~to that~~ cost ~~are:~~ site ~~and~~ sub-base preparation, site ~~access~~, small ~~floors under 500 sq. ft., and thicker~~ concrete. Integral ~~colored~~ concrete floor cost: ~~$3.75 dollars~~ per square foot, ~~that includes the basic~~ concrete floor package ~~and adding~~ color ~~to the~~ concrete.*" ► ($, 2.13), (concrete, 2.08), (integral, 2.02), (floor, 1.89), (cost, 1.83), (add, 1.43), (square, 1.38), (color, 1.22), (small, 1.16), (foot, 1.06), (package, 1.01), (price, 0.97), (per, 0.82), (sub, 0.73), (paint, 0.7), (site, 0.67), (cement, 0.64), (preparation, 0.46), (thickness, 0.31), (thick, 0.21), (base, 0.07), (building, 0.06), (large, 0.01)
`document` (*id*: 8530) → "*An hypothesis ~~is a~~ specific statement ~~of~~ prediction. ~~It describes~~ in concrete ~~(rather than~~ theoretical) terms ~~what you~~ expect ~~will happen~~ in ~~your~~ study. ~~Not all~~ studies ~~have~~ hypotheses. ~~Sometimes a~~ study ~~is~~ designed ~~to be~~ ~~exp~~loratory ~~(see~~ ~~inductive~~ research). ~~There is no~~ formal hypothesis, ~~and perhaps the~~ purpose ~~of the~~ study ~~is to explore some area more thoroughly~~ in ~~order to develop some~~ specific hypothesis ~~or~~ prediction ~~that can be tested~~ in ~~future~~ research.*" ► (hypothesis, 2.64), (theory, 1.85), (study, 1.83), (prediction, 1.75), (statement, 1.59), (predict, 1.33), (##oth, 1.21), (specific, 1.07), (research, 1.03), (candidate, 0.92), (,, 0.9), (##yp, 0.88), (science, 0.86), (an, 0.86), (expect, 0.78), (in, 0.76), (##ora, 0.69), (purpose, 0.67), (studies, 0.65), (theoretical, 0.6), (##uc, 0.37), (mathematical, 0.37), (scientific, 0.36), (formal, 0.35), (##tory, 0.34), (designed, 0.25), (predicted, 0.2), (contrast, 0.18), (expected, 0.18), (ind, 0.12), (h, 0.11), (assumptions, 0.08), (##eses, 0.07), (terms, 0.04), (concrete, 0.03), (describing, 0.01)

Context

This Section corresponds to (parts of) the SPLADE v2 publication [Formal et al. 2021a] (SPLADE-doc model), as well as several additional experiments. While we tackle the efficiency aspects of SPLADE, we only scratch the surface. [Lassance and Clinchant 2022] further push such considerations in their efficiency study of SPLADE models.

$\ell_{\texttt{FLOPS}}$ (or ℓ_1) only optimizes for a certain aspect of efficiency, that is not entirely aligned with what makes a model truly efficient. In particular, *i*) standard IR software stacks do not usually perform intra-query parallelism, and thus require small queries to process a low number of posting lists (Section 3.5.5.1); *ii*) query inference time, i.e., the time needed to compute query representations – based on Pre-trained Language Models or not – should be low. We propose to deal with both aspects in the following.

Lexical SPLADE While adding more tokens to queries (i.e., allowing more expansion by lowering λ_q) usually leads to better results for SPLADE (e.g., Figure 3.7), it can have a huge impact on query latency for systems relying on mono-CPU processing (cf Table 3.5). Such an issue can be addressed by either *i*) enforcing more sparsity for queries, *ii*) or more simply turning-off expansion on the query side.

For the former, we simply propose to regularize query representations with ℓ_1 instead of $\ell_{\texttt{FLOPS}}$. Indeed, while $\ell_{\texttt{FLOPS}}$ leads to more distributed representations (Figure 3.10), ℓ_1 however has a stronger "sparsity" effect. Thus, we propose to optimize the model – referred to as SPLADE-$\ell_1(q)$ – with:

$$\mathcal{L} = \mathcal{L}_{\texttt{rank}} + \lambda_q \ell_1^q + \lambda_d \ell_{\texttt{FLOPS}}^d \tag{3.20}$$

For the latter, we propose a simple extension – that we refer to as SPLADE-lexical – that only performs document expansion. Query terms are still weighted based on the PLM, but we apply a binary BoW mask on query representations – both at training and inference time – similarly to the gating controller in SparTerm. Thus, query vectors are inherently sparse, as the only non-zero dimensions correspond to query tokens, as shown in Figure 3.12.

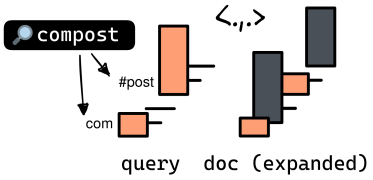


Figure 3.12: SPLADE-lexical. Inference is still performed on the query side, but a binary BoW mask is applied, to only keep dimensions corresponding to input query tokens.

SPLADE-doc While the previous extensions allow reducing query evaluation time by lowering the number of posting lists to traverse, they are still limited by the inference time of the PLM needed to get the query representation. While one can imagine considering more efficient query encoders[130], we actually propose to consider a document-only version of SPLADE – referred to as SPLADE-doc. In this case, there

[130] Such as MiniLM [W. Wang et al. 2020] with $33M$ parameters.

is neither query term weighting nor expansion, and the ranking score is simply given by:

$$s(q,d) = \sum_{v \in q} w_v^{(d)} \tag{3.21}$$

Here, we can see the query as a sparse binary BoW vector, as shown in Figure 3.13. Such a formulation is akin to traditional term-based approaches in IR, and offers an interesting efficiency boost. Because the ranking score solely depends on the document term weights, everything can be pre-computed offline, and inference cost is therefore reduced. It also enables GPU-free inference. Thus, the model becomes more comparable with approaches like DeepImpact [Mallia et al. 2021] or TILDEv2 [S. Zhuang and Zuccon 2021b]. However, document expansion is part of the training loop – contrary to the aforementioned approaches, which rely on fixed expansion techniques (doc2query-T5 and TILDE, respectively). As such, the model can seemingly take advantage of enhanced training procedures such as distillation for end-to-end learning (see Section 3.7).

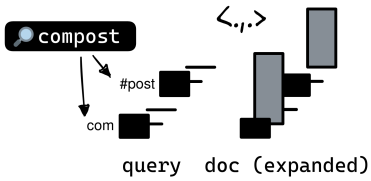


Figure 3.13: SPLADE doc. Here, we can see queries as binary BoW vectors in the WordPiece vocabulary.

> **Remark**
>
> As already mentioned in Section 3.3.2, SPARTA – despite a different formulation of the problem – can be seen as an early version of SPLADE-doc, without regularization constraint.

3.6.2 Experiments

We follow the experimental setting introduced in Section 3.4. For SPLADE-lexical as well as SPLADE-doc, we remove the query regularization part and train models with:

$$\mathcal{L} = \mathcal{L}_{\texttt{rank}} + \lambda_d \ell^d_{\texttt{FLOPS}} \tag{3.22}$$

In practice, we keep stopwords but remove the `[CLS]` and `[SEP]` tokens from the query representation[131]. We found out experimentally that such approaches need to perform fewer training steps to converge – training for longer usually degrades the performance. We hypothesize that joint query and document expansion needs careful calibration, while removing query expansion simplifies the learning problem, by inducing an explicit lexical prior. We thus train models for $50k$ iterations. We additionally train a baseline that does not perform query nor document expansion – referred to as SPLADE-lexical (full). It is thus similar to the SparTerm-lexical baseline in Table 3.3, with max pooling and log activation, and is trained without sparsity regularization.

We report in Table 3.9 the experimental results of efficient SPLADE extensions. First, our fully lexical SPLADE extension outperforms

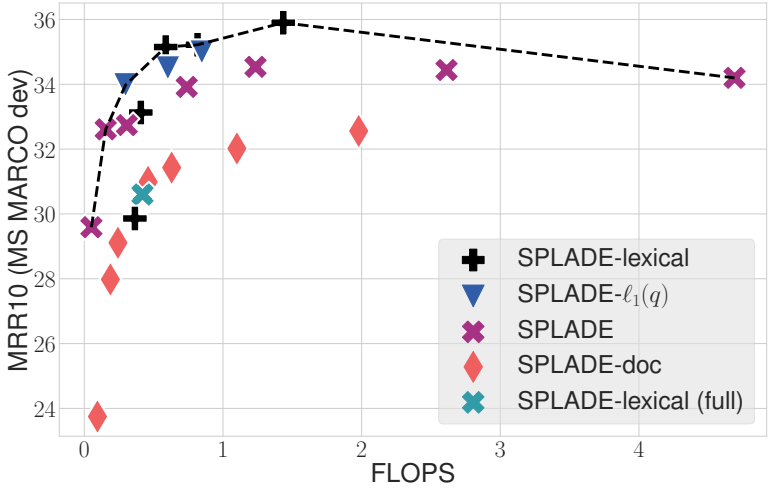
[131] It would drastically increase latency, as these two tokens are usually associated with long posting lists.

Table 3.9: Performance of various efficient SPLADE extensions on MS MARCO (dev) and TREC DL 19. ^{abcde} denote significant improvements over the corresponding rows, for a paired t-test with p-value=0.01.

model	MS MARCO dev		TREC DL 2019	
	MRR@10	R@1000	nDCG@10	R@1000
♢ SPLADE *(a)*	34.5^{bc}	96.5^{bc}	67.6	81.1^{c}
SPLADE-lexical (full) *(b)*	30.6	92.5	64.4	76.3
♢ SPLADE-doc *(c)*	32.6^{b}	94.0^{b}	65.1	71.9
SPLADE-$\ell_1(q)$ *(d)*	35.0^{bc}	96.6^{bc}	70.4^{c}	80.7^{c}
SPLADE-lexical *(e)*	35.9^{abcd}	97.2^{abcd}	69.8^{c}	82.9^{c}

term-weighting models like DeepCT, which reflects the advantage of the implicit "interaction" when mapping term embeddings to the vocabulary – compared to a single score. We notice how SPLADE-doc gets results on par with DeepImpact (32.6 MRR@10), with an overall simplified training scheme. Surprisingly, SPLADE-$\ell_1(q)$, as well as SPLADE-lexical, outperform SPLADE – while having a heavier constraint on the query side. While these results suggest that query expansion does not seem critical for SPLADE, such approaches however achieve much lower performance on zero-shot settings – see Section 3.8 – which might indicate some overfitting. We further report in Figure 3.14 the same type of trade-off already reported in previous Sections. However, in this setting, the FLOPS does not entirely reflect the efficiency gains of the approaches compared to SPLADE – although they still tend to be more on the "left" side of the plot.

Figure 3.14: Effectiveness-Efficiency trade-off on MS MARCO dev set (MRR@10). Note that efficient extensions are more efficient by design – but it is not necessarily reflected by the FLOPS.



We, therefore, evaluate query latency for SPLADE-doc, similarly to

Table 3.10: Experimental results on MS MARCO dev set for SPLADE-doc. $\ell_0(d)$ denotes the average document size, estimated on the full collection $\mathcal{C}$ after training – $\ell_0(q)$ being constant (average 7 on MS MARCO) across models. Note that the retrieval time reported here is the *actual* retrieval time, as no query inference is needed – contrary to SPLADE in Table 3.5.

model	MRR@10	$\ell_0(d)$	Latency (ms)		Index Size (GB)
			Anserini	Numba	
♢ SPLADE	34.5	90	828.37	44.1	2.3
λ_d					
0.5	23.7	11	110.5	16.0	0.9
0.1	28.0	19	177.9	16.6	1.1
0.05	29.1	25	218.8	15.6	1.3
0.01	31.0	47	378.5	19.9	1.7
0.005	31.4	62	490.8	25.2	1.9
0.001	32.1	125	796.3	30.6	3.0
0.0001 ♢	32.6	302	1079.4	50.2	5.8

Section 3.5.5.2, and report results in Table 3.10. Latency measured with Anserini (on a mono-CPU environment) is overall much lower than the ones reported in Table 3.5 – due to the lowest number of posting lists to process (no query expansion)[132]. Latency is still high – and this even for extremely sparse models (110ms for documents with $\ell_0(d) = 11$), which might be related to the distribution of term weights – as discussed in Section 3.5.5.1. To compensate for the lack of query expansion, SPLADE-doc can benefit from "aggressive" document expansion (e.g., documents up to $\ell_0(d) = 302$).

[132] In addition to *not* requiring query inference.

3.7 Towards Improving SPLADE Effectiveness

Context

This Section summarizes the experimental results started in SPLADE v2 [Formal et al. 2021a] and complemented in SPLADE++ [Formal et al. 2022a], towards improving SPLADE *effectiveness*. Distilling a cross-encoder into SPLADE was also part of our contribution to TREC'21 [Lassance et al. 2021b]. We also provide additional experiments and analyses.

Neural retrievers based on dense representations combined with ANN search have recently received a lot of attention, owing their success to distillation and/or a better sampling of examples for training. While sparse approaches have also gained increasing attraction, a lesser effort has been put into the training of such models, making it unclear whether they would experience the same boosts as dense architectures.

We thus wonder if these improvements are *additive*, in the sense that if a model performs better than another in a "base" training setting, would we still observe the same hierarchy with distillation? Answering such a question would allow decoupling architectures from training innovations when comparing neural retrievers.

In this Section, we therefore extensively study the influence of various training strategies on SPLADE *effectiveness*. More precisely, we study the effect of the three main innovations, namely *i*) distillation, *ii*) hard-negative mining, *iii*) as well as the PLM initialization. These techniques have already been introduced in Section 2.6.5.1. We furthermore analyze the link between effectiveness and efficiency and show how it still applies to the considered training settings.

3.7.1 Setting

Let us consider the "default" SPLADE model[133], defined by Eq. 3.9 and trained with Eq. 3.16, where $\mathcal{L}_{\texttt{rank}}$ is the contrastive InfoNCE loss, based on BM25 and In-Batch Negatives negatives. To further distinguish training settings, we thus note:

$$\mathcal{L} = \mathcal{L}_{\texttt{InfoNCE},BM25} + \lambda_q \mathcal{L}^q_{\texttt{FLOPS}} + \lambda_d \mathcal{L}^d_{\texttt{FLOPS}} \tag{3.23}$$

In the following, we detail several training tricks previously introduced for dense models that can seamlessly be applied to SPLADE. All the extensions rely on a simple modification of Eq. 3.23, either by modifying the ranking loss, the source of hard negatives, or a combination of both. We also discuss the initialization strategy, where models can further be improved by relying on PLM checkpoints that have been pre-trained on retrieval-oriented tasks.

[133] Initial experiments with models from Section 3.6, as well as the ones reported in [Lassance and Clinchant 2022], tend to indicate that efficient extensions are far less effective in OOD settings. We thus focus on the canonical SPLADE.

Distillation Distillation of (costly) well-performing models like cross-encoders has been shown to greatly improve the effectiveness of various more efficient neural ranking architectures [Hofstätter et al. 2020a, 2021; S.-C. Lin et al. 2021; Santhanam et al. 2022b]. We rely on the Margin-MSE loss [Hofstätter et al. 2020a] – Mean Squared Error between the positive-negative margins of a cross-encoder teacher and the student – and train SPLADE by optimizing:

$$\mathcal{L} = \mathcal{L}_{\texttt{MarginMSE},BM25} + \lambda_q \mathcal{L}^q_{\texttt{FLOPS}} + \lambda_d \mathcal{L}^d_{\texttt{FLOPS}} \tag{3.24}$$

MarginMSE distillation constitutes our default strategy for all the settings introduced below.

Mining Hard Negatives The standard setting using BM25 negatives is limited, and the benefit of using more informative negatives has been highlighted in several prior works [Qu et al. 2021; Ren et al. 2021b;

L. Xiong et al. 2021; Zhan et al. 2020]. Note that by considering MarginMSE distillation as the basis for hard negative mining, we remove the need to resort to denoising, i.e., filtering noisy false negatives [Qu et al. 2021; Ren et al. 2021b].

Self-mining Following ANCE [L. Xiong et al. 2021], which dynamically samples negatives from the model that is being trained, we propose to follow a simpler two-step strategy that has also been adopted in prior works [S.-C. Lin et al. 2021]:

- (`step 1`) we initially train a SPLADE model, as well as a cross-encoder re-ranker, in the previously introduced distillation setting;
- (`step 2`) we then generate triplets using the SPLADE model trained in (`step 1`), and use the cross-encoder to generate the scores needed for MarginMSE, and go for another round of training.

$$\mathcal{L} = \mathcal{L}_{\texttt{MarginMSE},self} + \lambda_q \mathcal{L}^q_{\texttt{FLOPS}} + \lambda_d \mathcal{L}^d_{\texttt{FLOPS}} \tag{3.25}$$

This simply leads to a distillation strategy where mined pairs are supposed to be of better "quality" compared to BM25 – as discussed in Section 2.6.5.1.

Ensemble-mining While the self-mining setting provides a better sampling strategy compared to BM25, it might be limited, as it only considers a single model, and one could wonder if using various types of models to mine negatives for (`step 2`) could be beneficial. We thus rely on a dataset containing for each query: *i*) the top-50 hard negatives mined from BM25 and each of 12 various dense retrievers, *ii*) the scores coming from a cross-encoder for each available (q, d^+, d^-) to perform MarginMSE knowledge distillation. Our training loss thus becomes:

$$\mathcal{L} = \mathcal{L}_{\texttt{MarginMSE},ensemble} + \lambda_q \mathcal{L}^q_{\texttt{FLOPS}} + \lambda_d \mathcal{L}^d_{\texttt{FLOPS}} \tag{3.26}$$

Pre-training Natural Language Processing has recently borrowed ideas from contrastive learning techniques in Computer Vision, with the goal of learning high-quality sentence or document representations *without annotation* [L. Gao and Callan 2022; Izacard et al. 2022; Neelakantan et al. 2022; Ram et al. 2022b; K. Wang et al. 2021; Z. Wu et al. 2020] – introduced in Section 2.6.5.2. The general idea consists in designing pre-training tasks, that are better suited for subsequently training neural retrievers. Most approaches in IR are very similar to CoCondenser [L. Gao and Callan 2022], which contrastively learns the embedding space from spans of documents – two spans from the same document are considered as positives, the other documents in the batch as negatives. We thus simply consider using such pre-trained checkpoint to initialize SPLADE, in the scenarios described above. Note

that while CoCondenser is not directly optimized for sparse retrieval, its learned embedding space might contain more informative knowledge for retrieval compared to models pre-trained with Masked Language Modeling.

3.7.2 Experiments

We describe our experimental setting in Section 3.7.2.1, and the results achieved by our approach in Section 3.7.2.2.

3.7.2.1 Experimental Setting

We conduct an extensive experimental study, by training and evaluating several models, for each setting introduced in Section 3.7.1. As we provide improvements over SPLADE, we refer to all our strategies under the same alias, coined SPLADE++. Overall, we follow the training and evaluation workflow described in Section 3.4. Especially, we use the same hyperparameters, as well as the validation strategy, but modify the components introduced in Section 3.7.1 – accordingly the ranking loss for distillation, the set of (hard) negatives, and/or the PLM initialization. We thus consider the following settings: `SPLADE`, which corresponds to the original setting (Eq. 3.23); `DistilMSE` which relies on Eq. 3.24 for training[134]; `SelfDistil` where models are trained using Eq. 3.25; `EnsembleDistil`[135]which rather makes use of Eq. 3.26; and finally `CoCondenser-SelfDistil` and `CoCondenser-EnsembleDistil` which correspond to the two latter settings, where the model has additionally been initialized from a pre-trained CoCondenser checkpoint[136].

For each setting, we train five models, corresponding to different values of the regularization magnitude (λ_q, λ_d) in, e.g., Eq. 3.23 – thus providing various trade-offs between effectiveness and efficiency. Note that, as each procedure either modifies the loss or the training pairs, the range taken by loss values slightly differs. We, therefore, need to adapt λ in each case; we simply rely on grid-search and keep five configurations that cover a broad range of effective and efficient models. Note that, as we only compare the efficiency of SPLADE models, we only report the FLOPS and not other measures such as query latency.

[134] We rely on the cross-encoder teacher scores provided at https://github.com/sebastian-hofstaetter/neural-ranking-kd.

[135] We rely on the `msmarco-hard-negatives` dataset https://huggingface.co/datasets/sentence-transformers/msmarco-hard-negatives, available in the Sentence Transformers library [Reimers and Gurevych 2019].

[136] Available via HuggingFace: https://huggingface.co/Luyu/co-condenser-marco.

3.7.2.2 Results

Overall results Results on MS MARCO dev and TREC DL 2019 are given in Table 3.11. For each, we report the best model (among the five that are trained) with a FLOPS value inferior to 3 – similar to results reported in previous Sections. The interplay between effectiveness and efficiency is given in Fig. 3.15: we report MRR@10 on MS MARCO dev *vs* FLOPS, for the five configurations in each training setting. Overall, we observe that:

Table 3.11: Evaluation on MS MARCO passage retrieval (dev set) and TREC DL 2019. We distinguish methods between simple training and training that benefit from various improvements. ^{abcdef} denote significant improvements over the corresponding rows for a paired t-test with p-value=0.01.

model	MS MARCO dev		TREC DL 2019	
	MRR@10	R1000	nDCG@10	R1000
► Simple training				
BM25	18.4	85.3	50.6	74.5
doc2query-T5	27.7	94.7	64.2	82.7
DeepImpact	32.6	94.8	69.5	-
Bi-encoder (ours)	31.2	94.1	63.7	71.1
COIL-full	35.5	96.3	70.4	-
ColBERT	36.8	96.9	-	-
♢ SPLADE *(a)*	34.5	96.5	67.6	81.1
► Distillation, negative mining or pre-training				
ANCE	33.0	95.9	64.8	-
TCT-ColBERT	35.9	97.0	71.9	76.0
TAS-B	34.7	97.8	71.7	84.3
RocketQA-v2	38.8	98.1	-	-
CoCondenser	38.2	98.4	-	-
Contriever	34.1	97.9	67.6	84.3
AR2	39.5	98.6	-	-
ColBERTv2	39.7	98.4	-	-
► Our methods: SPLADE++				
♢ DistilMSE *(b)*	35.8^{a}	97.8^{a}	72.9	85.9
♢ SelfDistil *(c)*	36.7^{ab}	98.0^{a}	72.6	87.9^{a}
♢ EnsembleDistil *(d)*	36.8^{ab}	97.8^{a}	70.7	86.7
♢ CoCondenser-SelfDistil† *(e)*	37.6^{abcd}	98.3^{abcd}	73.2^{a}	87.7^{ab}
♢ CoCondenser-EnsembleDistil‡ *(f)*	38.0^{abcd}	98.2^{abd}	72.7^{a}	87.1^{a}
► Additional baselines				
SPLADE (BERT-base)	34.0	96.8	71.3	81.0
SPLADE (BERT-base, EnsembleDistil)	37.2	97.9	72.6	86.1
SPLADE-doc (CoCondenser-EnsembleDistil)	36.6	96.7	70.2	79.5

1. *SPLADE is able to take advantage of various training strategies to increase its effectiveness*;
2. *performance boosts are additive*;
3. *our settings lead to competitive results on MS MARCO (in-domain evaluation), e.g.,* `CoCondenser-EnsembleDistil` *reaches* 38 *MRR@10*;
4. *model effectiveness is linked to efficiency for all the considered settings (the sparser, the less effective).*

More specifically, we observe from Table 3.11 that the use of distillation (`DistilMSE`) offers the largest boost in effectiveness across all settings (↑+1.3 MRR@10). Note that a model based on BERT (instead of DistilBERT) is able to better take advantage of distillation (37.2 MRR@10, so ↑+0.4 MRR@10), which indicates that larger models can better align to the cross-encoder teacher.

