Efficient and Explainable Neural Ranking

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor rerum naturalium
(abgekürzt: Dr. rer. nat.)

genehmigte Dissertation

von Herrn

M. Sc. Lutz Jurek Leonhardt

geboren am 13. September 1993
in Hannover, Deutschland

2023

Efficient and Explainable Neural Ranking

Dissertation

Lutz Jurek Leonhardt

Referent: Prof. Dr. techn. Wolfgang Nejdl
Leibniz Universität Hannover

Korreferent: Prof. Dr. Avishek Anand
Technische Universiteit Delft

Korreferent: Prof. Dr. Michael Rohs
Leibniz Universität Hannover

Vorsitz: Prof. Dr. Ziawasch Abedjan
Leibniz Universität Hannover

Tag der Promotion: 13. Dezember 2023

Abstract

The recent availability of increasingly powerful hardware has caused a shift from traditional information retrieval (IR) approaches based on term matching, which remained the state of the art for several decades, to large pre-trained neural language models. These neural rankers achieve substantial improvements in performance, as their complexity and extensive pre-training give them the ability of *understanding* natural language in a way. As a result, neural rankers go beyond term matching by performing relevance estimation based on the semantics of queries and documents.

However, these improvements in performance don't come without sacrifice. In this thesis, we focus on two fundamental challenges of neural ranking models, specifically, ones based on large language models: On the one hand, due to their complexity, the models are *inefficient*; they require considerable amounts of computational power, which often comes in the form of specialized hardware, such as GPUs or TPUs. Consequently, the carbon footprint is an increasingly important aspect of systems using neural IR. This effect is amplified when low latency is required, as in, for example, web search. On the other hand, neural models are known for being inherently *unexplainable*; in other words, it is often not comprehensible for humans *why* a neural model produced a specific output. In general, explainability is deemed important in order to identify undesired behavior, such as bias.

We tackle the *efficiency challenge* of neural rankers by proposing Fast-Forward indexes, which are simple vector forward indexes that heavily utilize pre-computation techniques. Our approach substantially reduces the computational load during query processing, enabling efficient ranking solely on CPUs without requiring hardware acceleration. Furthermore, we introduce BERT-DMN to show that the *training efficiency* of neural rankers can be improved by training only parts of the model.

In order to improve the *explainability* of neural ranking, we propose the Select-And-Rank paradigm to make ranking models *explainable by design*: First, a query-dependent subset of the input document is extracted to serve as an *explanation*; second, the ranking model makes its decision based only on the extracted subset, rather than the complete document. We show that our models exhibit performance similar to models that are not explainable by design and conduct a user study to determine the faithfulness of the explanations.

Finally, we introduce BoilerNet, a web content extraction technique that allows the removal of *boilerplate* from web pages, leaving only the main content in plain text. Our method requires no feature engineering and can be used to aid in the process of creating new document corpora from the web.

Keywords: Information retrieval, neural ranking, efficiency, explainability

Zusammenfassung

Die Verfügbarkeit zunehmend leistungsstärkerer Hardware hat eine Verlagerung im Bereich des Information Retrieval (IR) von traditionellen, auf dem Abgleich von Termen basierenden Ansätzen, die mehrere Jahrzehnte lang dem Stand der Technik entsprachen, hin zu großen vortrainierten neuronalen Sprachmodellen ausgelöst. Diese neuronalen Rankingmodelle erzielen deutlich bessere Ergebnisse, da sie durch ihre hohe Komplexität und umfangreiche Trainingsdaten die Fähigkeit erlangen, Sprache in gewissem Maße zu *verstehen*. Dies erlaubt neuronalen Rankingmodellen, die Relevanz von Dokumenten nicht nur mittels Termabgleich, sondern auch auf Basis ihrer Semantik abzuschätzen.