When combining distillation with hard negative mining, we note that the `SelfDistil` case seems to be the most effective. However, when changing the initialization checkpoint to CoCondenser, we observe the reverse trend, where `CoCondenser-EnsembleDistil` is able to outperform its counterpart. Distillation with hard negatives is, therefore, beneficial, but the exact "optimal" setting depends on the pre-training strategy. We also note that improvements are less clear when inspecting other metrics like R@1000 or nDCG@10 on TREC DL 2019.

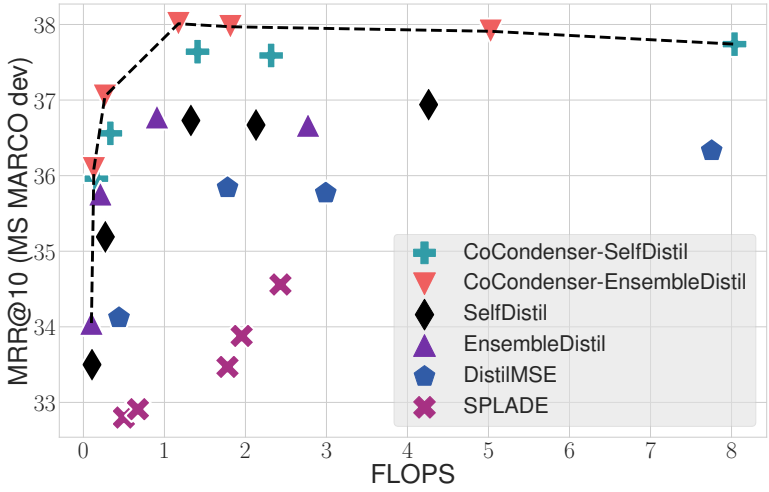
In Figure 3.15, we analyze the interplay between effectiveness and efficiency (in terms of FLOPS) on MS MARCO dev set. There is an overall trend that more expressive models tend to be more effective across all training settings. We also observe additivity in improvements, with the best model on MS MARCO (`CoCondenser-EnsembleDistil`) taking advantage of distillation, ensemble mining, and pre-trained checkpoints altogether. Additionally, by combining all the strategies, we are able to obtain extremely effective SPLADE-doc models (e.g., 36.6 MRR@10), which indicates that more efficient architectures can also benefit from such improvements[137].

[137] These aspects are further developed in [Lassance and Clinchant 2022].

Additivity To further demonstrate the "additivity" of training settings, we furthermore train, for each setting[138], a dense bi-encoder with the same DistilBERT backbone model. We rely on mean pooling of the contextualized embeddings and follow standard practices to train them. We plot in Figure 3.16 the MRR@10 achieved by the dense bi-encoders, alongside different trade-offs for SPLADE models. Results suggest that SPLADE can always outperform its dense counterpart – regardless of the training setting – given enough expansion power, i.e., up to a given FLOPS for the last two. This suggests that SPLADE architectural prior, based on sparse expansion and term-matching, is somewhat well suited for IR, when compared to the dense matching of

[138] Except the ones based on Cocondenser.

Figure 3.15: Effectiveness-Efficiency trade-off on MS MARCO dev set. Please note that the points corresponding to SPLADE slightly differ from previous Figures – they correspond to different sets of experiments, at different time frames.



semantic vectors. This study, however, needs to be complemented with the various pre-training techniques introduced for dense models: would we still observe the same gap? This further questions how to adapt pre-training techniques for models such as SPLADE – which is not as straightforward [Shen et al. 2022a].

3.8 Zero-Shot Evaluation

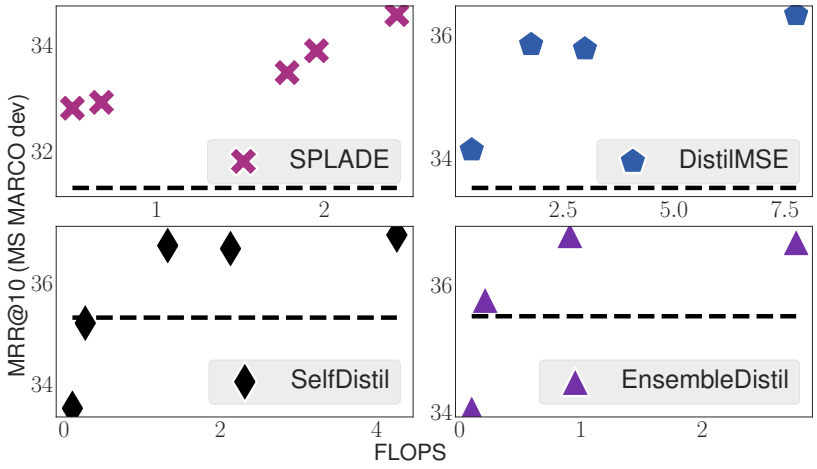
Context

This Section summarizes the zero-shot evaluation of SPLADE models on the BEIR benchmark [Formal et al. 2021a, 2022a].

In Information Retrieval, the question of generalization has often been eluded, due to the robust and long-standing performance of term-based approaches. However, with the recent advent of neural retrieval based on PLM, the generalization issue has become as relevant as ever. Initially evaluated on in-domain settings (like MS MARCO), where train and test queries follow the same distribution, conclusions became more contrasted when [Thakur et al. 2021] released the zero-shot BEIR benchmark (introduced in Section 2.3.2.2) – in which some models like DPR achieve lower overall performance compared to unsupervised term-based approaches like BM25. Knowing that many production systems use (or will use) models based on PLM (e.g., [L. Xiong et al. 2021]), while being exposed to new documents and queries every day, robustness is thus a critical aspect that must be assessed.

The BEIR benchmark consists of a test suite of 18 datasets – each

Figure 3.16: Additivity study: each subplot corresponds to a given training setting, and the dense bi-encoder performance is shown with the dotted black line. It reaches 31.3, 33.5, 35.3, and 35.5 MRR@10 on MS MARCO for, respectively the "standard", DistilMSE, SelfDistil, and EnsembleDistil training settings.



containing documents, queries, and corresponding *qrels*. They are used to evaluate models in a *zero-shot* setting, i.e., without any sort of training based on those datasets.

In this Section, we thus assess the zero-shot performance of our models on the BEIR benchmark[139]. For comparison with other approaches, we rely on the subset of 13 datasets that are readily available. Thus, we do not consider `CQADupstack`, `BioASQ`, `Signal-1M`, `TREC-NEWS` and `Robust04`; otherwise, we evaluate our models on the complete benchmark (18 datasets). We report the results from the models introduced in Section 3.7, and show the effect of various training strategies on generalization capabilities. We furthermore analyze the link between effectiveness and efficiency in Figure 3.17. Please note that for BEIR, averaging metrics over multiple datasets is questionable [Soboroff 2018] – but the observed trends are, however, useful to compare various instances of SPLADE models. We nevertheless provide an evaluation on every dataset in Table 3.14.

Overall, we observe that:

1. *the SPLADE models introduced in Section 3.7 lead to state-of-the-art results on zero-shot evaluation*;
2. *the link between effectiveness and efficiency still applies in out-of-domain evaluation*;
3. *out-of-domain improvements are consistent with the ones observed on MS MARCO.*

We indeed observe in Figure 3.17 a trend similar to the one observed in Section 3.7, where effectiveness comes with higher complexity.

[139] We are aware of other datasets that serve the same purpose, such as the LoTTE benchmark [Santhanam et al. 2022b], specifically tailored for OOD retrieval of natural search queries over long-tail topics.

Figure 3.17: Effectiveness-Efficiency trade-off on BEIR (18 datasets – results are thus not comparable to Table 3.12). We report mean nDCG@10.

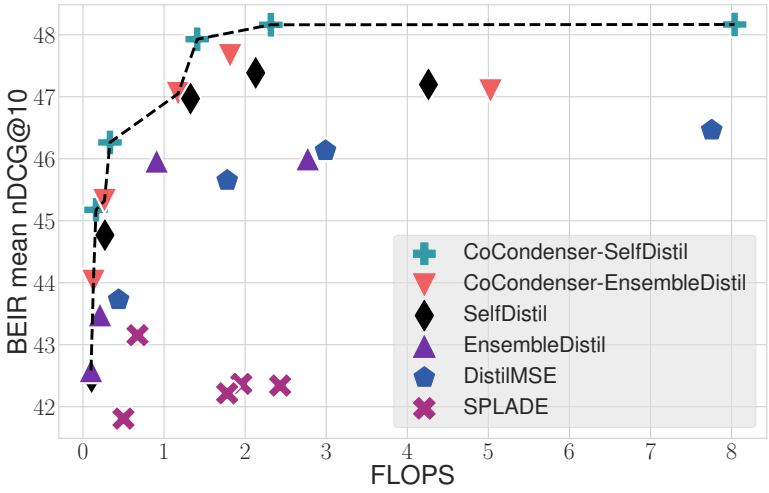


Table 3.12: Mean nDCG@10 on the subset of 13 BEIR datasets. SPLADE++‡,† respectively correspond to our best SPLADE models CoCondenser-EnsembleDistil‡ and CoCondenser-SelfDistil†.

	BM25	TAS-B	Contriever	ColBERTv2	SPLADE	SPLADE++‡	SPLADE++†
`nDCG@10`	43.7	43.7	47.5	49.7	45.7	50.4	50.7

However, the `SelfDistil` settings seem to be better suited for generalization – they both respectively outperform their `EnsembleDistil` counterparts, also see Table 3.12 – contrary to what is observed on MS MARCO. One reason for this behavior might be the use of only dense retrievers in the latter case. As such models have been shown to overfit on MS MARCO, the mined negatives might be too specialized on this dataset, thus hurting the generalization capabilities of subsequently trained models.

We additionally report in Table 3.12 the BEIR results for our two CoCondenser settings, reaching state-of-the-art results on zero-shot evaluation[140], strengthening the observation that sparse retrieval models seem to be better able to generalize [Formal et al. 2022b; Mokrii et al. 2021b; Thakur et al. 2021]. Note that the "standard" SPLADE is already competitive, while our SPLADE `CoCondenser-SelfDistil` is the best overall on five datasets (out of the 13), among other state-of-the-art baselines. Note that BM25 still performs well, and many baselines reported in [Thakur et al. 2021] achieve lower performance. If we look at individual results, we see that SPLADE models largely outperform BM25 on datasets like Robust04 or TREC-COVID.

Also note that when switching from DistilBERT to BERT, we notice a performance boost (↑+0.9 mean nDCG@10, not reported in Table 3.12). This indicates that larger models are helpful in zero-shot

[140] We acknowledge that recent works also achieve impressive zero-shot performance [Dai et al. 2022; Qian et al. 2022]. Additionally, domain adaptation techniques have been shown to be effective on the BEIR benchmark [K. Wang et al. 2022].

	SPLADE++‡ + BM25	SPLADE++† + BM25
nDCG@10	52.1	52.1

Table 3.13: Mean nDCG@10 on the subset of 13 BEIR datasets for a simple combination (sum) of SPLADE++ and BM25.

settings, despite reaching a similar level of performance on in-domain evaluation[141].

We also report in Table 3.13 the results of the two previous approaches combined with BM25 – with a simple sum. Additional gains can be obtained, showing that pure lexical approaches are still somehow complementary to sparse neural models, especially in a zero-shot setting. While such results have been shown for dense bi-encoders [Bruch et al. 2022; Tao Chen et al. 2022; H. Li et al. 2022b; Luan et al. 2021; Xueguang Ma et al. 2021; S. Wang et al. 2021], this is more surprising for sparse models.

[141] Such observations have also recently been made for cross-encoders [G. Rosa et al. 2022].

3.9 Conclusion

We have introduced SPLADE, an *effective* sparse bi-encoder that jointly learns expansion and term weighting (Section 3.3.4). The sparse nature of its learned representations – by means of an explicit regularization (Section 3.3.4.2) – makes it possible to rely on inverted indexes to perform *efficient* retrieval. Additionally, by learning document and query vectors grounded in the vocabulary, the model becomes *interpretable* (Section 3.5.7). The end-to-end nature of SPLADE makes it able to take advantage of the recent training improvements for neural retrievers such as distillation or hard-negative mining – allowing us to reach competitive and state-of-the-art results in *in-domain* and *zero-shot* evaluation settings (Sections 3.7 and 3.8, respectively). Overall, SPLADE is the first sparse model to outperform its dense counterparts.

While we have briefly discussed efficient extensions (Section 3.6), we have mostly been interested in effectiveness improvements – a more thorough study of efficient SPLADE models can be found in [Lassance and Clinchant 2022]. Overall, SPLADE is an appealing candidate to replace traditional term-based approaches such as BM25 in modern search engines: it is effective, robust, and can be made fast enough for production scenarios. It is also interpretable *by design*, which allows the provision of clear and accurate explanations to users about why a particular result was returned for a given query.

Table 3.14: nDCG@10 on BEIR for all the datasets (18). For comparison, we report results directly from corresponding papers, where the evaluation is generally done on the subset of 13 readily available BEIR datasets.

Model (→)	**BM25**	**TAS-B**	**Contriever**	**ColBERTv2**	**SPLADE**	**SPLADE++**‡	**SPLADE++**†
Dataset (↓)				nDCG@10			
TREC-COVID	65.6	48.1	59.6	**73.8**	62.9	72.1	72.5
BioASQ	46.5	38.3	-	-	43.4	50.5	**50.8**
NFCorpus	32.5	31.9	32.8	33.8	31.9	**34.6**	34.5
NQ	32.9	46.3	49.8	**56.2**	45.9	53.5	53.3
HotpotQA	60.3	58.4	63.8	66.7	65.4	68.4	**69.3**
FiQA-2018	23.6	30.0	32.9	**35.6**	27.1	34.7	34.9
Signal-1M (RT)	**33.0**	28.9	-	-	29.6	30.2	30.9
TREC-NEWS	39.8	37.7	-	-	36.3	**42.3**	41.9
Robust04	40.8	42.7	-	-	41.0	46.0	**48.5**
ArguAna	31.5	42.9	44.6	46.3	46.7	51.3	**51.8**
Touché-2020	**36.7**	16.2	23.0	26.3	22.0	25.5	24.2
CQADupStack	29.9	31.4	34.5	-	32.5	33.4	**35.4**
Quora	78.9	83.5	**86.5**	85.2	81.1	83.4	84.9
DBPedia	31.3	38.4	41.3	**44.6**	36.4	43.7	43.6
SCIDOCS	15.8	14.9	**16.5**	15.4	14.2	15.8	16.1
FEVER	75.3	70.0	75.8	78.5	76.2	79.2	**79.6**
Climate-FEVER	21.3	22.8	23.7	17.6	18.6	22.9	**23.7**
SciFact	66.5	64.3	67.7	69.3	65.7	70.5	**71.0**
best on (out of 13)	1	0	2	4	0	1	5

Chapter 4

Analyzing Neural Ranking Models

Outline

In this Chapter, we are interested in understanding *how* neural Information Retrieval models work. In particular, we focus on two specific behavior, namely *lexical match* and *term importance*. We first propose indicators specifically tailored for the ColBERT model and show how it still implicitly relies on such aspects – despite its semantic nature – especially for *important* terms. We furthermore propose an interpretation based on the properties of the learned embedding space. In a second contribution, we delve deeper into the ability of neural rankers to perform keyword matching *off-the-shelf*. In particular, we show how models based on a lexical prior are more inclined to match query terms exactly, and how the training frequency of terms heavily influences such a property.

This Chapter is built on our two contributions – both adapted as extended summaries[†]:

- [Formal et al. 2021b] *A White Box Analysis of ColBERT.*
- [Formal et al. 2021d][†] *Une Analyse du Modèle ColBERT.*
- [Formal et al. 2022b] *Match Your Words! A Study of Lexical Matching in Neural Information Retrieval.*
- [Formal et al. 2022c][†] *Match Your Words! A Study of Lexical Matching in Neural Information Retrieval (extended abstract).*

4.1 Introduction

The impact of Pre-trained Language Models on Information Retrieval goes beyond what the community thought would be possible a few years

back. In particular, neural retrievers – whether dense or sparse – hold the promise to replace traditional term-based approaches such as BM25 in modern search engines. This represents a significant paradigm shift for models and search architectures that have withstood the test of time.

Despite this rapid progress, little is known about the inner workings of large-scale transformers in the IR setting. This can challenge their widespread adoption in production systems, which are increasingly faced with providing transparent, trustworthy, and explainable results to users – as it can be challenging to understand why a particular result was returned for a given query. Models analysis is also interesting from a performance point of view, as it can help make adjustments or even guide the design of new approaches that better encode a specific behavior shown to be lacking. These aspects are particularly concerning given the evergrowing size of Pre-trained Language Models that operate beyond human comprehension, which urges the design of appropriate analysis tools.

This chapter contributes to a better understanding of current neural ranking models based on PLM. Section 4.2 provides the necessary background by introducing the literature around transformer analysis in Natural Language Processing and Information Retrieval. In Section 4.3, we detail our contribution towards understanding the matching process underlying the ColBERT model. Finally, in Section 4.5, we propose a measure that quantifies the extent to which a model relies on term matching to study the impact of the frequency of training terms on generalization.

4.2 Setting The Stage: Analyzing Transformers

In this Section, we present recent works dedicated to better understanding transformer models in Natural Language Processing (Section 4.2.1) and Information Retrieval (Section 4.2.2).

4.2.1 Towards Understanding Transformers in NLP

Given the wide adoption of Pre-trained Language Models by the NLP community – which goes way beyond IR – many studies have been dedicated to unveiling the underlying properties of transformers and what aspects of language they are able to learn from unlabeled data. We do not aim to be exhaustive but give a few pointers to significant contributions. Please refer to [Rogers et al. 2020] for a complete overview of what is informally referred to as "BERTology".

[Tenney et al. 2019] show how BERT encompasses the various steps of the standard NLP pipeline (POS tagging, Parsing, Named Entity

Recognition, etc.) in localized regions (i.e., layers) of the network, based on a series of probing tasks. In a nutshell, probing – which has been extensively used to carry out analyses, see [Belinkov 2022] – aims to assess whether a particular property can be extracted from representations. In its simplest form, it usually consists in training a small classifier to predict the quantity of interest (for instance, Named Entities) directly from the frozen embeddings.

Departing from the black-box nature of probing, [Voita et al. 2019a] propose to directly investigate representations as well as the information flow across transformer layers by relying on Mutual Information and Canonical Correlation Analysis. This allows showing – among other findings – how models trained with Masked Language Modeling preserve token identity.

Other works have directly investigated attention. As such, [Michel et al. 2019; Voita et al. 2019b] show how a large portion of the attention heads can safely be removed from the network without significantly hurting performance. On the other hand, [Clark et al. 2019] deep delve into BERT's attention patterns, showing, for instance, how it attends to delimiter tokens. [Brunner et al. 2020], however, demonstrate how attention weights are not *identifiable* for long sequences – challenging the reliability of previous works regarding the interpretability of attention distributions.

4.2.2 Towards Understanding Transformers in IR

While studies in NLP (Section 4.2.1) have identified various linguistics features learned by PLM, for what concerns Information Retrieval, findings are more limited. The specificity of the task (modeling *relevance* instead of *similarity*), as well as the structure of the problem (e.g., matching short keyword-based queries to long documents), influence what needs to be learned by neural rankers. In many retrieval scenarios, models only need to have a coarse *understanding* of the information need and the content of documents – contrary to complex NLP tasks that, for instance, require *reasoning*. Thus, many works have proposed identifying whether models encode certain well-studied IR principles such as exact matching or term importance.

Remark

While cross-encoders and dense bi-encoders are rather opaque and hard to analyze, some models are more *interpretable* by design. This is the case of sparse approaches such as SPLADE (e.g., Section 3.5.7) and interaction models like ColBERT (see Section 4.3).

4.2.2.1 Axiomatic Approaches

A first line of work has investigated how neural models fit to traditional IR axioms [Fang et al. 2004]. Axiomatic approaches are analytic techniques designed to analyze – and improve – traditional IR models. They are based on a set of formalized constraints that a "good" retrieval model should fulfill. [Câmara and Hauff 2020; Rennings et al. 2019] propose to build diagnostic datasets to assess whether certain IR properties (such as the IDF axiom) are respected (or not) by neural re-rankers. In particular, Câmara and Hauff show how a DistilBERT cross-encoder does not fully respect a large set of axioms – despite its large effectiveness boost when compared to BM25 – questioning their adequacy to analyze neural models. More fundamentally, it questions the axioms themselves, which are largely biased toward lexical approaches. [Völske et al. 2021] further delve into axiomatic diagnosis and show how existing axioms are too restrictive to fully explain observed rankings – for both pre- and post-BERT re-rankers.

4.2.2.2 Probing Approaches

Given the limitations of IR axiomatic, new approaches based on probing have been developed to analyze the behavior of neural IR models. [MacAvaney et al. 2022] propose a probing framework (dubbed ABNIRML – for Analyzing the Behavior of Neural IR ModeLs) that allows the systematic analysis of models beyond simple IR axioms. It comprises three probing strategies, including, for instance, Textual Manipulation Probes, that are used to test the effect of various text properties – such as word order or paraphrasing – on ranking effectiveness. Various models are shown to be sensitive to such perturbations. In the same way [Rau and Kamps 2022b] show how syntactic aspects of input sequences are *not* the reason for the effectiveness of BERT cross-encoders – by manipulating word order and position information.

Remark

These lines of work have shown – among other findings – that neural rankers remain quite sensitive to lexical matching and term statistics in documents or collections [J. Choi et al. 2022; Jiang et al. 2021; Rau and Kamps 2022a]. Such post-hoc analyses are however limited, in the sense that the observed phenomena cannot be explained. Additionally, they have, for the most part, been applied to re-rankers. In Section 4.3, we propose an approach to dissect ColBERT under the IR lens, leading to the same type of findings [Formal et al. 2021b].

4.2.2.3 Robustness and Generalization Aspects

Search engines are inherently faced with distribution shifts in the form of new topics, Web pages, entities, or even users' interests. Traditional term-based approaches, being unsupervised, are naturally robust to such changes. Neural ranking models are, however, dependent on the training data. Given their widespread adoption in Information Retrieval, another line of work has investigated the robustness of neural ranking models to various distribution shifts.

[Gerald and Soulier 2022; Lovón-Melgarejo et al. 2021] show how neural re-rankers (pre- and post-BERT) are subject to catastrophic forgetting [Kirkpatrick et al. 2017], by forgetting existing knowledge when transferring to new domains or datasets. [Penha et al. 2022] study the robustness of various re-ranking models to typos – as search engines directly interact with users and may be exposed to such issues. Additional works complement these findings for dense bi-encoders [Sidiropoulos and Kanoulas 2022; S. Zhuang and Zuccon 2021a, 2022]. Overall, models are shown to be sensitive to such shifts in query distributions. From a different perspective, [Song et al. 2022; C. Wu and R. Zhang 2022; C. Wu et al. 2022] show how various neural IR models (mostly re-rankers) are vulnerable to adversarial attacks – usually by substituting words in documents or queries. These studies usually focus on a single model type and lack the comparison between the various approaches proposed to tackle first-stage ranking, for which robustness might even be more critical. Re-rankers might rely differently on certain properties like exact matching by re-ranking a pool of documents retrieved by BM25.

The BEIR benchmark [Thakur et al. 2021] (Section 2.3.2.2) has pioneered large-scale *zero-shot* evaluation of various neural retrieval models – pointing out how brittle current approaches can be to various domain shifts. Dense bi-encoders are shown to achieve lower overall performance than term-based approaches like BM25. [Mokrii et al. 2021a] concurrently carried out an extensive evaluation of transfer capabilities on 5 English datasets – however limited to cross-encoders. By building an entity-rich question dataset based on Wikidata facts, [Sciavolino et al. 2021b] study how dense retrievers vastly underperform traditional sparse retrievers – highlighting the issue of matching entities not seen at training time. Using the two train sets of MS MARCO and Natural Question, [Ren et al. 2022] identify key factors that affect the zero-shot properties of dense models – including the overlap between the source and target query sets, as well as the query type distribution. [Zhan et al. 2022b] furthermore identify a strong train-test overlap within MS MARCO queries[142], questioning the actual *in-domain* effectiveness improvements made so far. To address the issue, they introduce two new resampling strategies that allow comparing the zero-shot properties of several retrieval architectures accurately, solely based on MS MARCO. In this new setting, dense bi-encoders fail to generalize compared to

[142] [Fröbe et al. 2022] identify the same type of leakage between MS MARCO and Robust queries.

cross-encoder or SPLADE – confirming the observations made earlier in [Tao Chen et al. 2022; Formal et al. 2022a; Thakur et al. 2021]. Similarly, [Lupart et al. 2023] build *controllable* MS MARCO query shifts that not only avoid train-test overlaps but also help to analyze the link between out-of-domain effectiveness drops and the similarity between train and test sets.

Remark

In Section 4.5, we propose to study zero-shot properties of various neural rankers [Formal et al. 2022b]. However, we do not focus on effectiveness transfer but rather the extent to which models perform lexical matching, especially for terms not seen at training time and/or for which statistics change (i.e., in OOD setting).

4.3 ColBERT Analysis

Context

This Section summarizes the first contribution of this thesis [Formal et al. 2021b,d] – that predates SPLADE – in which we devise indicators specifically tailored to analyze the ColBERT model. In particular, we study *i*) term importance and *ii*) lexical matching. We additionally extend the findings to the full-ranking setting.

There exists a wide variety of neural rankers based on Pre-trained Language Models, from cross-encoders (Section 2.6.1) to dense and sparse bi-encoders (Section 2.6.3 and Chapter 3, respectively) – each of them incorporating their own architectural biases into the way they order documents. While some studies have proposed generic analysis tools that can be applied to any given architecture [Câmara and Hauff 2020; MacAvaney et al. 2022], they are usually limited to shallow post-hoc analyses that cannot map the observed phenomena to specific parts of the models.

In this Section, we take an opposite direction and design IR indicators specifically tailored to analyze the ColBERT model [Khattab and Zaharia 2020; Santhanam et al. 2022b] introduced in Section 2.6.3.3. Besides its compelling effectiveness-efficiency trade-off, ColBERT has an appealing property: the score $s(q, d)$ between a query and a document can be expressed as a sum over query tokens[143] of a similarity score, based on the cosine similarities of the contextualized representations (colored squares in Figure 4.1). The contribution of each term is thus *explicit*. Its general structure is, therefore, akin to traditional term-based approaches. We propose to rely on such a view to dissect ColBERT from an IR perspective – hence "white box" analysis. On the

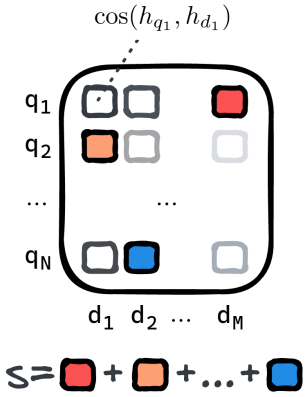


Figure 4.1: Reminder of ColBERT's architecture: for each query token q_i, we select the most similar token in the document – based on cosine similarity. The final score is a sum of each of such *contributions*.

[143] Note that the sum is over query *subwords* of the WordPiece vocabulary, and not actual query terms as traditionally defined in IR. In the remainder of the chapter, we usually use *token* to refer to subword units.

other hand, models like cross-encoders are difficult to analyze due to the difficulty in characterizing the attention mechanism [Brunner et al. 2020]. Similarly, the latent representation space of dense bi-encoders is rather opaque, limiting their analysis.

Very broadly, we want to assess whether ColBERT encodes different properties for terms depending on their statistics – such as IDF. In Section 4.3.2, we, therefore, investigate the link between term importance as computed by standard IR models and the one computed by ColBERT. In Section 4.3.3, we look at how ColBERT is dealing with *exact* and *soft* matches and how it stems from the learned embedding space (Section 4.3.3.2).

4.3.1 Experimental Setting

Model We introduce the slightly different variant of ColBERT [Khattab and Zaharia 2020] we use to simplify the analysis – compared to the model described in Section 2.6.3.3. In particular, we do not include the query (resp. document) specific tokens `[Q]` (resp. `[D]`) since they could bias the term representations. Second, while the artificial query expansion (based on the `[MASK]` tokens) was shown to be beneficial [Hofstätter et al. 2020a; Khattab and Zaharia 2020], we found no clear experimental evidence of its benefit – at least when evaluated on MS MARCO. These results were later confirmed in [Lassance et al. 2022; Tonellotto and Macdonald 2021]. We thus remove this component from our model to avoid the analysis of the induced implicit query expansion mechanism. We, however, keep the compression layer that maps token representations to a lower dimensional space (from $d = 768$ to $d = 128$). By fine-tuning our "light" ColBERT model, we obtain results on par with the ones reported in [Khattab and Zaharia 2020] – e.g., 35.4 MRR@10 for end-to-end retrieval on the MS MARCO dev set.

To understand the influence of fine-tuning on IR properties, we also contrast the results with a model that has not been fine-tuned (and without compression layer), that is solely based on the output of a pre-trained BERT model – referred to as `NF`, for non-fine-tuned. Note that such a model has an extremely low performance on MS MARCO (close to null MRR@10 on end-to-end ranking).

To ease reading, we re-introduce the formal definition of ColBERT from Section 2.6.3.3. Given the BERT embeddings $(h_q)_i$ and $(h_d)_j$ for respectively the query q and document d, ColBERT computes relevance

as:

$$s(q,d) = \sum_{i \in q} \text{MaxSim}(q_i, d) = \sum_{i \in q} \max_{j \in d} \cos(h_{q_i}, h_{d_j})$$
$$= \sum_{i \in q} \max_{j \in d} C_{ij}$$
$$= \sum_{i \in q} C^{\star}_{id} \quad (4.1)$$

In the following, we say that *a query token i matches the document token j^** if $C_{ij^*} = \max_{j \in d} C_{ij} = C^{\star}_{id}$. We denote this token j^* by $d^{\star}_i$.

Data For this analysis, we rely on the passage retrieval tasks from TREC DL 2019 and 2020 – 400 queries in total. We first consider a re-ranking setting, where for a given query q, the model needs to re-rank a set of documents $\mathcal{S}_{q,BM25}$ retrieved by BM25[144]. We additionally extend the analysis to the full-ranking scenario, where ColBERT retrieves (and ranks) a set of documents $\mathcal{S}_{q,Col}$ from the complete collection $\mathcal{C}$[145]. The two settings are somewhat complementary: in the former, the model might be less reliant on lexical signals, as they are abundant in documents retrieved by BM25. In the latter case, we directly analyze ColBERT's propensity to rely on lexical cues to *retrieve* relevant documents from $\mathcal{C}$.

To study the model properties, we are interested in *how ColBERT attributes scores to each query token, for documents in $\mathcal{S}_q$*. Note that, as our indicators do not rely on relevance judgments, we could use larger query sets (such as the MS MARCO dev queries) – we, however, initially observed the same phenomena and thus kept a lower number of queries to simplify the procedure. For term-level indicators (Sections 4.3.2 and 4.3.3), we rely on standard tokenization with Porter stemming and keep stopwords.

[144] The default re-ranking files provided by TREC.

[145] In the first case, we have $|\mathcal{S}_{q,BM25}| \leq 1000$, as BM25 might retrieve a lower number of documents for difficult queries. In the full ranking setting, we consider $|\mathcal{S}_{q,Col}| = 1000$.

4.3.2 Term Importance

We are first interested in comparing term importance for standard IR models such as BM25 with term importance as determined by ColBERT.

With respect to the former, we rely on a simple approximation. Given that we restrict our study to the MS MARCO passage collection, term frequency is close to 1 for most terms ($\text{avg}(tf) \approx 1.1$). Moreover, passage length does not vary much, and is capped at 512 tokens – due to BERT's maximum sequence length. Hence, we can reasonably assume that the BM25 score of a term – and thus its importance as determined by the model – roughly corresponds to its IDF[146].

For ColBERT, it is *a priori* difficult to measure the importance of a query term, as it depends on both document and query contexts ($C^{\star}_{id}$

[146] This might not be true for terms with low IDF. Still, it is a good enough approximation for other terms.

in Eq. 4.1). We thus resort to an indirect mean, by measuring the correlation between the initial ColBERT ranking π, and the ranking π_t obtained when we remove from the sum in Eq. 4.1 all the *contributions* of subwords that compose the corresponding term. More specifically, let us consider an input query term t, and denote by $J \in q$ the indices of its subword decomposition. We consider the modified scoring (leading to the modified ranking π_t):

$$\begin{aligned} s(q,d;\neg t) &= s(q,d) - \sum_{i \in J} C^{\star}_{id} \\ &= \sum_{i \in q, i \notin J} C^{\star}_{id} \end{aligned} \tag{4.2}$$

We then compare π and π_t by measuring their AP-correlation τ_{AP} [Yilmaz et al. 2008], which is akin to Kendall's τ rank correlation [Kendall 1938], but gives more weight to the differences at the top of the ranking. It is formally defined as:

$$\tau(\pi, \pi_t) = \frac{2}{N-1} \sum_{i=2}^{N} \left(\frac{C(i)}{i-1} \right) - 1 \tag{4.3}$$

where N denotes the size of the two lists to compare, and $C(i)$ corresponds to the number of items above rank i and correctly ranked w.r.t. to the item at rank i in π_t. Values close to 1 indicate a strong correlation, meaning that the two rankings are similar – thus implying a low contribution of the term to order documents. Note that we do not mask terms from the query themselves – only their scores – as representations from the PLM would be impacted by their full removal. The side effect is that some non-semantic term representations might be influenced by more semantically bearing words that are removed from the sum, thus impacting the actual term importance estimation (Section 4.3.3). The overall idea is illustrated in Figure 4.2, and we show four examples with TREC queries in Figure 4.3. Very intuitively, we observe that what seems to be *important* query terms have the largest contribution to the ranking (i.e., lowest τ_{AP}). On the contrary, terms like stopwords seem to have a rather low influence: removing their contribution from ColBERT's scoring results in a very similar ordering of documents (τ_{AP} close to 1). In the following, we aim to generalize these findings.

We show in Figure 4.4, the relation between τ_{AP} and a standard measure of importance – namely IDF[147] – on both re-ranking and full ranking settings[148], for terms in the TREC test queries. Note that τ_{AP} is query-dependent: when a term appears in several queries, we consider its average τ_{AP} as a final measure of importance. Additionally, query length could have an impact on the analysis – the more terms in the query, the less the contribution for each of them. We, however, didn't

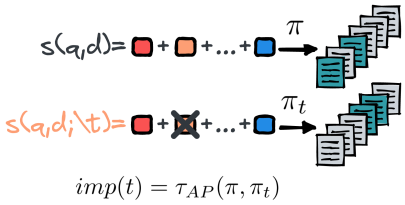
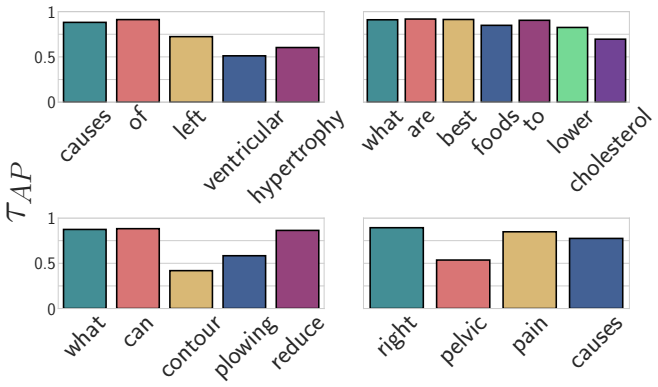


Figure 4.2: Illustration of the approach. We assess with τ_{AP} how the ranking changes when we remove a term contribution.

[147] Please note that we operate at the term level – by masking all the corresponding subword contributions – and rely on actual IDF.

[148] What changes is the set of documents we consider, i.e., $\mathcal{S}_{q,BM25}$ or $\mathcal{S}_{q,Col}$.

Figure 4.3: ColBERT term importance (as computed using τ_{AP}), for terms of four queries from TREC DL 2019-2020. Note that we operate at the term level: when a term is decomposed into subwords, we simply remove all their contributions in Eq. 4.1 – e.g., "*hypertrophy*" → "*hyper*","*##tro*","*##phy*" in the first query.



notice any significant difference when splitting the analysis by query length, and we thus kept the simplest formulation.

In the re-ranking setting, there is a low linear negative correlation between both metrics (Pearson correlation coefficient $r = -0.36$), showing that ColBERT's term-level matching implicitly captures IDF. Note that words with higher IDF tend to be longer and hence, to be split into multiple subwords more often – by construction of the WordPiece vocabulary – increasing the importance of such terms by design. The negative correlation even decreases in the full ranking ($r = -0.40$), indicating that ColBERT relies even more on traditional heuristics when retrieving from the full collection – thus being closer to how BM25 ranks documents. As should be expected, terms with low IDF tend to have an overall low contribution in both settings. [Tonellotto and Macdonald 2021] later showed how they can actually be removed from the model without impacting effectiveness.

Except for the last two bins, the variance of τ_{AP} also increases with IDF. It might partly be explained by the learned contextualized term importance, which is able to identify (un)important words beyond their simple statistics. Many terms with middle-valued IDF on MS MARCO have either high or low importance as determined by ColBERT – as shown in Table 4.1. For instance, the term "*lipids*" is considered important *in this particular query*. On the other hand, "*dot*" corresponds to a typo of "*dop*" (a brand); this term is, however, not necessary to answer the information need (about a specific hair gel, "*vivelle*").

We also observe that the link between IDF and term importance is not so direct for high values (> 8). One of the plausible explanations is that another query token (with no semantics) bears the same semantics as the target one. For each query token, we looked at the frequency of exact matching (i.e., the max similarity is obtained with the same token in a document) and at the frequency with which it matches in docu-

Figure 4.4: ColBERT term importance (as computed using τ_{AP}) with respect to IDF (x-axis, binned). Note that we do not consider the `NF` (non-fine-tuned) model here, as we aim to compare rankings – which are almost "random" for a model that hasn't been fine-tuned. $r = -0.36$ and -0.4 for, respectively, the re-ranking and full-ranking settings. Note that the $(4, 6]$ and $(6, 8]$ bins contain the largest number of observations, such that the observed increase in variance is not the consequence of a lower number of samples (c.f. Chebyshev).

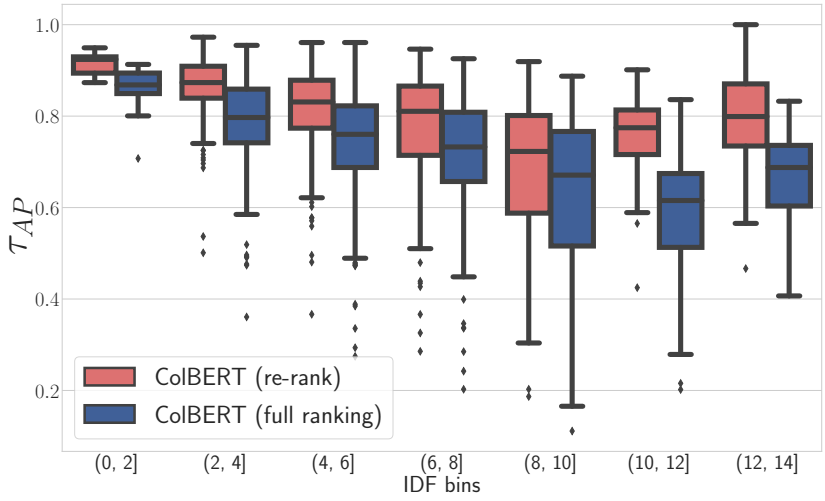


Table 4.1: Queries from TREC DL 2019-2020 for which one of the terms (in bold) has a low (`--`) or a high (`++`) ColBERT *importance* – the higher τ_{AP}, the less important. Values in parenthesis respectively represent the τ_{AP} and IDF of the corresponding term.

`--`	*exons definition* ***biology*** $(0.88, 6.42)$ *generic vivelle* ***dot*** $(0.85, 6.75)$ *cost*
`++`	*what are hydrocarbon in* ***lipids*** $(0.47, 6.93)$ ***hilton*** $(0.48, 7.95)$ *lifetime diamond member benefits*

ments *other query terms*. We observed that stopwords, in some cases, happen to match terms in documents that correspond to other query terms. For instance, in the query (and associated τ_{AP}) "*the* (0.94) *symptoms* (0.87) *of* (0.93) *shingles* (0.88)", the word "*of*" actually mostly matches with "*shingles*" in documents from $\mathcal{S}_q$. Additionally, terms with very large IDF (the two rightmost bins in Figure 4.4) appear by construction in fewer documents from $\mathcal{S}_q$ – e.g., terms with IDF = 12 only appear in around 50 MS MARCO passages. ColBERT thus *has* to rely on soft matches, for which scores (and thus, contributions) tend to have lower values (Section 4.3.3). This also explains the large difference between the re- and full-ranking settings observed for the last two bins. For the latter, ColBERT might be able to retrieve *semantically* related documents. Scores for such documents might still be impacted by these high IDF terms, thus boosting the importance of the term as defined

by τ_{AP}.

4.3.3 Matching Patterns in ColBERT

After showing how ColBERT implicitly encodes *term importance*, we now focus on how terms are actually matched by the model. Section 4.3.3.1 analyzes the distribution of exact and soft matching patterns in ColBERT, which can be explained from a representational perspective (Section 4.3.3.2).

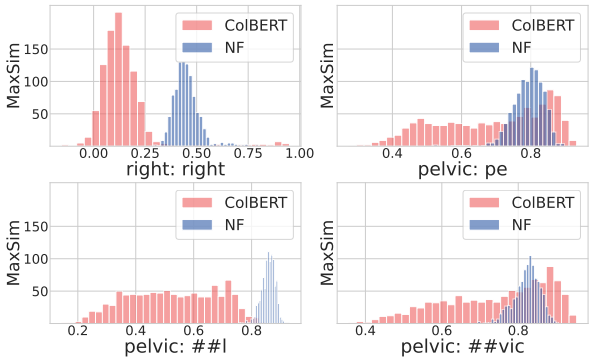
4.3.3.1 Analysis of Exact and Soft Matches

Given the ability of neural IR models to rely on soft (or semantic) matching, we wonder to which extent ColBERT still relies on exact (or lexical) matching. To this end, we propose to analyze, for each query token, the *distribution* of scores assigned to document tokens – either in $\mathcal{S}_{q,BM25}$ or $\mathcal{S}_{q,Col}$. We aim to assess whether these distributions exhibit different patterns, depending on the characteristics of the terms – such as IDF.

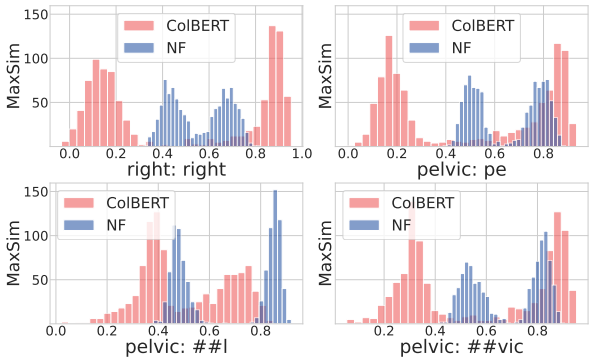
To illustrate our motivation, we show in Figure 4.5 the distribution of query token scores (i.e., MaxSim, or $C^{\star}_{id}$ in Eq. 4.1) – for four tokens of a TREC query. As a first observation, we notice how ColBERT tends to "flatten" the MaxSim distributions during fine-tuning (i.e., when compared to `NF`). Moreover, in the full-ranking setting, they seem to be bi-modal. It shows that ColBERT is more semantic than BM25, as it retrieves documents in which terms are assigned soft scores.

To refine these observations, we further differentiate when a score comes from an exact match with the query token – so the model has assigned the maximum score to the exact same token in the document – or a soft match – the maximum score comes from a match with a different token. We consider two types of score distribution per query token, as shown in the example in Figure 4.6. We first notice that, for both re-ranking and full ranking, some tokens that *seem* important in the query (e.g., "*##vic*" from the term "*pelvic*") tend to focus more on exact matching: their similarity scores tend to be higher w.r.t. to other tokens (e.g., "*causes*"), and w.r.t. to the soft case (on the right side of the plots). On the other hand, for tokens like "*causes*", the difference between the exact and soft cases seems to be lower, for instance, when comparing their mean – especially in the full ranking setting. This indicates that exact matching, in this case, does not play an essential part in ranking documents.

To generalize such observations, we define a measure indicating when ColBERT asserts whether a term favors an exact match over a soft one. More specifically, we compute, for each query token i, the difference between the average ColBERT score when i matches the same token j within a document (i.e, when $d^{\star}_i \rightarrow j$) or not (i.e., when $d^{\star}_i \not\rightarrow j$)



(a) Re-ranking



(b) Full ranking

Figure 4.5: MaxSim distributions, for four tokens of the TREC DL 19 query "*right pelvic pain causes*", for documents in $\mathcal{S}_{q,BM25}$ (re-ranking, 4.5a) and documents in $\mathcal{S}_{q,Col}$ (full ranking, 4.5b). Note how cosine similarities are positive, as a consequence of the max selection in ColBERT.

– as illustrated in Figure 4.7. We then average at query level to obtain one measure per term, for the ones appearing in several queries. This measure is formally defined as:

$$\Delta_{ES}(j) = \underset{i,q/i\to j}{\text{mean}} \left(\underset{d\in\mathcal{S}_q/d_i^\star\to j}{\text{mean}} \{C_{id}^\star\} \; - \underset{d\in\mathcal{S}_q/d_i^\star\not\to j}{\text{mean}} \{C_{id}^\star\} \right) \tag{4.4}$$

where $i \to j$ means that the i^{th} token corresponds to token j. Similarly to Section 4.3.2, for a term t, where $J \in q$ denotes the indices of its subword decomposition, we consider $\Delta_{ES}(t) = \sum_{j\in J} \Delta_{ES}(j)$, which corresponds to the way ColBERT operates (i.e., summing over subwords).

For each query term t, we plot in Figure 4.8 Δ_{ES} with respect to IDF. Higher Δ_{ES} tends to indicate that a matching value is higher if the term appears in the document (exact match), as the model learns to widen the gap (on average) between exact and soft scores.

For the re-ranking setting, we notice a moderate positive correlation between terms focusing more on exact matching – larger Δ_{ES} – and IDF. Interestingly, this effect can already be observed for a pre-trained

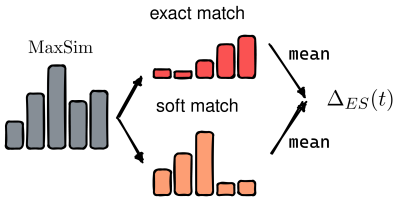
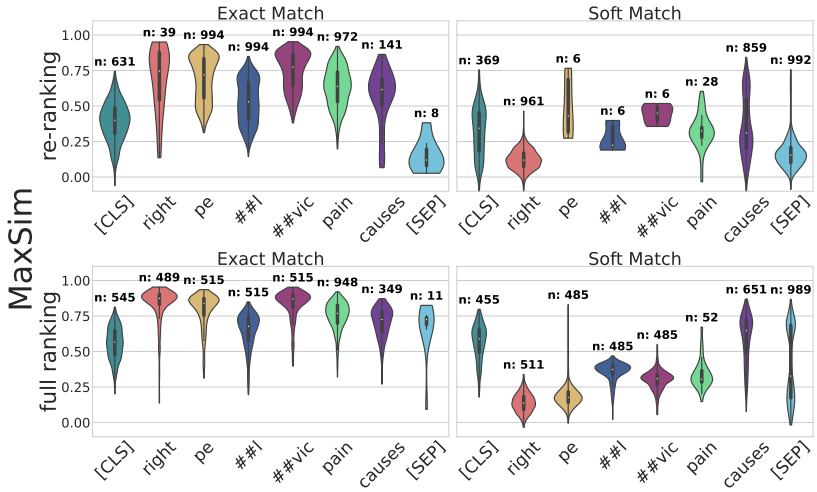


Figure 4.7: For each token (i.e., subword), we split the MaxSim distribution into *exact* and *soft* cases. We then quantify the gap between their average to measure how the model favors lexical matching (or not) for a given term.