Die oben angesprochenen Leistungssteigerungen haben jedoch ihren Preis. Diese Dissertation legt ihren Fokus auf zwei grundlegende Probleme neuronaler Rankingmodelle: Einerseits weisen sie, aufgrund ihrer Komplexität, eine *Ineffizienz* auf, die sich in ihren außergewöhnlich hohen Anforderungen an Rechenleistung niederschlägt; dies hat oft zur Folge, dass spezialisierte Hardware, wie GPUs oder TPUs, benötigt wird. Der ökologische Fußabdruck spielt daher bei Systemen, die neuronale Modelle einsetzen, eine immer größere Rolle. Dieser Effekt wird zusätzlich verstärkt, wenn kurze Antwortzeiten wichtig sind, wie etwa bei einer Suchmaschine im Web. Andererseits sind neuronale Modelle dafür bekannt, *unerklärbar* zu sein; es ist für Menschen oft nicht nachvollziehbar, *warum* ein solches Modell bestimmte Entscheidungen trifft. Erklärbarkeit wird jedoch als wichtige Eigenschaft angesehen, um unerwünschtes Verhalten, wie etwa Vorurteile, zu erkennen.

Wir gehen das Problem der *eingeschränkten Effizienz* von neuronalen Rankingmodellen mithilfe von Fast-Forward Indexen, die eine simple Struktur für Speicherung und Zugriff vorberechneter Vektoren darstellen, an. Unser Ansatz reduziert die für eine Anfrage benötigte Rechenleistung wesentlich und ermöglicht es dadurch, Ranking effizient ausschließlich auf CPUs durchzuführen. Weiterhin stellen wir BERT-DMN vor und zeigen, dass die *Trainingseffizienz* neuronaler Rankingmodelle verbessert werden kann, wenn nur Teile des Modells trainiert werden.

Um die *Erklärbarkeit* neuronaler Rankingmodelle zu verbessern, stellen wir mit Select-And-Rank ein Paradigma vor, das Rankingmodelle *von sich aus erklärbar* macht: Im ersten Schritt wird ein Teil des Dokumentes, abhängig von der Anfrage, extrahiert, um als *Erklärung* zu dienen; im zweiten Schritt schätzt das Rankingmodell die Relevanz des Dokuments nur auf Basis der Erklärung, anstelle des gesamten Dokumentes. Wir zeigen, dass die Leistung unserer Modelle vergleichbar ist mit der von Modellen, die nicht von sich aus erklärbar sind, und führen eine Studie durch, um den Nutzen der Erklärungen zu verdeutlichen.

Zuletzt stellen wir BoilerNet, einen Ansatz zur *Extraktion von Webinhalten*, vor, der sämtliche Strukturen von Webseiten entfernt und nur den Inhalt als Klartext ausgibt. Dies kann zum Erstellen und Indexieren von Korpora, die auf Webseiten basieren, eingesetzt werden.

Schlagworte: Information Retrieval, neuronale Rankingmodelle, Effizienz, Erklärbarkeit

Acknowledgements

I was very fortunate to be able complete my PhD as part of a group of talented researchers at L3S. I want to thank Prof. Wolfgang Nejdl for giving me this opportunity, allowing me to choose my area of research freely, and offering excellent advice throughout the course of my studies. I am also grateful for the opportunities to travel, attend conferences, and meet new people, which were plenty.

I would also like to thank Prof. Avishek Anand, who has been my mentor and daily advisor from day one and provided incredible guidance ever since. His continued support in the form of feedback, discussions, and new ideas has influenced me greatly and made the journey so much easier and more enjoyable. Furthermore, I thank Prof. Michael Rohs for agreeing to be my examiner for this thesis without hesitation, and Prof. Ziawasch Abedjan for chairing the committee.

Over the years, I have worked and collaborated with great colleagues and students at L3S, many of which have become my friends. The list of names is too long to mention them all, but you know who you are. A sincere thank you, in no particular order, to my co-authors Megha Khosla, Koustav Rudra, Abhijit Anand, Fabian Beringer, Henrik Müller, and Marcel Jahnke, and to Florian Supplie, whose master's thesis work helped us with one of our publications that made it into this thesis.

Finally, my special thanks go to my family, friends, and my girlfriend, Lijun, for always being there for me and offering amazing help and support whenever I needed it. This wouldn't have been possible without you.