Figure 4.6: Exact (*left*) and Soft (*right*) matching score distributions, for tokens in the TREC DL 19 query "*right pelvic pain causes*", for documents in $\mathcal{S}_{q,BM25}$ (re-ranking, *top*) and documents in $\mathcal{S}_{q,Col}$ (full ranking, *bottom*).



model ($r = 0.45$ for `NF`), but is, however, reinforced with fine-tuning ($r = 0.73$ for `ColBERT`). In particular, terms with a high IDF (above 8) are the most impacted: ColBERT thus *learns* to emphasize their lexical importance. For instance, for the query (and associated Δ_{ES}) "*causes* (0.35) *of* (0.11) *left* (0.64) *ventricular* (1.14) *hypertrophy* (1.62)", the model mostly relies on exact match for the last two terms – semantic matching has a lower contribution here. Note that the re-ranking setting has a lexical bias, as ColBERT is bound to re-order documents retrieved by BM25, usually containing "important" query terms. Thus it might be less critical for the model to rely on such signals[149].

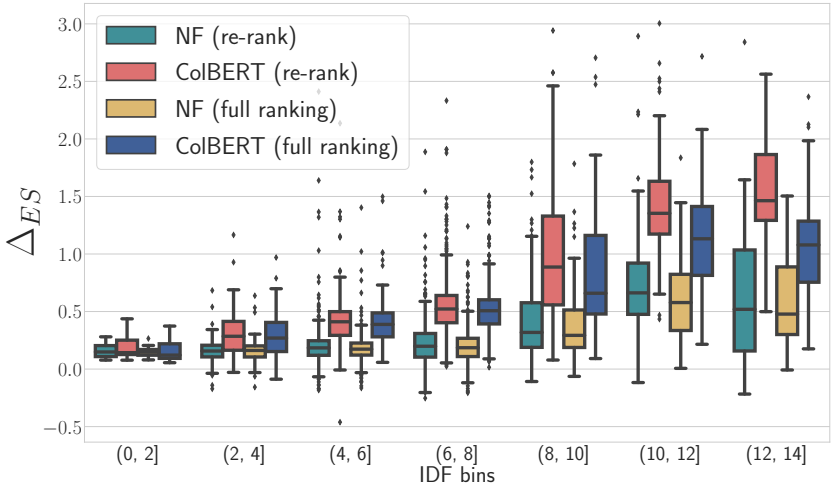
However, this property can also be observed in the full ranking setting ($r = 0.68$ and $r = 0.50$ for `ColBERT` and `NF` respectively) – highlighting the need for ColBERT to still partially take into account traditional retrieval heuristics when retrieving from the complete collection. Similarly to observations made in Section 4.3.2, variance increases with IDF, which illustrates the contextual nature of matching: some terms with high statistics are not as important as their IDF would tend to indicate when it comes to lexical match.

[149] Although several works have shown how cross-encoders still lean on exact matching [Jiang et al. 2021; Rau and Kamps 2022a].

4.3.3.2 A Representational Interpretation

To explain the patterns observed in Section 4.3.3.1, we hypothesize that exact matches correspond to tokens whose contextual representations do not *vary* much, i.e., lie in a narrow region of the embedding space. Hence, the cosine similarity between the query and document tokens would be closer to 1, and ColBERT will tend to select this token by construction. On the contrary, tokens that carry less "information" are more heavily influenced by their context – they act as some sort of

Figure 4.8: Δ_{ES} with respect to IDF (x-axis, binned). In the re-ranking setting, $r = 0.73$ and $r = 0.45$ for `ColBERT` and `NF`, respectively. In full ranking, $r = 0.68$ and $r = 0.50$ for `ColBERT` and `NF`, respectively.



"reservoirs" to encode concepts of the sequence – such that their embeddings vary more. To confirm this hypothesis, we conduct a spectral analysis of contextual embeddings. More specifically, for a given token, we perform Singular Value Decomposition on the matrix composed of its contextualized representations – collected on passages from the MS MARCO collection – as illustrated in Figure 4.9. SVD provides a way to factorize any real matrix into singular values and singular vectors. More specifically, for a $N \times d$ matrix[150] of representations R, we can decompose it as:

$$R = U\Sigma V^T \tag{4.5}$$

where U, Σ and V are a $N \times N$, $N \times d$ and $d \times d$ matrices respectively. In particular, U and V are orthogonal, while Σ is rectangular diagonal, and the elements $\sigma_1 \geq ... \geq \sigma_d$ alongside its diagonal correspond to the singular values. By inspecting their relative magnitude, we can assess how "spread" the representations are to some extent. If the magnitude of σ_1 is much larger than the others, it means that all the vectors point to the same general direction in the embedding space. In other words, it indicates how well a contextualized representation could be replaced by a static one [Ethayarajh 2019].

In Figure 4.10, we report the ratio of σ_1^2 over $\sum_k \sigma_k^2$, with respect to subword IDF[151], for tokens appearing in TREC queries. Results confirm our hypothesis, as the ratio increases with the subword IDF ($r = 0.84$). Moreover, this effect is much stronger after fine-tuning, indicating how ColBERT adapts its representation space to accommodate lexical matching ($r = 0.70$ for `NF`). In particular, words with a low IDF tend to point in different directions, showing that what they cap-

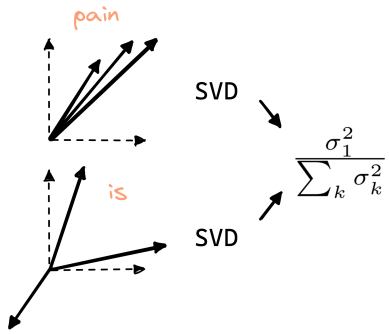


Figure 4.9: Illustration of the approach. For each term in the TREC queries (here, "*pain*" and "*is*"), we collect representations on the MS MARCO collection. Each matrix is then decomposed with SVD.

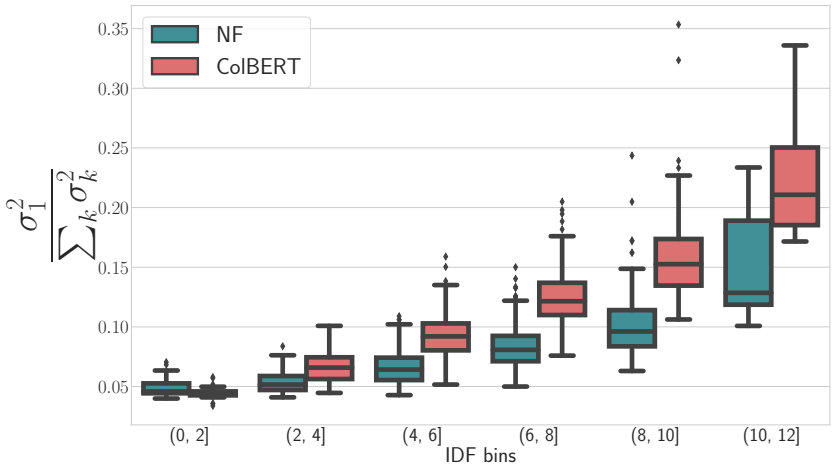
[150] N denotes the number of contextual representations, and d the dimension of representation ($d = 128$ for ColBERT)

[151] Note that as we directly consider representations, we operate at the subword level, and estimate collection statistics based on WordPiece tokenization.

ture is more about their context. For instance, in the query "*when did family feud come out ?*"[152], the token "*come*", for all the documents in $S_{q,BM25}$, matches 97% of the time to tokens that are not in the query but are synonyms (in a broad sense), e.g., {"*happen*","*released*", "*name*", "*going*", "*rodgers*"}. Such tokens are thus more dedicated to performing semantic matching.

[152] a TV show.

Figure 4.10: $\frac{\sigma_1^2}{\sum_k \sigma_k^2}$, with respect to WordPiece IDF (x-axis, binned). $r = 0.70$ and 0.84 for, respectively, `NF` and `ColBERT`.



4.4 Conclusion

Based on a careful analysis of its matching process, we have shown how ColBERT – despite its semantic nature – *implicitly* captures a notion of term importance (Section 4.3.2) and relies on exact matches for *important* terms (Section 4.3.3.1). The latter can be explained by the learned embedding space, in which representations of important terms tend to "vary" less (Section 4.3.3.2). This questions the very notion of term matching for models based on Pre-trained Language Models: is this behavior truly learned or merely memorized for terms seen at training time? We, therefore, delve deeper into this question in Section 4.5.

Additionally, the results of this study initiated a series of experiments around models that *explicitly* encode such phenomena *by design* – eventually resulting in the SPLADE model (Chapter 3).

4.5 Lexical Match Analysis

Context

This Section summarizes our analysis of lexical matching in neural IR models [Formal et al. 2022b,c]. In particular, we show how such behavior is tied to the various existing architectures and how models built on a lexical prior tend to generalize term matching better. Additionally, we study the impact of training frequency on the ability to retrieve query terms precisely.

While neural retrievers based on Pre-trained Language Models have fully shone on in-domain datasets like MS MARCO on which models have been trained, some models – especially dense bi-encoders – have recently been challenged on zero-shot settings (such as the BEIR benchmark), questioning their actual generalization capabilities when compared to traditional Bag-of-Words approaches. We wonder if these shortcomings could partly result from the inability of neural IR models to perform lexical matching off-the-shelf.

In this Section, we propose a discrepancy measure between the lexical matching performed by any (neural) model and an "ideal" one. Contrary to Section 4.3 – where we have devised specific indicators for the ColBERT model – this approach is model-agnostic. It is also very intuitive: we simply propose to keep track of statistics about query terms appearing in top documents retrieved by the systems (Section 4.5.1). Based on this, we study the behavior of different state-of-the-art neural IR models, and assess whether they can perform lexical matching *when it's actually useful*, i.e., for important terms (Section 4.5.3).

We hypothesize that generalization is linked – at least partly – to the ability of models to properly handle exact matching. Indeed, [Thakur et al. 2021] have shown that the only systems improving the overall performance over BM25 in the zero-shot setting have (somehow) a lexical bias. SPLADE also achieves impressive performance on zero-shot evaluation (Section 3.8). Therefore, we also propose to study in Section 4.5.4 the extent to which neural IR models can *generalize* lexical matching for query terms that have either not been seen in the training set or with different collection statistics – e.g., rare in the training set but common on an out-of-domain evaluation set.

4.5.1 Methodology

Our analysis rationale is the following: the more a term is important for a query (w.r.t. relevant documents), the more frequently it should be retrieved by the system for documents in top positions. Therefore, we first need to define what it means for a term to be *important for lexical matching*, and how to accurately measure its frequency in top

documents. Roughly speaking, we are interested in the models' ability to retrieve documents containing query terms, *when they are deemed important*.

Note that we are not interested in expansion mechanisms in our analysis since they are more related to semantic matching – for which the analysis is more intricate. Intuitively, term importance w.r.t. relevance can be measured by the extent to which a term allows to distinguish relevant from non-relevant documents in a collection of documents. It is thus natural to use the Robertson-Sparck Jones (RSJ) weights [S. E. Robertson and Jones 1976; C. T. Yu and Salton 1976]. The RSJ weights have been shown, if estimated correctly, to order documents in the optimal order w.r.t. the Probability Ranking Principle [Stephen E. Robertson 1977] – when the score of a document d is obtained by summing the RSJ weights of query terms appearing in d. For a given user information need U, the user RSJ_U weight for term t is defined as follows – the conditioning on query q being implicit:

$$\text{RSJ}_{t,U} = \log \frac{\mathcal{P}(t|R)\mathcal{P}(\neg t|\neg R)}{\mathcal{P}(\neg t|R)\mathcal{P}(t|\neg R)} \tag{4.6}$$

where $\mathcal{P}(t|R)$ (resp. $\mathcal{P}(t|\neg R)$) corresponds to the probability that term t occurs in a relevant (resp. non-relevant) document. $\text{RSJ}_{t,U}$ is thus high when a term, for a document to be relevant, is both *necessary* ($\mathcal{P}(.|R)$) and *sufficient* ($\mathcal{P}(.|\neg R)$). The above probabilities can be estimated from relevance annotations and collection statistics as follows:

$$\tilde{\mathcal{P}}(t|R) = \frac{c(t,R)}{|R|}$$
$$\tilde{\mathcal{P}}(t|\neg R) = \frac{c(t,\mathcal{C}) - c(t,R)}{|\mathcal{C}| - |R|}$$

where $c(t,R)$ is the number of *relevant* documents containing term t, $c(t,\mathcal{C})$ the number of documents (in the collection) containing t, and $|R|$ is the total number of relevant documents for the considered query q. We give a few examples of estimated $\text{RSJ}_{t,U}$ in Table 4.2. We see how it is low for e.g. stopwords, as they have equal *odds* to appear in relevant and irrelevant documents. On the contrary, it tends to increase for what seems to be *important* query terms. Additionally, note that RSJ weights and IDF are correlated – as shown on Figure 4.11. The former are estimated from relevance annotation, while the latter are related to term importance estimation. Indeed, under certain assumptions and simplifications, IDF can be derived from RSJ (see for instance [Metzler 2008]). However, for our analysis purpose, RSJ weights are perfectly suited and should provide more accurate estimates of how discriminative a term is[153].

We now want to compute the same weight, when relevance is defined by the *system* (and not the *user*). In other words, we would like

[153] Note that analyses from Section 4.3 could have been based on RSJ (rather than IDF). We didn't make this connection at that time. Note that, however, observations remain similar (plots not shown here) – and are actually reinforced.

Table 4.2: Queries from TREC DL 2019-2020 and associated $\text{RSJ}_{t,U}$ – estimated from collection statistics and relevance annotations.

RSJ_U
do (1.1) *goldfish* (13.6) *grow* (6.2)
right (6.7) *pelvic* (5.4) *pain* (7.2) *causes* (4.9)
definition (3.5) *of* (0.5) *laudable* (14.1)
who (0.5) *sings* (3.9) *monk* (8.7) *theme* (6.1) *song* (5.4)
rsa (12.7) *definition* (1.2) *key* (6.1)

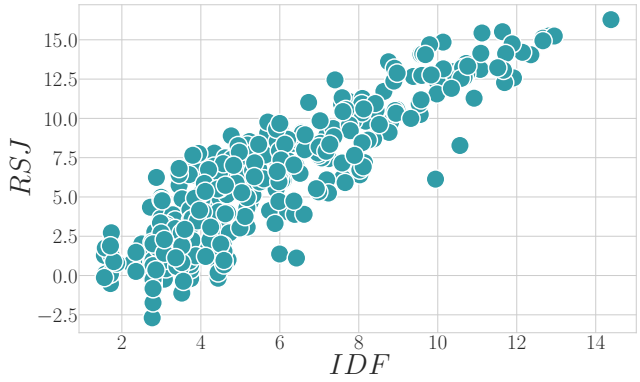


Figure 4.11: RSJ *vs* IDF, for terms of TREC DL 2019-2020 queries. Pearson $r = 0.90$.

to measure how much a model "retrieves" a given term t. One way to proceed is to suppose that top-K documents are *relevant from the point of view of the system*, for a suitable K. Different definitions of system relevance could be used – we however found this approximation to be both simple yet well-sounded. We also observed from preliminary analyses that results are not very sensitive to the choice of K. We hence define the system RSJ_S weight for term t as:

$$\text{RSJ}_{t,S} = \log \frac{\mathcal{P}(t|\text{top-}K)\mathcal{P}(\neg t|\neg\text{top-}K)}{\mathcal{P}(\neg t|\text{top-}K)\mathcal{P}(t|\neg\text{top-}K)} \tag{4.7}$$

Intuitively, it gives us a means to properly "count" occurrences of query terms in retrieved documents – taking into account collection statistics. It is estimated similarly to Eq. 4.6.

Once RSJ_U and RSJ_S have been estimated, we can look at the difference between both, i.e., $\Delta\text{RSJ}_t = \text{RSJ}_{t,S} - \text{RSJ}_{t,U}$. If $\Delta\text{RSJ}_t > 0$ (resp. $\Delta\text{RSJ}_t < 0$), it tends to indicate that the model overestimates (resp. underestimates) the importance of term t when considering its document ordering. In other words, the model retrieves "too much" (resp. "too few") this term. Please note that a high correlation between RSJ_S and RSJ_U *is not* indicative of the absolute performance of a model, as RSJ_U is neither a perfect model nor performance measure[154]. Besides lexical matching, other factors need to be taken into

[154] For instance, terms are assumed to be independent.

account – especially semantic matching – to fully characterize models' effectiveness. However, we argue that it can still partly indicate the performance of the model w.r.t. lexical matching, especially for terms whose RSJ_U are high.

4.5.2 Experimental Setting

We conduct experiments by analyzing models trained on MS MARCO. Following Section 4.3, we evaluate models on the *in-domain* TREC Deep Learning 2019-2020 datasets (97 annotated queries in total)[155], and two *out-of-domain* datasets from the BEIR benchmark: *i*) `TREC-COVID`, a bio-medical collection containing 50 queries (described in Section 2.3.2.2), *ii*) `FiQA-2018`, an opinion-based Question Answering dataset oriented towards finance, containing 648 queries. Particularly, these two datasets contain specific terms that contrast with the ones from MS MARCO.

For all our experiments, we measure the system relevance by using[156] top-$K = 100$. For the term-level analysis, we keep stopwords, and use standard tokenization with Porter stemming. Furthermore, we only consider first-stage retrievers (and not re-rankers), as our main focus is the ability of neural models to *retrieve* documents containing important query terms. We thus propose to compare the different "families" of models, that we expect to behave differently w.r.t. lexical matching. We, therefore, consider several state-of-the-art models based on the BEIR benchmark, including two purely lexical models, BM25 and doc2query-T5; SPLADE[157]; ColBERT[158], and two dense bi-encoders, TAS-B[159] as well as a "standard" model trained with contrastive loss and In-Batch Negatives.

[155] Note that to accurately estimate the RSJ weights, we need "deep" annotations – the analysis cannot be transposed for instance on MS MARCO dev queries.

[156] Similar conclusions can be made for different values of K, ranging from 10 to 1000.

> **Remark**
>
> We have shown in Section 4.3 how ColBERT implicitly relies on term matching. Additionally, SPLADE (Chapter 3) has a lexical prior by design. On the contrary, dense models are expected to be more semantic. We show in Section 4.5.3 and 4.5.4 a certain "hierarchy" between architectures when it comes to exactly matching important query words.

4.5.3 Lexical Match in Neural IR

We are interested in the impact of training frequency on lexical matching. In Section 4.5.3.1, we first conduct the analysis on terms that appear in the training set (In-Training). Section 4.5.3.2 focuses on query terms (almost) not seen at training time (Out-of-Training).

[157] `CoCondenser-EnsembleDistil` from Section 3.7, available at https://huggingface.co/naver/splade-cocondenser-ensembledistil.

[158] The same model used in Section 4.3.

[159] Available at sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco.

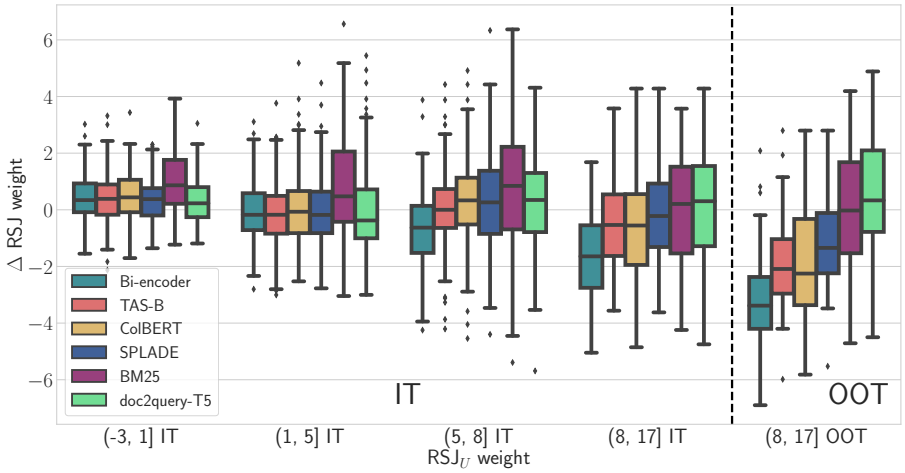


Figure 4.12: ΔRSJ with respect to user RSJ$_U$ (x-axis, binned), splitting according to query terms seen during training (`IT`, left) or not (`OOT`, right). We consider that terms appearing in less than 10 training queries (out of $> 500k$) are `OOT`, leading to 499 and 42 terms in TREC queries, for `IT` and `OOT` respectively. Note that due to the fact that `OOT` terms are also generally rare in the collection, their RSJ$_U$ are always > 8 – hence the single bin.

4.5.3.1 In-Training

We first focus on *in-domain* lexical matching, by requiring terms to have appeared in a sufficient number of training queries. In Figure 4.12, we plot the relationship between the user weight and ΔRSJ, for each term in the test queries appearing at least 10 times in the training queries (left, `IT` for In-Training). We first note that lexical-based models (BM25 and doc2query-T5) tend to overestimate the importance of query terms ($\Delta\text{RSJ} > 0$) for the complete spectrum of RSJ$_U$. The second observation is that all models are roughly similar in their estimations for low user RSJ$_U$ weights (below 5).

There is an overall decreasing trend for neural models, which tend to underestimate term importance as RSJ$_U$ increases. More specifically, there is a clear distinction between the bi-encoder and other neural models (both dense and sparse): we can see that the former retrieves fewer documents, on average, containing precisely the important query terms. A hierarchy between models also seems to emerge: in terms of lexical matching, dense bi-encoders fall short, followed by ColBERT and SPLADE[160]. This result is kind of expected, as the two latter are somewhat based on lexical priors. Comparing dense and sparse/interaction models overall – by considering the average ΔRSJ over terms – we observe that, interestingly, dense models underestimate RSJ$_U$ ($\overline{\Delta\text{RSJ}} = -0.07$ for TAS-B and -0.26 for the bi-encoder) while sparse/interaction slightly overestimate it ($\overline{\Delta\text{RSJ}}$ around 0.03 for ColBERT and SPLADE). Note again, as mentioned in Section 4.5.1, that the measure is not necessarily indicative of performance: for in-

[160] Although TAS-B seems to reach levels similar to ColBERT.

stance, TAS-B performs better than BM25 (71.7 *vs* 50.6 nDCG@10 on TREC DL 19), suggesting that the former is better for semantic search.

To illustrate the above, let us consider the two examples in Table 4.3 and Table 4.4. For the first query, the two-dense bi-encoders underestimate the importance of both "*rsa*" and "*key*". On the other hand, BM25 and doc2query-T5 greatly overestimate their importance, while ColBERT and SPLADE are similarly slightly above. Results are overall aligned on the second example: dense bi-encoders are still underestimating importance – although BM25, as well as neural approaches, also tend to underestimate importance – for instance, for the term "*statutory*".

Finally, ΔRSJ for BM25 seems to have a much larger variance for intermediate RSJ_U values compared to neural models. We hypothesize that training models enables a better estimation of term importance.

Table 4.3: RSJ estimation for the TREC DL 2019 query "*rsa definition key*". RSJ_U denotes the "ground truth" term importance, and we indicate such estimation from the point of view of each system (RSJ_S).

`model`	RSJ		
	rsa	*definition*	*key*
RSJ_U	12.7	1.2	6.1
RSJ_S(BM25)	14.3	0.8	7.8
RSJ_S(doc2query-T5)	13.1	0.2	8.5
RSJ_S(Bi-encoder)	10.4	0.8	5.3
RSJ_S(TAS-B)	11.8	0.8	5.9
RSJ_S(ColBERT)	12.8	0.5	7.2
RSJ_S(SPLADE)	13.6	0.2	7.1

Table 4.4: RSJ estimation for the TREC DL 2020 query "*what is statutory deed*". RSJ_U denotes the "ground truth" term importance, and we indicate such estimation from the point of view of each system (RSJ_S).

`model`	RSJ			
	what	*is*	*statutory*	*deed*
RSJ_U	0.1	1.9	10.4	9.9
RSJ_S(BM25)	1.8	0.9	8.9	9.6
RSJ_S(doc2query-T5)	0.1	1.0	8.5	10.5
RSJ_S(Bi-encoder)	0.1	2.3	7.0	5.9
RSJ_S(TAS-B)	0.1	1.9	7.9	8.3
RSJ_S(ColBERT)	0.0	2.4	10.2	6.8
RSJ_S(SPLADE)	0.0	1.8	10.0	8.9

4.5.3.2 Out-of-Training

We now shift our attention to the behavior of models for query words that are *(almost) not* in the training set. In Figure 4.12, we show the distribution of ΔRSJ for terms appearing in less than 10 training queries (out of $> 500k$) – right, `OOT` for Out-of-Training. Comparing with ΔRSJ for terms in the training set, we can see that all neural models are affected somehow, showing that lexical match does not fully generalize to "new" terms. As should be expected, BM25 is not impacted. More specifically, when considering the $(8, 17]$ bin, and for every model (except BM25), the difference in mean between `IT` and `OOT` is significant, based on a t-test with $p = 0.01$. We also note that the trend observed earlier becomes clearer: models with a lexical prior, such as SPLADE, better encode lexical matching, and this holds even for "new" terms.

Finally, we also look at the relationship between `IT` and `OOT`, and model performance. More precisely, for terms in the $(8, 17]$ bin, we compute the mean nDCG@10 for queries containing at least one term either in `IT` or `OOT` (respectively 55 and 37 queries out of the 97, with 9 queries in both sets). We find that BM25 and doc2query-T5 performance increases by 0.1 and 0.02, respectively, while there is a performance loss for all neural models (≈ 0 for TAS-B, -0.11 for SPLADE, -0.27 for the bi-encoder and -0.38 for ColBERT). The fact that BM25 performance increases is likely due to the fact that the mean IDF increases (from 7.3 to 10.9), i.e., important terms are more discriminative in the `OOT` query set. We give in Table 4.5 an example of a query, for which a term ("*theraderm*") appears in the `OOT` set. We see that the *relative* performance loss in terms of nDCG@10 can, to some extent, be attributed to a large ΔRSJ. Note that it is not the case on every query, and it is additionally difficult to obtain a query-level measure of "lexical drop"[161]. With this in mind, the decrease of all neural models might still suggest that a potential reason for the relative performance loss (w.r.t. BM25) is due to a worse estimate of high RSJ_U. This raises important questions regarding generalization, which we further explore in the following Section.

4.5.4 Lexical Match and Zero-Shot Transfer Learning

We now analyze whether lexical match can generalize to the zero-shot setting[162]. More specifically, how do models deal with lexical match for *i*) new domain words – for instance, medical words not seen at training time (related to Section 4.5.3 and `OOT` words), *ii*) "shifted" words, i.e., words for which collection statistics changed? We, therefore, distinguish two categories of terms, namely those which occur 5 times more in the target collection than in MS MARCO (`IDF+`), or those for which term statistics were more preserved (`IDF-`), allowing us to split query terms into sets of roughly equal size[163]. Since term importance is related to

[161] We have, for instance, tried to sum, average, or max out the ΔRSJ for each query – without noticing a significant correlation with performance loss. Note that effectiveness is also linked to other factors such as semantic matching. Also note that predicting the performance of retrieval systems has a long history in IR [Carmel and Yom-Tov 2010], and is known to be a difficult task.

[162] We exclude doc2query-T5 from the analysis, due to *i*) the high computation cost for obtaining the expanded collections, *ii*) the rather similar behavior when compared to BM25 in Section 4.5.3.

[163] Note that the reverse case, i.e., terms for which IDF decreases happens far less.

Table 4.5: ΔRSJ for the term "*theraderm*" (OOT) in the query "*what is theraderm used for*". We also indicate the *relative* performance loss w.r.t. the queries in the IT set. Note that for all models (but the bi-encoder), there is actually a performance *increase*.

model	$\Delta\text{RSJ}(\textit{theraderm})$	$\Delta_R(\text{nDCG@10})$
BM25	-2.3	$\uparrow$ 1.05
doc2query-T5	-2.3	$\uparrow$ 0.46
TAS-B	-2.4	$\uparrow$ 0.37
SPLADE	-3.2	$\uparrow$ 0.33
ColBERT	-3.9	$\uparrow$ 0.27
Bi-encoder	-4.5	$\downarrow$ 0.59

collection frequency (Figure 4.11), we can compare ΔRSJ in those two settings.

Figure 4.13 shows the ΔRSJ with respect to RSJ_U for the TREC-COVID and FiQA-2018 collections. We first observe that BM25 similarly overestimates term importance on both datasets (except on the $(5, 13]$ bin on FiQA) and is not sensitive to the gap in collection statistics. Note that this is expected: BM25 does not rely on any training and is thus quite invariant to train-test distribution shifts. It additionally "adapts" to new collections when estimating IDF. Neural models on the other hand – which solely transfer from MS MARCO – tend to underestimate RSJ_U for terms that are more frequent in the target collection than in the training one (IDF+). It might indicate that models have learned dataset-specific term importance – confirming the results obtained in Section 4.5.3 on OOT terms. When comparing dense and sparse/interaction models overall – by considering the average ΔRSJ over terms – we observe that dense models underestimate even more RSJ_U than on in-domain ($\overline{\Delta\text{RSJ}} = -0.17$ for TAS-B and -0.38 for the bi-encoder) while sparse/interaction seem to overestimate ($\overline{\Delta\text{RSJ}} = 0.18$ for ColBERT and 0.30 for SPLADE), but however to a lesser extent than BM25 ($\overline{\Delta\text{RSJ}} = 0.83$). Furthermore, we notice the same pattern observed earlier, where SPLADE $\succ_{\texttt{lex}}$ ColBERT $\succ_{\texttt{lex}}$ dense bi-encoders – particularly strong on FiQA-2018. Overall, SPLADE is less impacted by distribution shifts when it comes to term matching. These results are in line with its overall good performance in zero-shot evaluation settings (Section 3.8). Finally, we observe that when transferring, all the models have a higher ΔRSJ variance compared to what is observed on MS MARCO. In all cases, the standard deviation – when normalized by the one for BM25 – is around 0.8 for MS MARCO, but around 1.1 for TREC-COVID and FiQA-2018. This further strengthens our point on the issue of generalizing lexical matching to out-of-domain collections[164].

[164] Other studies, such as [Sciavolino et al. 2021b; Tänzer et al. 2022], have drawn similar conclusions.

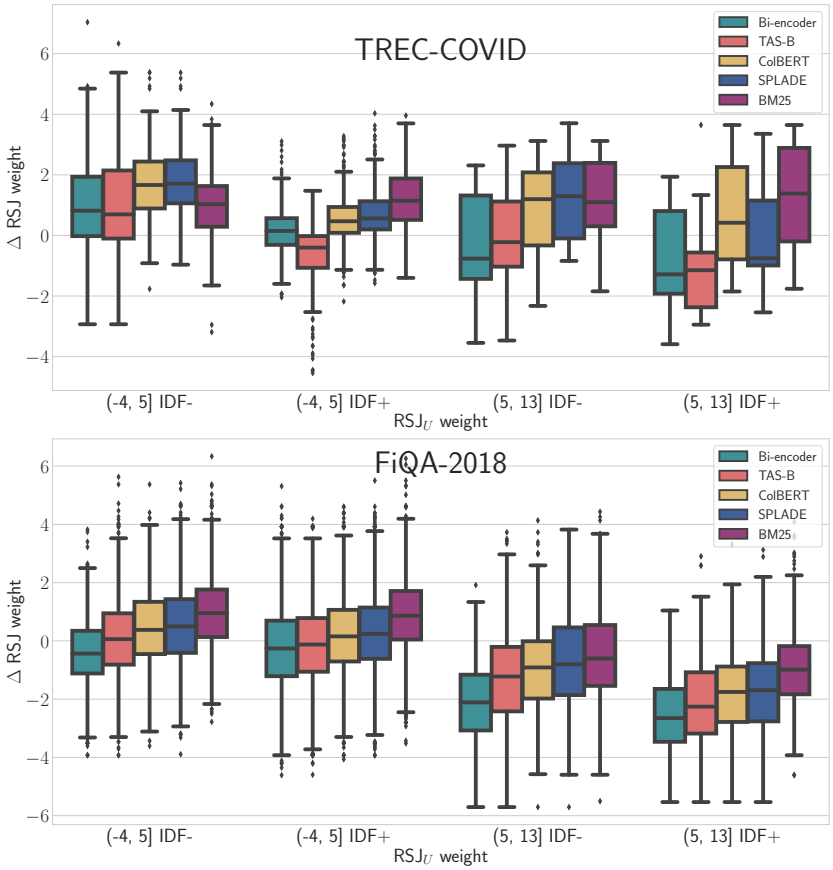


Figure 4.13: ΔRSJ with respect to RSJ$_U$ (x-axis, binned) in the zero-shot setting. `IDF-` includes 108 and 933 terms, while `IDF+` includes 112 and 428 terms for respectively `TREC-COVID` and `FiQA-2018`. Note that bins are not similar compared to Fig. 4.12, as RSJ weights have different distributions on BEIR datasets.

4.6 Conclusion

We have introduced a model-agnostic approach – based on the Robertson-Sparck Jones weights – that enables a systematic comparison of different neural ranking systems with respect to term matching (Section 4.5.1). Overall, we have shown that the ability to perform lexical matching off-the-shelf is architecture-dependent and is heavily influenced by the presence of query terms in the training set (Section 4.5.3). The rarer the term, the harder it is to find documents containing it for most neural models. Furthermore, this phenomenon is amplified if term statistics change across collections, for instance, in out-of-domain settings (Section 4.5.3.2). This suggests that models learn keyword matching in a dataset-specific manner and further highlight their limitations in dealing with tail queries in real-world scenarios.

Chapter 5

Conclusion

In the course of this Ph.D., beginning in late 2019, we have witnessed and, to a certain extent, contributed to the undergoing paradigm shift in Information Retrieval. Pre-trained Language Models have impacted the field beyond expectation, to the point that neural ranking models have become real competitors for well-established term-based approaches such as BM25 – opening the path for the next generation of semantic search systems. This paradigm has strongly supported our answers to the two research questions:

- **RQ1** Can we design *effective*, *efficient*, *interpretable* and *robust* neural IR models to replace proven traditional lexical approaches?
- **RQ2** Despite their semantic nature, do neural ranking models based on PLM rely on well-established IR properties?

5.1 Summary of Contributions

In addition to a review of the current state of IR research (Chapter 2), we can summarize the main findings of this thesis as follows:

RQ1 In Chapter 3, we have proposed an original[165] approach – dubbed SPLADE – to the retrieval problem, by representing queries and documents as sparse vectors in the vocabulary space. SPLADE is *effective*, *efficient*, and its representations can be *interpreted* by design. It unifies expansion and term weighting in a simple framework and allows controlling the effectiveness-efficiency trade-off in an end-to-end fashion. SPLADE furthermore achieves state-of-the-art results in zero-shot evaluation settings when combined with training techniques such as distillation or hard-negative sampling – highlighting its *robustness* when facing distribution shifts.

[165] At that time, dense models were on a roll, while sparse approaches – due to their lack of effectiveness – were not so popular.

RQ2 In Chapter 4, we have analyzed neural ranking models from an IR perspective. In particular, we have focused on two specific behavior,

namely *lexical match* and *term importance*. We have first designed indicators tailored explicitly for the ColBERT model and showed how it still implicitly relies on such aspects – despite its semantic nature. In a second contribution, we have shown that the ability to perform lexical matching is architecture-dependent and is heavily influenced by the presence of query terms in the training set. This suggests that neural models learn keyword matching in a dataset-specific manner and further highlights their limitations in dealing with tail queries for real-world scenarios. SPLADE, however, is better at handling such shifts when it comes to precisely matching query terms.

5.2 Perspectives

Our answers to **RQ1** and **RQ2** are incomplete, and we are still far from having solved "search". Within the last three years, there has been an increasing number of researchers contributing to the field, making overall progress fast, at the expense of individual contributions, which are sometimes diluted in the mass. We have proposed with SPLADE an original and impactful approach that has proven effective in many scenarios. In the following, we give a light overview of various works that have, to some extent, built on SPLADE. We group those into what we believe to be promising perspectives in the neural IR space.

Towards More Efficient Learned Sparse Retrieval As discussed in Section 3.5.5.1, a line of work has studied the limitations of traditional index-based search algorithms for learned sparse models. [Mackenzie et al. 2021, 2022b] show how SPLADE generates "wacky" weights that greatly impact the retrieval time of standard *document-at-a-time* techniques. [Mackenzie et al. 2022a; Mallia et al. 2022; Qiao et al. 2022] therefore propose various adaptations of existing query processing techniques to accommodate with the specificities of this new class of models. [E. Choi et al. 2022; Lassance and Clinchant 2022] tackle efficiency aspects from an architecture point-of-view, by designing lighter SPLADE query encoders. [J.-H. Yang et al. 2021] extend SPLADE with a top-k masking scheme to better control the sparsity on both query and document sides. In contrast, [Shen et al. 2022a] show how a simple *post-processing* top-k pruning strategy allows reducing both query latency and index size. This suggests that SPLADE's regularization constraint (either ℓ_1 or $\ell_{\texttt{FLOPS}}$) is not completely aligned with what makes a model truly efficient. More broadly, inverted indexes have been vastly optimized for lexical approaches, based on the statistical properties of natural language, which differ from what we observe with learned sparse representations. Instead of bridging this gap, [Bruch et al. 2023] propose a novel approximate solution for retrieval over sparse real-valued vectors drawn from arbitrary distributions, laying the ground for future

research directions in this regard. From a training perspective, [Pal et al. 2023] propose to study the role of adapters [Houlsby et al. 2019] for parameter-efficient training and transfer learning of SPLADE models.

Towards Unifying Dense and Sparse Models Current IR research tends to oppose sparse and dense approaches, which inherit different behavior from their respective architectural biases. They both focus on different aspects of relevance (i.e., somehow lexical *vs* semantic) – thus complementing each other. Many works have shown how combining results from traditional lexical models with those from dense approaches is usually beneficial, especially in zero-shot settings [Bruch et al. 2022; Tao Chen et al. 2022; H. Li et al. 2022b; Luan et al. 2021; Xueguang Ma et al. 2021; S. Wang et al. 2021]. Following works have therefore bridged the gap between learned sparse and dense bi-encoders. [K. Zhang et al. 2022] propose to align a dense retriever with a SPLADE-like model to boost its effectiveness. [Shen et al. 2022b] additionally unifies dense and sparse bi-encoders learning into a single framework – dubbed UnifieR – to take advantage of their complementary views. More recently, [Ram et al. 2022a] propose interpreting dense retrievers by projecting document and query vectors into the vocabulary space, based on the MLM head. Despite these results, the interplay between sparse and dense approaches remains yet to be fully uncovered. It is, however, possible to foresee a future for *hybrid* search systems based on learned retrievers.

Towards Universal Sparse Retrieval Learning sparse representations combined with an implicit entropy regularization has shown to be an effective and robust approach for the ad-hoc IR task. It is, therefore, natural to question whether it could apply to other settings. In our recent contribution, we show, for instance, how SPLADE can be leveraged for Conversational Search [Hai Le et al. 2023]. In the context of cross-lingual retrieval – where queries and documents come from different languages – [Nair et al. 2022] introduce SPLADE-X, a cross-language expansion model based on a multilingual version of BERT. From a different perspective, [Iida and Okazaki 2022] propose an unsupervised domain adaptation technique specifically tailored for SPLADE, by filling vocabulary and word-frequency gaps – improving its effectiveness when facing domain shifts.

We can see emerging in the above works the question of a *representation basis* for SPLADE[166]. We hypothesize that depending on the task, the model could take advantage of different "indexing units": specific terms in out-of-domain settings, tokens from other languages, or even more abstract concepts or topics[167]. For instance, [C. Chen et al. 2023; luo et al. 2023] extend the approach to cross-modal retrieval by mapping images and text to a sparse vocabulary space. We can even imagine

[166] Recall that SPLADE is limited to representing queries and documents in the WordPiece vocabulary, due to the nature of BERT pre-trained MLM head. However, [Lassance et al. 2023], study the impact of pre-training on SPLADE – allowing to control the vocabulary, and therefore, the representation space.

[167] Think, for instance, about the latent terms in SNRM [Zamani et al. 2018].

extending to pure image retrieval, where "tokens" could, for instance, correspond to visual patches. Given the recent trend around Retrieval-Enhanced Machine Learning in various domains (Section 2.6.6) – for which dense approaches remain the norm – we also hypothesize that generic sparse retrievers constitute an exciting research direction.

Towards Predictable Neural Ranking Models In Chapter 4, we have shown that we can associate – albeit loosely – effectiveness drops of various neural models with their inability to retrieve important query terms in zero-shot settings[168]. More generally, in the case of supervised models, we hypothesize that predicting the behavior of a model should be linked to training statistics. To this end, we have recently shown how zero-shot performance loss correlates with lexical and semantic "similarity" indicators [Lupart et al. 2023]. Overall, given the training queries, the closer the test queries, the better the performance. This further poses the question of query *difficulty* for supervised ranking models: is this merely an effect of memorization, or are there queries intrinsically more difficult from their point of view? This has motivated us to turn our attention to the Query Performance Prediction (QPP) task in IR, which focuses on predicting the possible performance of a retrieval method on a given input user query [Carmel and Yom-Tov 2010]. This setting is particularly appealing in out-of-domain settings, as we would like to be able to predict system performance for "new" queries or even collections – for instance, tail queries in real-world search engines. The IR community has come up with a large number of techniques dedicated to predicting the performance of traditional lexical approaches prevailing in the pre-BERT era. We show in a recent work that existing methods are not suited to predict the performance of neural ranking models based on PLM [Faggioli et al. 2023], highlighting the need for indicators designed for this new class of models. In addition to having *interpretable* models, we seek models that are *predictable*.

[168] We, however, still lack the complementary aspect of semantic matching to characterize their performance fully.

Chapter 6

References

Akkalyoncu Yilmaz, Zeynep, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin (Nov. 2019a). "Applying BERT to Document Retrieval with Birch". 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): System Demonstrations*. Hong Kong, China: Association for Computational Linguistics, pp. 19–24. DOI: 10.18653/v1/D19-3004. URL: https://aclanthology.org/D19-3004.

Akkalyoncu Yilmaz, Zeynep, Wei Yang, Haotian Zhang, and Jimmy Lin (Nov. 2019b). "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)*. Hong Kong, China: Association for Computational Linguistics, pp. 3490–3496. DOI: 10.18653/v1/D19-1352. URL: https://aclanthology.org/D19-1352.

Amati, Gianni and Cornelis Joost Van Rijsbergen (Oct. 2002). "Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness". In: *ACM Trans. Inf. Syst.* 20.4, pp. 357–389. ISSN: 1046-8188. DOI: 10.1145/582415.582416. URL: https://doi.org/10.1145/582415.582416.

Amini, Massih-Reza and Éric Gaussier (2013). *Recherche d'Information - applications, modèles et algorithmes*. Eyrolles. ISBN: 978-2-212-13532-9. URL: http://www.eyrolles.com/Informatique/Livre/recherche-d-information-9782212135329.

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma (2017). "A Simple but Tough-to-Beat Baseline for Sentence Embeddings". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SyK00v5xx.

Atreya, Avinash and Charles Elkan (Mar. 2011). "Latent Semantic Indexing (LSI) Fails for TREC Collections". In: *SIGKDD Explor.*

Newsl. 12.2, pp. 5–10. ISSN: 1931-0145. DOI: 10.1145/1964897.1964900. URL: https://doi.org/10.1145/1964897.1964900.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450.

Bai, Yang, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu (2020). *SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval*. DOI: 10.48550/ARXIV.2010.00768. URL: https://arxiv.org/abs/2010.00768.

Bajaj, Payal, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. (2016). "Ms marco: A human generated machine reading comprehension dataset". In: *arXiv preprint arXiv:1611.09268*.

Bassani, Elias (2022). "ranx: A Blazing-Fast Python Library for Ranking Evaluation and Comparison". In: *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II*. Ed. by Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty. Vol. 13186. Lecture Notes in Computer Science. Springer, pp. 259–264. DOI: 10.1007/978-3-030-99739-7_30. URL: https://doi.org/10.1007/978-3-030-99739-7%5C_30.

Belinkov, Yonatan (Mar. 2022). "Probing Classifiers: Promises, Shortcomings, and Advances". In: *Computational Linguistics* 48.1, pp. 207–219. DOI: 10.1162/coli_a_00422. URL: https://aclanthology.org/2022.cl-1.7.

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). "Longformer: The Long-Document Transformer". In: *arXiv:2004.05150*.

Berger, Adam and John Lafferty (1999). "Information Retrieval as Statistical Translation". In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '99. Berkeley, California, USA: Association for Computing Machinery, pp. 222–229. ISBN: 1581130961. DOI: 10.1145/312624.312681. URL: https://doi.org/10.1145/312624.312681.

Bevilacqua, Michele, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni (2022). *Autoregressive Search Engines: Generating Substrings as Document Identifiers*. DOI: 10.48550/ARXIV.2204.10628. URL: https://arxiv.org/abs/2204.10628.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3, pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://portal.acm.org/citation.cfm?id=944937.

Bonifacio, Luiz, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira (2022). "InPars: Unsupervised Dataset Generation for Information Retrieval". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2387–2392. ISBN: 9781450387323. DOI: 10.1145/3477495.3531863. URL: https://doi.org/10.1145/3477495.3531863.

Boytsov, Leonid and Zico Kolter (2021). "Exploring Classic and Neural Lexical Translation Models for Information Retrieval: Interpretability, Effectiveness, and Efficiency Benefits". In: *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, pp. 63–78. ISBN: 978-3-030-72112-1. DOI: 10.1007/978-3-030-72113-8_5. URL: https://doi.org/10.1007/978-3-030-72113-8_5.

Boytsov, Leonid, Tianyi Lin, Fangwei Gao, Yutian Zhao, Jeffrey Huang, and Eric Nyberg (2022). *Understanding Performance of Long-Document Ranking Models through Comprehensive Evaluation and Leaderboarding*. DOI: 10.48550/ARXIV.2207.01262. URL: https://arxiv.org/abs/2207.01262.

Boytsov, Leonid, David Novak, Yury Malkov, and Eric Nyberg (2016). "Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, pp. 1099–1108. ISBN: 9781450340731. DOI: 10.1145/2983323.2983815. URL: https://doi.org/10.1145/2983323.2983815.

Boytsov, Leonid, Preksha Patel, Vivek Sourabh, Riddhi Nisar, Sayani Kundu, Ramya Ramanathan, and Eric Nyberg (2023). *InPars-Light: Cost-Effective Unsupervised Training of Efficient Rankers*. DOI: 10.48550/ARXIV.2301.02998. URL: https://arxiv.org/abs/2301.02998.

Broder, Andrei Z., David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien (2003). "Efficient Query Evaluation Using a Two-Level Retrieval Process". In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. CIKM '03. New Orleans, LA, USA: Association for Computing Machinery, pp. 426–434. ISBN: 1581137230. DOI: 10.1145/956863.956944. URL: https://doi.org/10.1145/956863.956944.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1993). "Signature Verification Using a "Siamese" Time Delay Neural Network". In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. NIPS'93. Denver, Colorado: Morgan Kaufmann Publishers Inc., pp. 737–744.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam,

Girish Sastry, et al. (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Bruch, Sebastian, Siyu Gai, and Amir Ingber (2022). *An Analysis of Fusion Functions for Hybrid Retrieval*. DOI: 10.48550/ARXIV.2210.11934. URL: https://arxiv.org/abs/2210.11934.

Bruch, Sebastian, Franco Maria Nardini, Amir Ingber, and Edo Liberty (2023). *An Approximate Algorithm for Maximum Inner Product Search over Streaming Sparse Vectors*. DOI: 10.48550/ARXIV.2301.10622. URL: https://arxiv.org/abs/2301.10622.

Brunner, Gino, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer (2020). "On Identifiability in Transformers". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJg1f6EFDB.

Buckley, Chris (2004). "Why Current IR Engines Fail". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. Sheffield, United Kingdom: Association for Computing Machinery, pp. 584–585. ISBN: 1581138814. DOI: 10.1145/1008992.1009132. URL: https://doi.org/10.1145/1008992.1009132.

Burges, Chris, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender (2005). "Learning to Rank Using Gradient Descent". In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. Bonn, Germany: Association for Computing Machinery, pp. 89–96. ISBN: 1595931805. DOI: 10.1145/1102351.1102363. URL: https://doi.org/10.1145/1102351.1102363.

Burges, Christopher J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An Overview*. Tech. rep. Microsoft Research. URL: http://research.microsoft.com/en-us/um/people/cburges/tech%5C_reports/MSR-TR-2010-82.pdf.

Câmara, Arthur and Claudia Hauff (2020). "Diagnosing BERT with Retrieval Heuristics". In: *Advances in Information Retrieval*. Ed. by Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins. Cham: Springer International Publishing, pp. 605–618. ISBN: 978-3-030-45439-5.

Carmel, David and Elad Yom-Tov (2010). "Estimating the Query Difficulty for Information Retrieval". In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '10. Geneva, Switzerland: Association for Computing Machinery, p. 911. ISBN: 9781450301534. DOI: 10.1145/1835449.1835683. URL: https://doi.org/10.1145/1835449.1835683.

Chang, Wei-Cheng, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar (2020). "Pre-training Tasks for Embedding-based Large-scale Retrieval". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=rkg-mA4FDr.

Chen, Chen, Bowen Zhang, Liangliang Cao, Jiguang Shen, Tom Gunter, Albin Madappally Jose, Alexander Toshev, Jonathon Shlens, Ruoming Pang, and Yinfei Yang (2023). *STAIR: Learning Sparse Text and Image Representation in Grounded Tokens*. arXiv: 2301.13081 [cs.CV].

Chen, Jia, Yiqun Liu, Yan Fang, Jiaxin Mao, Hui Fang, Shenghao Yang, Xiaohui Xie, Min Zhang, and Shaoping Ma (2022). "Axiomatically Regularized Pre-Training for Ad Hoc Search". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 1524–1534. ISBN: 9781450387323. DOI: 10.1145/3477495.3531943. URL: https://doi.org/10.1145/3477495.3531943.

Chen, Jiangui, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng (2022). "CorpusBrain: Pre-Train a Generative Retrieval Model for Knowledge-Intensive Language Tasks". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, pp. 191–200. ISBN: 9781450392365. DOI: 10.1145/3511808.3557271. URL: https://doi.org/10.1145/3511808.3557271.

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021). "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: *arXiv preprint arXiv:2106.01345*.

Chen, Tao, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork (2022). "Out-of-Domain Semantics to the Rescue! Zero-Shot Hybrid Retrieval Models". In: *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part I*. Ed. by Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty. Vol. 13185. Lecture Notes in Computer Science. Springer, pp. 95–110. DOI: 10.1007/978-3-030-99736-6_7. URL: https://doi.org/10.1007/978-3-030-99736-6%5C_7.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2022). "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org.

Choi, Eunseong, Sunkyung Lee, Minijn 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 & Knowledge Management*. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, pp. 272–282. ISBN: 9781450392365. DOI: 10.1145/3511808.3557456. URL: https://doi.org/10.1145/3511808.3557456.

Choi, Jaekeol, Euna Jung, Sungjun Lim, and Wonjong Rhee (2022). *Finding Inverse Document Frequency Information in BERT*. DOI: 10.48550/ARXIV.2202.12191. URL: https://arxiv.org/abs/2202.12191.

Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, et al. (2022). *PaLM: Scaling Language Modeling with Pathways*. DOI: 10.48550/ARXIV.2204.02311. URL: https://arxiv.org/abs/2204.02311.

Clark, Kevin, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning (Aug. 2019). "What Does BERT Look at? An Analysis of BERT's Attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, pp. 276–286. DOI: 10.18653/v1/W19-4828. URL: https://aclanthology.org/W19-4828.

Clark, Kevin, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning (2020). "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1xMH1BtvB.

Claveau, Vincent (2021). "Neural Text Generation for Query Expansion in Information Retrieval". In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. WI-IAT '21. Melbourne, VIC, Australia: Association for Computing Machinery, pp. 202–209. ISBN: 9781450391153. DOI: 10.1145/3486622.3493957. URL: https://doi.org/10.1145/3486622.3493957.

Clinchant, Stéphane and Florent Perronnin (Aug. 2013). "Aggregating Continuous Word Embeddings for Information Retrieval". In: *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 100–109. URL: https://aclanthology.org/W13-3212.

Craswell, Nick, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos (2021a). "Overview of the TREC 2020 deep learning track". In: *CoRR* abs/2102.07662. arXiv: 2102.07662. URL: https://arxiv.org/abs/2102.07662.

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

Craswell, Nick, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees (2020). "Overview of the TREC 2019 deep learning track". In: *CoRR* abs/2003.07820. arXiv: 2003.07820. URL: https://arxiv.org/abs/2003.07820.

Craswell, Nick, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Ian Soboroff (2021b). "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. Virtual Event, Canada: Association for Computing Machinery, pp. 2369–2375. ISBN: 9781450380379. DOI: 10.1145/3404835.3463249. URL: https://doi.org/10.1145/3404835.3463249.

Dai, Zhuyun and Jamie Callan (2019a). *Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval*. DOI: 10.48550/ARXIV.1910.10687. URL: https://arxiv.org/abs/1910.10687.

— (2019b). "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 2019, Paris, France, July 21-25, 2019*. Ed. by Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer. ACM, pp. 985–988. DOI: 10.1145/3331184.3331303. URL: https://doi.org/10.1145/3331184.3331303.

— (2020a). "Context-Aware Document Term Weighting for Ad-Hoc Search". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, pp. 1897–1907. ISBN: 9781450370233. DOI: 10.1145/3366423.3380258. URL: https://doi.org/10.1145/3366423.3380258.

— (2020b). "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. Virtual Event, China: Association for Computing Machinery, pp. 1533–1536. ISBN: 9781450380164. DOI: 10.1145/3397271.3401204. URL: https://doi.org/10.1145/3397271.3401204.

Dai, Zhuyun, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu (2018). "Convolutional Neural Networks for Soft-Matching N-Grams in Ad-Hoc Search". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: Association for Computing Machinery, pp. 126–

134. ISBN: 9781450355810. DOI: 10.1145/3159652.3159659. URL: https://doi.org/10.1145/3159652.3159659.

Dai, Zhuyun, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang (2022). *Promptagator: Few-shot Dense Retrieval From 8 Examples*. DOI: 10.48550/ARXIV.2209.11755. URL: https://arxiv.org/abs/2209.11755.

Datar, Mayur, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni (2004). "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Brooklyn, New York, USA: Association for Computing Machinery, pp. 253–262. ISBN: 1581138857. DOI: 10.1145/997817.997857. URL: https://doi.org/10.1145/997817.997857.

Deerwester, Scott (1988). "Improving information retrieval with latent semantic indexing". In.

Dehghani, Mostafa, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft (2017). "Neural Ranking Models with Weak Supervision". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, pp. 65–74. ISBN: 9781450350228. DOI: 10.1145/3077136.3080832. URL: https://doi.org/10.1145/3077136.3080832.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B* 39, pp. 1–38. URL: http://web.mit.edu/6.435/www/Dempster77.pdf.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Ding, Shuai and Torsten Suel (2011). "Faster Top-k Document Retrieval Using Block-Max Indexes". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, pp. 993–1002. ISBN: 9781450307574. DOI: 10.1145/2009916.2010048. URL: https://doi.org/10.1145/2009916.2010048.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, et al. (2021). "An Image is Worth

16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR*.

Ethayarajh, Kawin (Nov. 2019). "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings". 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)*. Hong Kong, China: Association for Computational Linguistics, pp. 55–65. DOI: 10.18653/v1/D19-1006. URL: https://aclanthology.org/D19-1006.

Faggioli, Guglielmo, Thibault Formal, Stefano Marchesin, Stéphane Clinchant, Nicola Ferro, and Benjamin Piwowarski (2023). "Query Performance Prediction for Neural IR: Are We There Yet?" In: *Advances in Information Retrieval*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Cham: Springer Nature Switzerland, pp. 232–248. ISBN: 978-3-031-28244-7.

Fan, Angela, Mike Lewis, and Yann Dauphin (July 2018). "Hierarchical Neural Story Generation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 889–898. DOI: 10.18653/v1/P18-1082. URL: https://aclanthology.org/P18-1082.

Fang, Hui, Tao Tao, and ChengXiang Zhai (2004). "A Formal Study of Information Retrieval Heuristics". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '04. Sheffield, United Kingdom: Association for Computing Machinery, pp. 49–56. ISBN: 1581138814. DOI: 10.1145/1008992.1009004. URL: https://doi.org/10.1145/1008992.1009004.

Ferrante, Marco, Nicola Ferro, and Norbert Fuhr (2021). "Towards Meaningful Statements in IR Evaluation: Mapping Evaluation Measures to Interval Scales". In: *IEEE Access* 9, pp. 136182–136216. DOI: 10.1109/ACCESS.2021.3116857.

Fisher, R. A. (1922). "On the mathematical foundations of theoretical statistics". In: *Philosophical Transactions of the Royal Society of London, A* 222. Ed. by R. A. Fisher, pp. 309–368.

Formal, Thibault, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant (2021a). *"SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval"*. DOI: 10.48550/ARXIV.2109.10086. URL: https://arxiv.org/abs/2109.10086.

— (2022a). "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*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2353–2359. ISBN: 9781450387323.

DOI: 10.1145/3477495.3531857. URL: https://doi.org/10.1145/3477495.3531857.

— (*Under review*). "Towards Effective and Efficient Sparse Neural Information Retrieval". In: *ACM Trans. Web*.

Formal, Thibault, Benjamin Piwowarski, and Stéphane Clinchant (2020). "Naver Labs Europe @ TREC Deep Learning 2020". In: *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*. Ed. by Ellen M. Voorhees and Angela Ellis. Vol. 1266. NIST Special Publication. National Institute of Standards and Technology (NIST). URL: https://trec.nist.gov/pubs/trec29/papers/NLE.DL.pdf.

— (2021b). "A White Box Analysis of ColBERT". In: *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. Vol. 12657. Lecture Notes in Computer Science. Springer, pp. 257–263. DOI: 10.1007/978-3-030-72240-1_23. URL: https://doi.org/10.1007/978-3-030-72240-1%5C_23.

— (2021c). "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. Virtual Event, Canada: Association for Computing Machinery, pp. 2288–2292. ISBN: 9781450380379. DOI: 10.1145/3404835.3463098. URL: https://doi.org/10.1145/3404835.3463098.

— (Apr. 2021d). "Une Analyse du Modèle ColBERT". In: *Conférence en Recherche d'Information et Applications*. Grenoble (virtuelle), France. URL: https://hal.sorbonne-universite.fr/hal-03364403.

— (2022b). "Match Your Words! A Study of Lexical Matching in Neural Information Retrieval". In: *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*. Stavanger, Norway: Springer-Verlag, pp. 120–127. ISBN: 978-3-030-99738-0. DOI: 10.1007/978-3-030-99739-7_14. URL: https://doi.org/10.1007/978-3-030-99739-7_14.

— (2022c). "Match Your Words! A Study of Lexical Matching in Neural Information Retrieval (extended abstract)". In: *Joint Conference of the Information Retrieval Communities in Europe*. Samatan, France.

Frej, Jibril, Philippe Mulhem, Didier Schwab, and Jean-Pierre Chevallet (2020). "Learning Term Discrimination". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: As-

sociation for Computing Machinery, pp. 1993–1996. ISBN: 9781450380164. DOI: 10.1145/3397271.3401211. URL: https://doi.org/10.1145/3397271.3401211.

Fröbe, Maik, Christopher Akiki, Martin Potthast, and Matthias Hagen (2022). "How Train–Test Leakage Affects Zero-Shot Retrieval". In: *String Processing and Information Retrieval: 29th International Symposium, SPIRE 2022, Concepción, Chile, November 8–10, 2022, Proceedings*. Concepción, Chile: Springer-Verlag, pp. 147–161. ISBN: 978-3-031-20642-9. DOI: 10.1007/978-3-031-20643-6_11. URL: https://doi.org/10.1007/978-3-031-20643-6_11.

Furnas, G. W., T. K. Landauer, L. M. Gomez, and S. T. Dumais (Nov. 1987). "The Vocabulary Problem in Human-System Communication". In: *Commun. ACM* 30.11, pp. 964–971. ISSN: 0001-0782. DOI: 10.1145/32206.32212. URL: https://doi.org/10.1145/32206.32212.

Gao, Luyu and Jamie Callan (Nov. 2021). "Condenser: a Pre-training Architecture for Dense Retrieval". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 981–993. DOI: 10.18653/v1/2021.emnlp-main.75. URL: https://aclanthology.org/2021.emnlp-main.75.

— (May 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)*. Dublin, Ireland: Association for Computational Linguistics, pp. 2843–2853. DOI: 10.18653/v1/2022.acl-long.203. URL: https://aclanthology.org/2022.acl-long.203.

Gao, Luyu, Zhuyun Dai, and Jamie Callan (Nov. 2020). "Modularized Transfomer-based Ranking Framework". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 4180–4190. DOI: 10.18653/v1/2020.emnlp-main.342. URL: https://aclanthology.org/2020.emnlp-main.342.

— (June 2021a). "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*. Online: Association for Computational Linguistics, pp. 3030–3042. DOI: 10.18653/v1/2021.naacl-main.241. URL: https://aclanthology.org/2021.naacl-main.241.

— (2021b). "Rethink Training of BERT Rerankers in Multi-Stage Retrieval Pipeline". In: *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, pp. 280–286. ISBN: 978-3-030-72239-5. DOI: 10.1007/978-

3-030-72240-1_26. URL: https://doi.org/10.1007/978-3-030-72240-1_26.

Ge, Tiezheng, Kaiming He, Qifa Ke, and Jian Sun (2014). "Optimized Product Quantization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.4, pp. 744–755. DOI: 10.1109/TPAMI.2013.240.

Gerald, Thomas and Laure Soulier (2022). "Continual Learning of Long Topic Sequences in Neural Information Retrieval". In: *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*. Stavanger, Norway: Springer-Verlag, pp. 244–259. ISBN: 978-3-030-99735-9. DOI: 10.1007/978-3-030-99736-6_17. URL: https://doi.org/10.1007/978-3-030-99736-6_17.

Gillick, Daniel, Alessandro Presta, and Gaurav Singh Tomar (2018). *End-to-End Retrieval in Continuous Space*. DOI: 10.48550/ARXIV.1811.08008. URL: https://arxiv.org/abs/1811.08008.

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press.

Gospodinov, Mitko, Sean MacAvaney, and Craig Macdonald (2023). *Doc2Query–: When Less is More*. DOI: 10.48550/ARXIV.2301.03266. URL: https://arxiv.org/abs/2301.03266.

Guo, Chuan, Ali Mousavi, Xiang Wu, Daniel Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar (2019). "Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Guo, Jiafeng, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng (Mar. 2022). "Semantic Models for the First-Stage Retrieval: A Comprehensive Review". In: *ACM Trans. Inf. Syst.* 40.4. ISSN: 1046-8188. DOI: 10.1145/3486250. URL: https://doi.org/10.1145/3486250.

Guo, Jiafeng, Yixing Fan, Qingyao Ai, and W. Bruce Croft (2016). "A Deep Relevance Matching Model for Ad-Hoc Retrieval". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, pp. 55–64. ISBN: 9781450340731. DOI: 10.1145/2983323.2983769. URL: https://doi.org/10.1145/2983323.2983769.

Guo, Jiafeng, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng (2020). "A Deep Look into neural ranking models for information retrieval". In: *Inf. Process. Manag.* 57.6, p. 102067. DOI: 10.1016/j.ipm.2019.102067. URL: https://doi.org/10.1016/j.ipm.2019.102067.

Guo, Yu, Zhengyi Ma, Jiaxin Mao, Hongjin Qian, Xinyu Zhang, Hao Jiang, Zhao Cao, and Zhicheng Dou (2022). "Webformer: Pre-Training with Web Pages for Information Retrieval". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 1502–1512. ISBN: 9781450387323. DOI: 10.1145/3477495.3532086. URL: https://doi.org/10.1145/3477495.3532086.

Gutmann, M. and A. Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*. Ed. by Y.W. Teh and M. Titterington. Vol. 9. JMLR W&CP, pp. 297–304.

Guu, Kelvin, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang (2020). "REALM: Retrieval-Augmented Language Model Pre-Training". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org.

Hai Le, Nam, Thomas Gerald, Thibault Formal, Jian-Yun Nie, Benjamin Piwowarski, and Laure Soulier (2023). "CoSPLADE: Contextualizing SPLADE For Conversational Information Retrieval". In: *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part I*. Dublin, Ireland: Springer-Verlag, pp. 537–552. ISBN: 978-3-031-28243-0. DOI: 10.1007/978-3-031-28244-7_34. URL: https://doi.org/10.1007/978-3-031-28244-7_34.

Harris, Zellig (1954). "Distributional structure". In: *Word* 10.2-3, pp. 146–162. DOI: 10.1007/978-94-009-8467-7_1. URL: https://link.springer.com/chapter/10.1007/978-94-009-8467-7_1.

Hasibi, Faegheh, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan (2017). "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*. SIGIR '17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, pp. 1265–1268. ISBN: 9781450350228. DOI: 10.1145/3077136.3080751. URL: https://doi.org/10.1145/3077136.3080751.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

He, Pengcheng, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen (2020). *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. DOI: 10.48550/ARXIV.2006.03654. URL: https://arxiv.org/abs/2006.03654.

Heaps, H. S. (1978). *Information Retrieval: Computational and Theoretical Aspects*. USA: Academic Press, Inc. ISBN: 0123357500.

Hendrycks, Dan and Kevin Gimpel (2016). *Gaussian Error Linear Units (GELUs)*. DOI: 10.48550/ARXIV.1606.08415. URL: https://arxiv.org/abs/1606.08415.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). *Distilling the Knowledge in a Neural Network*. DOI: 10.48550/ARXIV.1503.02531. URL: https://arxiv.org/abs/1503.02531.

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis". In: *Proc. of Uncertainty in Artificial Intelligence, UAI'99*. Stockholm. URL: http://citeseer.ist.psu.edu/hofmann99probabilistic.html.

Hofstätter, Sebastian, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury (2020a). *Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation*. DOI: 10.48550/ARXIV.2010.02666. URL: https://arxiv.org/abs/2010.02666.

Hofstätter, Sebastian, 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: DOI: 10.48550/ARXIV.2203.13088. URL: https://arxiv.org/abs/2203.13088.

Hofstätter, Sebastian, 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*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 113–122. ISBN: 9781450380379. DOI: 10.1145/3404835.3462891. URL: https://doi.org/10.1145/3404835.3462891.

Hofstätter, Sebastian, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury (2020b). "Local Self-Attention over Long Text for Efficient Document Retrieval". In: *Proc. of SIGIR*.

Hofstätter, Sebastian, Markus Zlabinger, and Allan Hanbury (2020c). "Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking". In: *Proc. of ECAI*.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly (2019). *Parameter-Efficient Transfer Learning for NLP*. arXiv: 1902.00751 [cs.LG].

Howard, Jeremy and Sebastian Ruder (2018). "Universal Language Model Fine-tuning for Text Classification". In: *ACL*. Association for Computational Linguistics. URL: http://arxiv.org/abs/1801.06146.

Huang, Po-Sen, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck (2013). "Learning deep structured semantic models for web search using clickthrough data." In: *CIKM*. Ed. by Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi. ACM,

pp. 2333–2338. ISBN: 978-1-4503-2263-8. URL: http://dblp.uni-trier.de/db/conf/cikm/cikm2013.html#HuangHGDAH13.

Hui, Kai, Andrew Yates, Klaus Berberich, and Gerard de Melo (Sept. 2017). "PACRR: A Position-Aware Neural IR Model for Relevance Matching". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1049–1058. DOI: 10.18653/v1/D17-1110. URL: https://aclanthology.org/D17-1110.

— (2018). "Co-PACRR: A Context-Aware Neural IR Model for Ad-Hoc Retrieval". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: Association for Computing Machinery, pp. 279–287. ISBN: 9781450355810. DOI: 10.1145/3159652.3159689. URL: https://doi.org/10.1145/3159652.3159689.

Humeau, Samuel, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston (2020). "Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SkxgnnNFvH.

Iida, Hiroki and Naoaki Okazaki (Nov. 2022). "Unsupervised Domain Adaptation for Sparse Retrieval by Filling Vocabulary and Word Frequency Gaps". In: *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online only: Association for Computational Linguistics, pp. 752–765. URL: https://aclanthology.org/2022.aacl-main.57.

Indyk, Piotr and Rajeev Motwani (1998). "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In: *STOC*. Ed. by Jeffrey Scott Vitter. ACM, pp. 604–613. ISBN: 0-89791-962-9. URL: http://dblp.uni-trier.de/db/conf/stoc/stoc1998.html#IndykM98.

Izacard, Gautier, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave (2022). "Unsupervised Dense Information Retrieval with Contrastive Learning". In: *Transactions on Machine Learning Research*. URL: https://openreview.net/forum?id=jKN1pXi7b0.

Izacard, Gautier, Fabio Petroni, Lucas Hosseini, Nicola de Cao, Sebastian Riedel, and Edouard Grave (Dec. 2021). "A Memory Efficient Baseline for Open Domain Question Answering". working paper or preprint. URL: https://hal.archives-ouvertes.fr/hal-03463488.

Jaleel, Nasreen Abdul, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade (2004). "UMass at TREC 2004: Novelty and HARD". In: *TREC*.

Järvelin, Kalervo and Jaana Kekäläinen (Oct. 2002). "Cumulated Gain-Based Evaluation of IR Techniques". In: *ACM Trans. Inf. Syst.* 20.4, pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418. URL: https://doi.org/10.1145/582415.582418.

Jégou, Hervé, Matthijs Douze, and Cordelia Schmid (2011). "Product Quantization for Nearest Neighbor Search." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.1, pp. 117–128. URL: http://dblp.uni-trier.de/db/journals/pami/pami33.html#JegouDS11.

Jeronymo, Vitor, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira (2023). *InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval.* DOI: 10.48550/ARXIV.2301.01820. URL: https://arxiv.org/abs/2301.01820.

Jiang, Zhiying, Raphael Tang, Ji Xin, and Jimmy Lin (2021). "How Does BERT Rerank Passages? An Attribution Analysis with Information Bottlenecks". en. In: *EMNLP Workshop, Black Box NLP*, p. 14.

Joachims, Thorsten (2002). "Optimizing Search Engines Using Clickthrough Data". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, pp. 133–142. ISBN: 158113567X. DOI: 10.1145/775047.775067. URL: https://doi.org/10.1145/775047.775067.

Joachims, Thorsten, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay (Apr. 2007). "Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search". In: *ACM Trans. Inf. Syst.* 25.2, 7–es. ISSN: 1046-8188. DOI: 10.1145/1229179.1229181. URL: https://doi.org/10.1145/1229179.1229181.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2019). "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3, pp. 535–547.

Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih (Nov. 2020). "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Online: Association for Computational Linguistics, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://www.aclweb.org/anthology/2020.emnlp-main.550.

Kendall, M. G. (1938). "A New Measure of Rank Correlation". In: *Biometrika* 30.1/2, pp. 81–93. ISSN: 00063444. URL: http://www.jstor.org/stable/2332226.

Khattab, Omar and Matei Zaharia (2020). "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT". In: *Proceedings of the 43rd International ACM SIGIR Con-*

ference on Research and Development in Information Retrieval. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 39–48. ISBN: 9781450380164. DOI: 10.1145/3397271.3401075. URL: https://doi.org/10.1145/3397271.3401075.

Kim, Gyuwan, Jinhyuk Lee, Barlas Oguz, Wenhan Xiong, Yizhe Zhang, Yashar Mehdad, and William Yang Wang (2022). "Bridging the Training-Inference Gap for Dense Phrase Retrieval". In: *ICML 2022 Workshop on Knowledge Retrieval and Language Models*. URL: https://openreview.net/forum?id=vtOrgYjli9R.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, et al. (Mar. 2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13, pp. 3521–3526. DOI: 10.1073/pnas.1611835114. URL: https://doi.org/10.1073%5C%2Fpnas.1611835114.

Kitaev, Nikita, Lukasz Kaiser, and Anselm Levskaya (2020). "Reformer: The Efficient Transformer". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkgNKkHtvB.

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert (2015). "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6.

Lassance, Carlos 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. Madrid, Spain: Association for Computing Machinery, pp. 2220–2226. ISBN: 9781450387323. DOI: 10.1145/3477495.3531833. URL: https://doi.org/10.1145/3477495.3531833.

Lassance, Carlos, Hervé Dejean, and Stéphane Clinchant (2023). "An Experimental Study on Pretraining Transformers from Scratch for IR". In: *Advances in Information Retrieval*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Cham: Springer Nature Switzerland, pp. 504–520. ISBN: 978-3-031-28244-7.

Lassance, Carlos, Thibault Formal, and Stéphane Clinchant (2021a). "Composite Code Sparse Autoencoders for First Stage Retrieval". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 2136–

2140. ISBN: 9781450380379. DOI: 10.1145/3404835.3463066. URL: https://doi.org/10.1145/3404835.3463066.

Lassance, Carlos, Thibault Formal, Benjamin Piwowarski, Arnaud Sors, and Stéphane Clinchant (2021b). "NAVER LABS EUROPE (SPLADE) @ TREC DEEP LEARNING 2021". In: *Proceedings of the Thirtieth Text REtrieval Conference, TREC 2021, Virtual Event [Gaithersburg, Maryland, USA], November 15-19, 2021*. Ed. by Ellen M. Voorhees and Angela Ellis. NIST Special Publication. National Institute of Standards and Technology (NIST). URL: https://trec.nist.gov/pubs/trec30/papers/NLE-DL.pdf.

Lassance, Carlos, 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. Madrid, Spain: Association for Computing Machinery, pp. 2232–2236. ISBN: 9781450387323. DOI: 10.1145/3477495.3531835. URL: https://doi.org/10.1145/3477495.3531835.

Lavrenko, Victor 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. New Orleans, Louisiana, USA: Association for Computing Machinery, pp. 120–127. ISBN: 1581133316. DOI: 10.1145/383952.383972. URL: https://doi.org/10.1145/383952.383972.

Le, Quoc and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, II–1188–II–1196.

Lee, Hyunji, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vlad Karpukhin, Yi Lu, and Minjoon Seo (2022a). *Contextualized Generative Retrieval*. DOI: 10.48550/ARXIV.2210.02068. URL: https://arxiv.org/abs/2210.02068.

Lee, Hyunji, Sohee Yang, Hanseok Oh, and Minjoon Seo (2022b). *Generative Multi-hop Retrieval*. DOI: 10.48550/ARXIV.2204.13596. URL: https://arxiv.org/abs/2204.13596.

Lee, Jinhyuk, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y. Zhao (2023). *Rethinking the Role of Token Retrieval in Multi-Vector Retrieval*. arXiv: 2304.01982 [cs.CL].

Lee, Jinhyuk, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang (July 2020). "Contextualized Sparse Representations for Real-Time Open-Domain Question Answering". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 912–919. DOI:

10.18653/v1/2020.acl-main.85. URL: https://aclanthology.org/2020.acl-main.85.

Lee, Kenton, Ming-Wei Chang, and Kristina Toutanova (July 2019). "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 6086–6096. DOI: 10.18653/v1/P19-1612. URL: https://aclanthology.org/P19-1612.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (July 2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

Li, Canjia, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun (2020). *PARADE: Passage Representation Aggregation for Document Reranking*. DOI: 10.48550/ARXIV.2008.09093. URL: https://arxiv.org/abs/2008.09093.

Li, Hang, Ahmed Mourad, Bevan Koopman, and Guido Zuccon (2022a). "How Does Feedback Signal Quality Impact Effectiveness of Pseudo Relevance Feedback for Passage Retrieval". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2154–2158. ISBN: 9781450387323. DOI: 10.1145/3477495.3531822. URL: https://doi.org/10.1145/3477495.3531822.

Li, Hang, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon (Jan. 2023). "Pseudo Relevance Feedback with Deep Language Models and Dense Retrievers: Successes and Pitfalls". In: *ACM Trans. Inf. Syst.* Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3570724. URL: https://doi.org/10.1145/3570724.

Li, Hang, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon (2022b). "To Interpolate or Not to Interpolate: PRF, Dense and Sparse Retrievers". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2495–2500. ISBN: 9781450387323. DOI: 10.1145/3477495.3531884. URL: https://doi.org/10.1145/3477495.3531884.

Lin, Jimmy (Jan. 2019). "The Neural Hype and Comparisons Against Weak Baselines". In: *SIGIR Forum* 52.2, pp. 40–51. ISSN: 0163-5840. DOI: 10.1145/3308774.3308781. URL: https://doi.org/10.1145/3308774.3308781.

— (Mar. 2022). "A Proposed Conceptual Framework for a Representational Approach to Information Retrieval". In: *SIGIR Forum* 55.2. ISSN: 0163-5840. DOI: 10.1145/3527546.3527552. URL: https://doi.org/10.1145/3527546.3527552.

Lin, Jimmy and Xueguang Ma (2021). *A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques*. DOI: 10.48550/ARXIV.2106.14807. URL: https://arxiv.org/abs/2106.14807.

Lin, Jimmy, Rodrigo Nogueira, and Andrew Yates (2020). "Pretrained Transformers for Text Ranking: BERT and Beyond". In: *CoRR* abs/2010.06467. arXiv: 2010.06467. URL: https://arxiv.org/abs/2010.06467.

Lin, Sheng-Chieh, Jheng-Hong Yang, and Jimmy Lin (2020). *Distilling Dense Representations for Ranking using Tightly-Coupled Teachers*. DOI: 10.48550/ARXIV.2010.11386. URL: https://arxiv.org/abs/2010.11386.

— (Aug. 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)*. Online: Association for Computational Linguistics, pp. 163–173. DOI: 10.18653/v1/2021.repl4nlp-1.17. URL: https://aclanthology.org/2021.repl4nlp-1.17.

Lindgren, Erik, Sashank Reddi, Ruiqi Guo, and Sanjiv Kumar (2021). "Efficient Training of Retrieval Models using Negative Cache". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 4134–4146. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/2175f8c5cd9604f6b1e576b252d4c86e-Paper.pdf.

Liu, Tie-Yan (2011). *Learning to Rank for Information Retrieval*. Springer, pp. I–XVII, 1–285. ISBN: 978-3-642-14266-6.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Lovón-Melgarejo, Jesús, Laure Soulier, Karen Pinel-Sauvagnat, and Lynda Tamine (Apr. 2021). "Studying Catastrophic Forgetting in Neural Ranking Models". In: *43rd European Conference on Information Retrieval - ECIR 2021*. Lucca (on line), Italy. URL: https://hal.archives-ouvertes.fr/hal-03156630.

Luan, Yi, Jacob Eisenstein, Kristina Toutanova, and Michael Collins (2021). "Sparse, Dense, and Attentional Representations for Text Retrieval". In: *Transactions of the Association for Computational Linguistics* 9, pp. 329–345. DOI: 10.1162/tacl_a_00369. URL: https://aclanthology.org/2021.tacl-1.20.

luo, Ziyang, Pu Zhao, Can Xu, Xiubo Geng, Tao Shen, Chongyang Tao, Jing Ma, Qingwen lin, and Daxin Jiang (2023). *LexLIP: Lexicon-Bottlenecked Language-Image Pre-Training for Large-Scale Image-Text Retrieval*. arXiv: 2302.02908 [cs.CV].

Lupart, Simon, Thibault Formal, and Stéphane Clinchant (2023). "MS-Shift: An Analysis of MS MARCO Distribution Shifts on Neural Retrieval". In: *Advances in Information Retrieval*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Cham: Springer Nature Switzerland, pp. 636–652. ISBN: 978-3-031-28244-7.

Ma, Ji, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald (Apr. 2021). "Zero-shot Neural Passage Retrieval via Domain-targeted Synthetic Question Generation". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 1075–1088. DOI: 10.18653/v1/2021.eacl-main.92. URL: https://aclanthology.org/2021.eacl-main.92.

Ma, Xinyu, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng (2021a). "PROP: Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. WSDM '21. Virtual Event, Israel: Association for Computing Machinery, pp. 283–291. ISBN: 9781450382977. DOI: 10.1145/3437963.3441777. URL: https://doi.org/10.1145/3437963.3441777.

Ma, Xinyu, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng (2021b). "B-PROP: Bootstrapped Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 1513–1522. ISBN: 9781450380379. DOI: 10.1145/3404835.3462869. URL: https://doi.org/10.1145/3404835.3462869.

Ma, Xueguang, Kai Sun, Ronak Pradeep, and Jimmy Lin (2021). "A Replication Study of Dense Passage Retriever". In: *CoRR* abs/2104.05740. arXiv: 2104.05740. URL: https://arxiv.org/abs/2104.05740.

Ma, Zhengyi, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen (2021). "Pre-Training for Ad-Hoc Retrieval: Hyperlink is Also You Need". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. Virtual Event, Queensland, Australia: Association for Computing Machinery, pp. 1212–1221. ISBN: 9781450384469. DOI: 10.1145/3459637.3482286. URL: https://doi.org/10.1145/3459637.3482286.

MacAvaney, Sean, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan (2022). "ABNIRML: Analyzing the Behavior of Neural IR Models". In: *Transactions of the Association for Computa-*

tional Linguistics 10, pp. 224–239. DOI: 10.1162/tacl_a_00457. URL: https://aclanthology.org/2022.tacl-1.13.

MacAvaney, Sean, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder (2020a). "Efficient Document Re-Ranking for Transformers by Precomputing Term Representations". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 49–58. DOI: 10.1145/3397271.3401093. URL: https://arxiv.org/abs/2004.14255.

— (2020b). "Expansion via Prediction of Importance with Contextualization". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 1573–1576. ISBN: 9781450380164. DOI: 10.1145/3397271.3401262. URL: https://doi.org/10.1145/3397271.3401262.

— (2020c). "Expansion via Prediction of Importance with Contextualization". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1573–1576. DOI: 10.1145/3397271.3401262. URL: https://arxiv.org/abs/2004.14245.

MacAvaney, Sean, Andrew Yates, Arman Cohan, and Nazli Goharian (2019a). "CEDR: Contextualized Embeddings for Document Ranking". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1101–1104. DOI: 10.1145/3331184.3331317. URL: https://arxiv.org/abs/1904.07094.

MacAvaney, Sean, Andrew Yates, Kai Hui, and Ophir Frieder (2019b). "Content-Based Weak Supervision for Ad-Hoc Re-Ranking". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 993–996. ISBN: 9781450361729. DOI: 10.1145/3331184.3331316. URL: https://doi.org/10.1145/3331184.3331316.

Macdonald, Craig and Nicola Tonellotto (2021). "On Approximate Nearest Neighbour Selection for Multi-Stage Dense Retrieval". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. Virtual Event, Queensland, Australia: Association for Computing Machinery, pp. 3318–3322. ISBN: 9781450384469. DOI: 10.1145/3459637.3482156. URL: https://doi.org/10.1145/3459637.3482156.

Macdonald, Craig, Nicola Tonellotto, and Iadh Ounis (2021). "On Single and Multiple Representations in Dense Passage Retrieval". In: *Proceedings of the 11th Italian Information Retrieval Workshop 2021, Bari, Italy, September 13-15, 2021*. Ed. by Vito Walter Anelli, Tommaso Di Noia, Nicola Ferro, and Fedelucio Narducci. Vol. 2947.

CEUR Workshop Proceedings. CEUR-WS.org. URL: http://ceur-ws.org/Vol-2947/paper5.pdf.

Mackenzie, Joel, 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. Virtual Event, China: Association for Computing Machinery, pp. 1821–1824. ISBN: 9781450380164. DOI: 10.1145/3397271.3401263. URL: https://doi.org/10.1145/3397271.3401263.

Mackenzie, Joel, Antonio Mallia, Alistair Moffat, and Matthias Petri (2022a). "Accelerating learned sparse indexes via term impact decomposition". In: *EMNLP 2022*. URL: https://www.amazon.science/publications/accelerating-learned-sparse-indexes-via-term-impact-decomposition.

Mackenzie, Joel, Andrew Trotman, and Jimmy Lin (2021). *Wacky Weights in Learned Sparse Representations and the Revenge of Score-at-a-Time Query Evaluation*. DOI: 10.48550/ARXIV.2110.11540. URL: https://arxiv.org/abs/2110.11540.

— (Dec. 2022b). "Efficient Document-at-a-Time and Score-at-a-Time Query Evaluation for Learned Sparse Representations". In: *ACM Trans. Inf. Syst.* Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3576922. URL: https://doi.org/10.1145/3576922.

Malkov, Yu A. and D. A. Yashunin (Apr. 2020). "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 42.4, pp. 824–836. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2018.2889473. URL: https://doi.org/10.1109/TPAMI.2018.2889473.

Mallia, Antonio, 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*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 1723–1727. ISBN: 9781450380379. DOI: 10.1145/3404835.3463030. URL: https://doi.org/10.1145/3404835.3463030.

Mallia, Antonio, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto (2022). "Faster Learned Sparse Retrieval with Guided Traversal". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 1901–1905. ISBN: 9781450387323. DOI: 10.1145/3477495.3531774. URL: https://doi.org/10.1145/3477495.3531774.

Mallia, Antonio, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel (2019). "PISA: Performant Indexes and Search for Academia". In: *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris,*

France, July 25, 2019. Pp. 50–56. URL: http://ceur-ws.org/Vol-2409/docker08.pdf.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press. ISBN: 978-0-521-86571-5. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Matsui, Yusuke, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh (2018). "A Survey of Product Quantization". In: *ITE Transactions on Media Technology and Applications* 6.1, pp. 2–10.

Mehta, Sanket Vaibhav, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler (2022). *DSI++: Updating Transformer Memory with New Documents*. DOI: 10.48550/ARXIV.2212.09744. URL: https://arxiv.org/abs/2212.09744.

Metzler, Donald (2008). "Generalized Inverse Document Frequency". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. Napa Valley, California, USA: Association for Computing Machinery, pp. 399–408. ISBN: 9781595939913. DOI: 10.1145/1458082.1458137. URL: https://doi.org/10.1145/1458082.1458137.

Metzler, Donald, Yi Tay, Dara Bahri, and Marc Najork (July 2021). "Rethinking Search: Making Domain Experts out of Dilettantes". In: *SIGIR Forum* 55.1. ISSN: 0163-5840. DOI: 10.1145/3476415.3476428. URL: https://doi.org/10.1145/3476415.3476428.

Michel, Paul, Omer Levy, and Graham Neubig (2019). "Are Sixteen Heads Really Better than One?" In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, et al. (2018). "Mixed Precision Training". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1gs9JgRZ.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119.

Mitra, Bhaskar and Nick Craswell (Dec. 2018). "An Introduction to Neural Information Retrieval". In: *Found. Trends Inf. Retr.* 13.1, pp. 1–126. ISSN: 1554-0669. DOI: 10.1561/1500000061. URL: https://doi.org/10.1561/1500000061.

Mitra, Bhaskar and Nick Craswell (2019). *An Updated Duet Model for Passage Re-ranking*. arXiv: 1903.07666 [cs.IR].

Mitra, Bhaskar, Fernando Diaz, and Nick Craswell (2017). "Learning to Match Using Local and Distributed Representations of Text for Web Search". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 1291–1299. ISBN: 9781450349130. DOI: 10.1145/3038912.3052579. URL: https://doi.org/10.1145/3038912.3052579.

Mitra, Bhaskar, Eric Nalisnick, Nick Craswell, and Rich Caruana (Feb. 2016). *A Dual Embedding Space Model for Document Ranking*. This paper is an extended evaluation and analysis of the model proposed in a poster to appear in WWW'16, April 11 - 15, 2016, Montreal, Canada. URL: https://www.microsoft.com/en-us/research/publication/a-dual-embedding-space-model-for-document-ranking/.

Mitra, Bhaskar, 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". In: *CoRR* abs/1907.03693. arXiv: 1907.03693. URL: http://arxiv.org/abs/1907.03693.

Mokrii, Iurii, Leonid Boytsov, and Pavel Braslavski (2021a). "A Systematic Evaluation of Transfer Learning and Pseudo-Labeling with BERT-Based Ranking Models". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 2081–2085. ISBN: 9781450380379. DOI: 10.1145/3404835.3463093. URL: https://doi.org/10.1145/3404835.3463093.

— (July 2021b). "A Systematic Evaluation of Transfer Learning and Pseudo-labeling with BERT-based Ranking Models". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. DOI: 10.1145/3404835.3463093. URL: https://doi.org/10.1145%5Ctextit%7B%7D%2F3404835.3463093.

Muennighoff, Niklas (2022). "SGPT: GPT Sentence Embeddings for Semantic Search". In: *arXiv preprint arXiv:2202.08904*.

Nair, Suraj, Eugene Yang, Dawn Lawrie, James Mayfield, and Douglas Oard (2022). "Learning a Sparse Representation Model for Neural CLIR". In: *Proceedings of the 2022 Design of Experimental Search and Information REtrieval Systems*. San Jose.

Nalisnick, Eric, Bhaskar Mitra, Nick Craswell, and Rich Caruana (Apr. 2016). "Improving Document Ranking with Dual Word Embeddings". In: *WWW'16*. WWW - World Wide Web Consortium (W3C). URL: https://www.microsoft.com/en-us/research/publication/improving-document-ranking-with-dual-word-embeddings/.

Naseri, Shahrzad, Jeffrey Dalton, Andrew Yates, and James Allan (2021). "CEQE: Contextualized Embeddings for Query Expansion". In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani. Cham: Springer International Publishing, pp. 467–482. ISBN: 978-3-030-72113-8.

Neelakantan, Arvind, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, et al. (2022). *Text and Code Embeddings by Contrastive Pre-Training*. DOI: 10.48550/ARXIV.2201.10005. URL: https://arxiv.org/abs/2201.10005.

Nguyen, Thong, Sean MacAvaney, and Andrew Yates (2023). "A Unified Framework for Learned Sparse Retrieval". In: *Advances in Information Retrieval*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Cham: Springer Nature Switzerland, pp. 101–116. ISBN: 978-3-031-28241-6.

Nogueira, Rodrigo, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin (Nov. 2020). "Document Ranking with a Pretrained Sequence-to-Sequence Model". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 708–718. DOI: 10.18653/v1/2020.findings-emnlp.63. URL: https://aclanthology.org/2020.findings-emnlp.63.

Nogueira, Rodrigo and Jimmy Lin (2019). *From doc2query to docTTTTTquery*.

Nogueira, Rodrigo, Wei Yang, Kyunghyun Cho, and Jimmy Lin (2019a). "Multi-stage document ranking with BERT". In: *arXiv preprint arXiv:1910.14424*.

Nogueira, Rodrigo, Wei Yang, Jimmy Lin, and Kyunghyun Cho (2019b). *Document Expansion by Query Prediction*. DOI: 10.48550/ARXIV.1904.08375. URL: https://arxiv.org/abs/1904.08375.

Nogueira, Rodrigo Frassetto and Kyunghyun Cho (2019). "Passage Reranking with BERT". In: *CoRR* abs/1901.04085. arXiv: 1901.04085. URL: http://arxiv.org/abs/1901.04085.

Nogueira dos Santos, Cicero, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang (Nov. 2020). "Beyond [CLS] through Ranking by Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 1722–1727. DOI: 10.18653/v1/2020.emnlp-main.134. URL: https://aclanthology.org/2020.emnlp-main.134.

Onal, Kezban Dilek, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, et al. (June 2018). "Neural Information Retrieval: At the End of the Early Years". In: *Inf. Retr.* 21.2–3, pp. 111–

182. ISSN: 1386-4564. DOI: 10.1007/s10791-017-9321-y. URL: https://doi.org/10.1007/s10791-017-9321-y.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (Nov. 1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab. URL: http://ilpubs.stanford.edu:8090/422/.

Pal, Vaishali, Carlos Lassance, Hervé Déjean, and Stéphane Clinchant (2023). "Parameter-Efficient Sparse Retrievers and Rerankers Using Adapters". In: *Advances in Information Retrieval*. Ed. by Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo. Cham: Springer Nature Switzerland, pp. 16–31. ISBN: 978-3-031-28238-6.

Paria, Biswajit, Chih-Kuan Yeh, Ian E.H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos (2020). "Minimizing FLOPs to Learn Efficient Sparse Representations". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SygpC6Ntvr.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Penha, Gustavo, Arthur Câmara, and Claudia Hauff (2022). "Evaluating the Robustness of Retrieval Pipelines with Query Variation Generators". In: *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*. Stavanger, Norway: Springer-Verlag, pp. 397–412. ISBN: 978-3-030-99735-9. DOI: 10.1007/978-3-030-99736-6_27. URL: https://doi.org/10.1007/978-3-030-99736-6_27.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14, pp. 1532–1543.

Perronnin, Florent and Christopher Dance (2007). "Fisher Kernels on Visual Vocabularies for Image Categorization". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2007.383266.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (June 2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

Ponte, Jay M. and W. Bruce Croft (1998). "A Language Modeling Approach to Information Retrieval". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98. Melbourne, Australia: Association for Computing Machinery, pp. 275–281. ISBN: 1581130155. DOI: 10.1145/290941.291008. URL: https://doi.org/10.1145/290941.291008.

Porter, Martin F (1980). "An algorithm for suffix stripping". In: *Program* 14.3, pp. 130–137.

Pradeep, Ronak, Rodrigo Nogueira, and Jimmy Lin (2021). *The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models*. DOI: 10.48550/ARXIV.2101.05667. URL: https://arxiv.org/abs/2101.05667.

Press, Ofir and Lior Wolf (Apr. 2017). "Using the Output Embedding to Improve Language Models". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 157–163. URL: https://aclanthology.org/E17-2025.

Qian, Yujie, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y. Zhao (2022). *Multi-Vector Retrieval as Sparse Alignment*. DOI: 10.48550/ARXIV.2211.01267. URL: https://arxiv.org/abs/2211.01267.

Qiao, Yifan, Yingrui Yang, Haixin Lin, Tianbo Xiong, Xiyue Wang, and Tao Yang (2022). *Dual Skipping Guidance for Document Retrieval with Learned Sparse Representations*. DOI: 10.48550/ARXIV.2204.11154. URL: https://arxiv.org/abs/2204.11154.

Qin, Tao, Tie-Yan Liu, Jun Xu, and Hang Li (Aug. 2016). "LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval". In: URL: https://www.microsoft.com/en-us/research/publication/letor-benchmark-collection-research-learning-rank-information-retrieval/.

Qu, Yingqi, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang (June 2021). "RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 5835–5847. DOI: 10.18653/v1/2021.naacl-main.466. URL: https://aclanthology.org/2021.naacl-main.466.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). *Improving Language Understanding by Generative Pre-Training*. URL: https://openai.com/blog/language-unsupervised/.

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners". In.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

Ram, Ori, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson (2022a). *What Are You Token About? Dense Retrieval as Distributions Over the Vocabulary*. DOI: 10.48550/ARXIV.2212.10380. URL: https://arxiv.org/abs/2212.10380.

Ram, Ori, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson (July 2022b). "Learning to Retrieve Passages without Supervision". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 2687–2700. DOI: 10.18653/v1/2022.naacl-main.193. URL: https://aclanthology.org/2022.naacl-main.193.

Rau, David and Jaap Kamps (2022a). "How Different are Pre-trained Transformers for Text Ranking?" In: *European Conference on Information Retrieval*. Springer, pp. 207–214. DOI: 10.1007/978-3-030-99739-7_24.

— (2022b). "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. Madrid, Spain: Association for Computing Machinery, pp. 153–160. ISBN: 9781450394123. DOI: 10.1145/3539813.3545144. URL: https://doi.org/10.1145/3539813.3545144.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *EMNLP/IJCNLP (1)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Association for Computational Linguistics, pp. 3980–3990. ISBN: 978-1-950737-90-1. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2019-1.html#ReimersG19.

Ren, Ruiyang, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen (Aug. 2021a). "PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, pp. 2173–2183. DOI: 10.18653/

v1/2021.findings-acl.191. URL: https://aclanthology.org/2021.findings-acl.191.

Ren, Ruiyang, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen (Nov. 2021b). "RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2825–2835. DOI: 10.18653/v1/2021.emnlp-main.224. URL: https://aclanthology.org/2021.emnlp-main.224.

Ren, Ruiyang, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qifei Wu, Yuchen Ding, Hua Wu, Haifeng Wang, and Ji-Rong Wen (2022). *A Thorough Examination on Zero-shot Dense Retrieval*. DOI: 10.48550/ARXIV.2204.12755. URL: https://arxiv.org/abs/2204.12755.

Rennings, Daniël, Felipe Moraes, and Claudia Hauff (2019). "An Axiomatic Approach to Diagnosing Neural IR Models". en. In: *Advances in Information Retrieval*. Ed. by Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra. Lecture Notes in Computer Science. ZSCC: NoCitationData[s0]. Cham: Springer International Publishing, pp. 489–503. ISBN: 978-3-030-15712-8. DOI: 10/ggcmnb.

Robertson, S. E. (1997). "The Probability Ranking Principle in IR". In: *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 281–286. ISBN: 1558604545.

Robertson, S. E. and K. Sparck Jones (1976). "Relevance weighting of search terms". en. In: *Journal of the American Society for Information Science* 27.3. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.4630270302, pp. 129–146. ISSN: 1097-4571. DOI: 10/dvgb84. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630270302 (visited on 10/20/2021).

Robertson, Stephen and Hugo Zaragoza (Apr. 2009). "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Found. Trends Inf. Retr.* 3.4, pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: https://doi.org/10.1561/1500000019.

Robertson, Stephen E. (Jan. 1977). "The Probability Ranking Principle in IR". In: *Journal of Documentation* 33.4. Publisher: MCB UP Ltd, pp. 294–304. ISSN: 0022-0418. DOI: 10/ckqfpm. URL: https://doi.org/10.1108/eb026647 (visited on 10/22/2021).

Robertson, Stephen E., Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford (1994). "Okapi at TREC-3." In: *TREC*. Ed. by Donna K. Harman. Vol. 500-225. NIST Special Publication. National Institute of Standards and Technology (NIST), pp. 109–126. URL: http://dblp.uni-trier.de/db/conf/trec/trec94.html#RobertsonWJHG94.

Rocchio, J. J. (1971). "Relevance feedback in information retrieval". In: *The Smart retrieval system - experiments in automatic document*

processing. Ed. by G. Salton. Englewood Cliffs, NJ: Prentice-Hall, pp. 313–323.

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky (2020). "A Primer in BERTology: What We Know About How BERT Works". In: *Transactions of the Association for Computational Linguistics* 8, pp. 842–866. DOI: 10.1162/tacl_a_00349. URL: https://aclanthology.org/2020.tacl-1.54.

Rosa, Guilherme, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira (2022). *In Defense of Cross-Encoders for Zero-Shot Retrieval*. DOI: 10.48550/ARXIV.2212.06121. URL: https://arxiv.org/abs/2212.06121.

Rosa, Guilherme Moraes, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira (2022). *No Parameter Left Behind: How Distillation and Model Size Affect Zero-Shot Retrieval*. DOI: 10.48550/ARXIV.2206.02873. URL: https://arxiv.org/abs/2206.02873.

Saad-Falcon, Jon, Omar Khattab, Keshav Santhanam, Radu Florian, Martin Franz, Salim Roukos, Avirup Sil, Md Arafat Sultan, and Christopher Potts (2023). *UDAPDR: Unsupervised Domain Adaptation via LLM Prompting and Distillation of Rerankers*. arXiv: 2303.00807 [cs.IR].

Sablayrolles, Alexandre, Matthijs Douze, Cordelia Schmid, and Hervé Jégou (2019). "Spreading vectors for similarity search". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SkGuG2R5tm.

Sachan, Devendra Singh, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer (2022a). "Improving Passage Retrieval with Zero-Shot Question Generation". In: *arXiv preprint arXiv:2204.07496*.

Sachan, Devendra Singh, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer (2022b). *Questions Are All You Need to Train a Dense Passage Retriever*. DOI: 10.48550/ARXIV.2206.10658. URL: https://arxiv.org/abs/2206.10658.

Salton, G., A. Wong, and C. S. Yang (Nov. 1975). "A Vector Space Model for Automatic Indexing". In: *Commun. ACM* 18.11, pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: https://doi.org/10.1145/361219.361220.

Salton, G. and C. S. Yang (1973). "On the specification of term values in automatic indexing." In: *Journal of Documentation*. 29.4, pp. 351–372.

Sanderson, Mark (2008). "Ambiguous Queries: Test Collections Need More Sense". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: Association for Computing Machinery, pp. 499–506. ISBN: 9781605581644. DOI: 10.1145/

1390334.1390420. URL: https://doi.org/10.1145/1390334.1390420.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019*. arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

Santhanam, Keshav, Omar Khattab, Christopher Potts, and Matei Zaharia (2022a). *PLAID: An Efficient Engine for Late Interaction Retrieval*. DOI: 10.48550/ARXIV.2205.09707. URL: https://arxiv.org/abs/2205.09707.

Santhanam, Keshav, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia (July 2022b). "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 3715–3734. DOI: 10.18653/v1/2022.naacl-main.272. URL: https://aclanthology.org/2022.naacl-main.272.

Santhanam, Keshav, Jon Saad-Falcon, Martin Franz, Omar Khattab, Avirup Sil, Radu Florian, Md Arafat Sultan, Salim Roukos, Matei Zaharia, and Christopher Potts (2022c). *Moving Beyond Downstream Task Accuracy for Information Retrieval Benchmarking*. DOI: 10.48550/ARXIV.2212.01340. URL: https://arxiv.org/abs/2212.01340.

Sciavolino, Christopher, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen (Nov. 2021a). "Simple Entity-Centric Questions Challenge Dense Retrievers". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 6138–6148. DOI: 10.18653/v1/2021.emnlp-main.496. URL: https://aclanthology.org/2021.emnlp-main.496.

— (2021b). "Simple Entity-centric Questions Challenge Dense Retrievers". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

Shen, Tao, Xiubo Geng, Chongyang Tao, Can Xu, Xiaolong Huang, Binxing Jiao, Linjun Yang, and Daxin Jiang (2022a). *LexMAE: Lexicon-Bottlenecked Pretraining for Large-Scale Retrieval*. DOI: 10.48550/ARXIV.2208.14754. URL: https://arxiv.org/abs/2208.14754.

Shen, Tao, Xiubo Geng, Chongyang Tao, Can Xu, Kai Zhang, and Daxin Jiang (2022b). *UnifieR: A Unified Retriever for Large-Scale Retrieval*. DOI: 10.48550/ARXIV.2205.11194. URL: https://arxiv.org/abs/2205.11194.

Sidiropoulos, Georgios and Evangelos Kanoulas (2022). "Analysing the Robustness of Dual Encoders for Dense Retrieval Against Misspellings". In: *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*. Ed. by Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai. ACM, pp. 2132–2136. DOI: 10.1145/3477495.3531818. URL: https://doi.org/10.1145/3477495.3531818.

Sivic and Zisserman (2003). "Video Google: a text retrieval approach to object matching in videos". In: *Proceedings Ninth IEEE International Conference on Computer Vision*, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

Soboroff, Ian (2018). "Meta-Analysis for Retrieval Experiments Involving Multiple Test Collections". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. Torino, Italy: Association for Computing Machinery, pp. 713–722. ISBN: 9781450360142. DOI: 10.1145/3269206.3271719. URL: https://doi.org/10.1145/3269206.3271719.

Song, Junshuai, Jiangshan Zhang, Jifeng Zhu, Mengyun Tang, and Yong Yang (May 2022). "TRAttack: Text Rewriting Attack Against Text Retrieval". In: *Proceedings of the 7th Workshop on Representation Learning for NLP*. Dublin, Ireland: Association for Computational Linguistics, pp. 191–203. DOI: 10.18653/v1/2022.repl4nlp-1.20. URL: https://aclanthology.org/2022.repl4nlp-1.20.

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

Sun, Yu, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu (2019). *ERNIE: Enhanced Representation through Knowledge Integration*. DOI: 10.48550/ARXIV.1904.09223. URL: https://arxiv.org/abs/1904.09223.

Tänzer, Michael, Sebastian Ruder, and Marek Rei (May 2022). "Memorisation versus Generalisation in Pre-trained Language Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 7564–7578. DOI: 10.18653/v1/2022.acl-long.521. URL: https://aclanthology.org/2022.acl-long.521.

Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler (Apr. 2022a). "Efficient Transformers: A Survey". In: *ACM Comput. Surv.*

Just Accepted. ISSN: 0360-0300. DOI: 10.1145/3530811. URL: https://doi.org/10.1145/3530811.

Tay, Yi, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, et al. (2022b). *Transformer Memory as a Differentiable Search Index*. DOI: 10.48550/ARXIV.2202.06991. URL: https://arxiv.org/abs/2202.06991.

Taylor, Wilson L. (1953). ""Cloze Procedure": A New Tool for Measuring Readability". In: *Journalism Quarterly* 30.4, pp. 415–433. DOI: 10.1177/107769905303000401. eprint: https://doi.org/10.1177/107769905303000401. URL: https://doi.org/10.1177/107769905303000401.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick (July 2019). "BERT Rediscovers the Classical NLP Pipeline". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4593–4601. DOI: 10.18653/v1/P19-1452. URL: https://aclanthology.org/P19-1452.

Thakur, Nandan, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych (2021). "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

Tonellotto, Nicola (2022). *Lecture Notes on Neural Information Retrieval*. DOI: 10.48550/ARXIV.2207.13443. URL: https://arxiv.org/abs/2207.13443.

Tonellotto, Nicola and Craig Macdonald (2021). "Query Embedding Pruning for Dense Retrieval". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. Virtual Event, Queensland, Australia: Association for Computing Machinery, pp. 3453–3457. ISBN: 9781450384469. DOI: 10.1145/3459637.3482162. URL: https://doi.org/10.1145/3459637.3482162.

Tonellotto, Nicola, Craig Macdonald, and Iadh Ounis (2018). "Efficient Query Processing for Scalable Web Search". In: *Foundations and Trends® in Information Retrieval* 12.4-5, pp. 319–500. ISSN: 1554-0669. DOI: 10.1561/1500000057. URL: http://dx.doi.org/10.1561/1500000057.

Turtle, Howard and James Flood (Nov. 1995). "Query Evaluation: Strategies and Optimizations". In: *Inf. Process. Manage.* 31.6, pp. 831–850. ISSN: 0306-4573. DOI: 10.1016/0306-4573(95)00020-H. URL: https://doi.org/10.1016/0306-4573(95)00020-H.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Voita, Elena, Rico Sennrich, and Ivan Titov (Nov. 2019a). "The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives". 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)*. Hong Kong, China: Association for Computational Linguistics, pp. 4396–4406. DOI: 10.18653/v1/D19-1448. URL: https://aclanthology.org/D19-1448.

Voita, Elena, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov (July 2019b). "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5797–5808. DOI: 10.18653/v1/P19-1580. URL: https://aclanthology.org/P19-1580.

Völske, Michael, Alexander Bondarenko, Maik Fröbe, Benno Stein, Jaspreet Singh, Matthias Hagen, and Avishek Anand (2021). "Towards Axiomatic Explanations for Neural Ranking Models". In: *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 13–22. ISBN: 9781450386111. DOI: 10.1145/3471158.3472256. URL: https://doi.org/10.1145/3471158.3472256.

Voorhees, Ellen (Jan. 2004). "Overview of the TREC 2004 Robust Track." In.

Voorhees, Ellen, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang (Feb. 2021). "TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection". In: *SIGIR Forum* 54.1. ISSN: 0163-5840. DOI: 10.1145/3451964.3451965. URL: https://doi.org/10.1145/3451964.3451965.

Voorhees, Ellen M. (1994). "Query Expansion Using Lexical-Semantic Relations". In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '94. Dublin, Ireland: Springer-Verlag, pp. 61–69. ISBN: 038719889X.

Voorhees, Ellen M., Nick Craswell, and Jimmy Lin (2022a). "Too Many Relevants: Whither Cranfield Test Collections?" In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2970–2980. ISBN: 9781450387323.

DOI: 10.1145/3477495.3531728. URL: https://doi.org/10.1145/3477495.3531728.

Voorhees, Ellen M., Ian Soboroff, and Jimmy Lin (2022b). "Can Old TREC Collections Reliably Evaluate Modern Neural Retrieval Models?" In: *CoRR* abs/2201.11086. arXiv: 2201.11086. URL: https://arxiv.org/abs/2201.11086.

Wang, Kexin, Nils Reimers, and Iryna Gurevych (Nov. 2021). "TSDAE: Using Transformer-based Sequential Denoising Auto-Encoderfor Unsupervised Sentence Embedding Learning". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 671–688. DOI: 10.18653/v1/2021.findings-emnlp.59. URL: https://aclanthology.org/2021.findings-emnlp.59.

Wang, Kexin, Nandan Thakur, Nils Reimers, and Iryna Gurevych (July 2022). "GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 2345–2360. DOI: 10.18653/v1/2022.naacl-main.168. URL: https://aclanthology.org/2022.naacl-main.168.

Wang, Lidan, Jimmy Lin, and Donald Metzler (2010). "Learning to Efficiently Rank". In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '10. Geneva, Switzerland: Association for Computing Machinery, pp. 138–145. ISBN: 9781450301534. DOI: 10.1145/1835449.1835475. URL: https://doi.org/10.1145/1835449.1835475.

Wang, Lucy Lu, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, et al. (July 2020). "CORD-19: The COVID-19 Open Research Dataset". In: *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*. Online: Association for Computational Linguistics. URL: https://aclanthology.org/2020.nlpcovid19-acl.1.

Wang, Shuai, 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. Virtual Event, Canada: Association for Computing Machinery, pp. 317–324. ISBN: 9781450386111. DOI: 10.1145/3471158.3472233. URL: https://doi.org/10.1145/3471158.3472233.

Wang, Wenhui, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou (2020). "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 5776–5788. URL: https://proceedings.neurips.

cc/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, Xiao, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis (July 2021). "Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval". In: *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. ACM. DOI: 10.1145/3471158.3472250. URL: https://doi.org/10.1145%5C%2F3471158.3472250.

Wang, Yujing, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, et al. (2022). "A Neural Corpus Indexer for Document Retrieval". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. URL: https://openreview.net/forum?id=fSfcEYQP_qc.

Wei, Jason, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le (2022). "Finetuned Language Models are Zero-Shot Learners". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=gEZrGCozdqR.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, et al. (Oct. 2020). "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

Wu, Chen and Ruqing Zhang (June 2022). "Are Neural Ranking Models Robust?" In: *ACM Trans. Inf. Syst.* Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3534928. URL: https://doi.org/10.1145/3534928.

Wu, Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng (Dec. 2022). "PRADA: Practical Black-Box Adversarial Attacks against Neural Ranking Models". In: *ACM Trans. Inf. Syst.* Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3576923. URL: https://doi.org/10.1145/3576923.

Wu, Ledell, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston (Apr. 2018). "StarSpace: Embed All The Things!" In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1. DOI: 10.1609/aaai.v32i1.11996. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11996.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, et al. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. DOI: 10.48550/ARXIV.1609.08144. URL: https://arxiv.org/abs/1609.08144.

Wu, Zhuofeng, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma (2020). *CLEAR: Contrastive Learning for Sentence Representation*. DOI: 10.48550/ARXIV.2012.15466. URL: https://arxiv.org/abs/2012.15466.

Xiao, Shitao and Zheng Liu (2022). *RetroMAE v2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models*. DOI: 10.48550/ARXIV.2211.08769. URL: https://arxiv.org/abs/2211.08769.

Xiao, Shitao, Zheng Liu, Yingxia Shao, and Zhao Cao (2022). *RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder*. DOI: 10.48550/ARXIV.2205.12035. URL: https://arxiv.org/abs/2205.12035.

Xiong, Chenyan, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power (2017). "End-to-End Neural Ad-hoc Ranking with Kernel Pooling". In: *Proceedings of the 40th International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM.

Xiong, Lee, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk (2021). "Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=zeFrfgyZln.

Yamada, Ikuya, Akari Asai, and Hannaneh Hajishirzi (Aug. 2021). "Efficient Passage Retrieval with Hashing for Open-domain Question Answering". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, pp. 979–986. DOI: 10.18653/v1/2021.acl-short.123. URL: https://aclanthology.org/2021.acl-short.123.

Yang, Jheng-Hong, Xueguang Ma, and Jimmy Lin (2021). *Sparsifying Sparse Representations for Passage Retrieval by Top-k Masking*. DOI: 10.48550/ARXIV.2112.09628. URL: https://arxiv.org/abs/2112.09628.

Yang, Peilin, Hui Fang, and Jimmy Lin (2017). "Anserini: Enabling the Use of Lucene for Information Retrieval Research". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, pp. 1253–1256. ISBN: 9781450350228. DOI: 10.1145/3077136.3080721. URL: https://doi.org/10.1145/3077136.3080721.

Yang, Wei, Kuang Lu, Peilin Yang, and Jimmy Lin (2019a). "Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models". In: *Pro-*

ceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 1129–1132. ISBN: 9781450361729. DOI: 10.1145/3331184.3331340. URL: https://doi.org/10.1145/3331184.3331340.

Yang, Wei, Haotian Zhang, and Jimmy Lin (2019b). "Simple Applications of BERT for Ad Hoc Document Retrieval". In: *ArXiv* abs/1903.10972. URL: http://arxiv.org/abs/1903.10972.

Yilmaz, Emine, Javed A. Aslam, and Stephen Robertson (2008). "A New Rank Correlation Coefficient for Information Retrieval". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: Association for Computing Machinery, pp. 587–594. ISBN: 9781605581644. DOI: 10.1145/1390334.1390435. URL: https://doi.org/10.1145/1390334.1390435.

Yu, C. T. and G. Salton (Jan. 1976). "Precision Weighting - An Effective Automatic Indexing Method". In: *Journal of the ACM* 23.1, pp. 76–88. ISSN: 0004-5411. DOI: 10/d3fgsz. URL: https://doi.org/10.1145/321921.321930 (visited on 10/20/2021).

Yu, HongChien, Chenyan Xiong, and Jamie Callan (2021). "Improving Query Representations for Dense Retrieval with Pseudo Relevance Feedback". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. Virtual Event, Queensland, Australia: Association for Computing Machinery, pp. 3592–3596. ISBN: 9781450384469. DOI: 10.1145/3459637.3482124. URL: https://doi.org/10.1145/3459637.3482124.

Zamani, Hamed and W. Bruce Croft (2017). "Relevance-Based Word Embedding". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, pp. 505–514. ISBN: 9781450350228. DOI: 10.1145/3077136.3080831. URL: https://doi.org/10.1145/3077136.3080831.

Zamani, Hamed, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps (2018). "From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. Torino, Italy: Association for Computing Machinery, pp. 497–506. ISBN: 9781450360142. DOI: 10.1145/3269206.3271800. URL: https://doi.org/10.1145/3269206.3271800.

Zamani, Hamed, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky (2022). "Retrieval-Enhanced Machine Learning". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 2875–

2886. ISBN: 9781450387323. DOI: 10.1145/3477495.3531722. URL: https://doi.org/10.1145/3477495.3531722.

Zhan, Jingtao, 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*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 1503–1512. ISBN: 9781450380379. DOI: 10.1145/3404835.3462880. URL: https://doi.org/10.1145/3404835.3462880.

— (2022a). "Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. WSDM '22. Virtual Event, AZ, USA: Association for Computing Machinery, pp. 1328–1336. ISBN: 9781450391320. DOI: 10.1145/3488560.3498443. URL: https://doi.org/10.1145/3488560.3498443.

Zhan, Jingtao, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma (2020). *Learning To Retrieve: How to Train a Dense Retrieval Model Effectively and Efficiently*. DOI: 10.48550/ARXIV.2010.10469. URL: https://arxiv.org/abs/2010.10469.

Zhan, Jingtao, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma (2022b). "Evaluating Interpolation and Extrapolation Performance of Neural Retrieval Models". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, pp. 2486–2496. ISBN: 9781450392365. DOI: 10.1145/3511808.3557312. URL: https://doi.org/10.1145/3511808.3557312.

Zhang, Hang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen (2022). "Adversarial Retriever-Ranker for Dense Text Retrieval". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=MR7XubKUFB.

Zhang, Kai, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang (2022). *LED: Lexicon-Enlightened Dense Retriever for Large-Scale Retrieval*. DOI: 10.48550/ARXIV.2208.13661. URL: https://arxiv.org/abs/2208.13661.

Zhao, Tiancheng, Xiaopeng Lu, and Kyusong Lee (June 2021). "SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 565–575. DOI: 10.18653/v1/2021.naacl-main.47. URL: https://aclanthology.org/2021.naacl-main.47.

Zhou, Yujia, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen (2022). *Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer*. arXiv: 2208.09257 [cs.IR].

Zhuang, Honglei, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky (2022). *RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses*. DOI: 10.48550/ARXIV.2210.10634. URL: https://arxiv.org/abs/2210.10634.

Zhuang, Shengyao, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang (2022). *Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation*. DOI: 10.48550/ARXIV.2206.10128. URL: https://arxiv.org/abs/2206.10128.

Zhuang, Shengyao and Guido Zuccon (Nov. 2021a). "Dealing with Typos for BERT-based Passage Retrieval and Ranking". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2836–2842. DOI: 10.18653/v1/2021.emnlp-main.225. URL: https://aclanthology.org/2021.emnlp-main.225.

— (2021b). *Fast Passage Re-ranking with Contextualized Exact Term Matching and Efficient Passage Expansion*. DOI: 10.48550/ARXIV.2108.08513. URL: https://arxiv.org/abs/2108.08513.

— (2021c). "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. Virtual Event, Canada: Association for Computing Machinery, pp. 1483–1492. ISBN: 9781450380379. DOI: 10.1145/3404835.3462922. URL: https://doi.org/10.1145/3404835.3462922.

— (2022). "CharacterBERT and Self-Teaching for Improving the Robustness of Dense Retrievers on Queries with Typos". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 1444–1454. ISBN: 9781450387323. DOI: 10.1145/3477495.3531951. URL: https://doi.org/10.1145/3477495.3531951.

Zobel, Justin and Alistair Moffat (July 2006). "Inverted Files for Text Search Engines". In: *ACM Comput. Surv.* 38.2, 6–es. ISSN: 0360-0300. DOI: 10.1145/1132956.1132959. URL: https://doi.org/10.1145/1132956.1132959.