Foreword

This dissertation is cumulative. Chapters 3 to 6 each correspond to a publication, respectively. The content of these chapters is taken, for the most part, verbatim from the papers, save for some changes, such as the correction of minor mistakes or adjustments in order to unify the notation. Additionally, some plots and figures have been remade, and some preliminaries have been moved to Chapter 2.

In Chapter 3, we propose our main contribution, Fast-Forward indexes. The chapter is based on a full paper, which is published in the ACM Web Conference, and an extension to appear as an article in the ACM TOIS journal. Furthermore, we participated in the TREC 2021 Deep Learning track:

- Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. “Efficient Neural Ranking using Forward Indexes”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 266–276. ISBN: 9781450390965. DOI: 10.1145/3485447.3511955. URL: https://doi.org/10.1145/3485447.3511955 [111]

- Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. “Efficient Neural Ranking using Forward Indexes and Lightweight Encoders”. In: *ACM Trans. Inf. Syst.* (Nov. 2023). Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3631939. URL: https://doi.org/10.1145/3631939 [112]

- Jurek Leonhardt, Koustav Rudra, and Avishek Anand. *L3S at the TREC 2021 Deep Learning Track*. 2021 [110]

Our work in Chapter 4, BERT-DMN, is published in the LWDA workshop:

- Jurek Leonhardt, Fabian Beringer, and Avishek Anand. “Exploiting Sentence-Level Representations for Passage Ranking”. In: *Proceedings of the LWDA 2021 Workshops: FGWM, KDML, FGWI-BIA, and FGIR, Online, September 1-3, 2021*. Ed. by Thomas Seidl, Michael Fromm, and Sandra Obermeier. Vol. 2993. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 287–302. URL: https://ceur-ws.org/Vol-2993/paper-27.pdf [107]

In Chapter 5, we propose the Select-And-Rank paradigm for explainable-by-design neural ranking, which is published as an article in the ACM TOIS journal:

- Jurek Leonhardt, Koustav Rudra, and Avishek Anand. "Extractive Explanations for Interpretable Text Ranking". In: *ACM Trans. Inf. Syst.* 41.4 (Mar. 2023). issn: 1046-8188. doi: 10.1145/3576924. url: https://doi.org/10.1145/3576924 [109]

In Chapter 6, we present BoilerNet, a web content extraction method, which is published as a demonstration paper in the ACM Web Conference:

- Jurek Leonhardt, Avishek Anand, and Megha Khosla. "Boilerplate Removal using a Neural Sequence Labeling Model". In: *Companion Proceedings of the Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 226–229. isbn: 9781450370240. doi: 10.1145/3366424.3383547. url: https://doi.org/10.1145/3366424.3383547 [105]

Some of the evaluation in this paper was conducted by Supplie [189] as part of his master's thesis, including the preparation of datasets, the setup and execution of baseline methods, and the manual annotation of web pages.

Although not part of this thesis, the following preprint proposes a training approach for more efficient neural retrieval models:

- Jurek Leonhardt, Marcel Jahnke, and Avishek Anand. *Distribution-Aligned Fine-Tuning for Efficient Neural Retrieval*. 2022. arXiv: 2211.04942 [cs.IR] [108]

The first two authors have contributed equally to this paper.

The complete list of published papers and articles that I was involved in during my PhD can be found in Appendix A.

Contents

1

Introduction

The evolution and adoption of the World Wide Web over the past decades have enabled access to close to arbitrary amounts of knowledge to a significant part of the population, making the process of information acquisition as easy as it has ever been. Therefore, the usage of web search engines, such as *Google* or *Bing*, and personal assistants, such as Apple's *Siri* or Amazon's *Alexa*, has become second nature and a part of everyday life for many people. In the year 2012, the total number of queries issued by users to the Google search engine was approximately 38 000 per second.[1] Despite the lack of official statistics in subsequent years, recent estimations suggest that this trend is still increasing steadily.[2]

As a consequence, the task of *information retrieval* (IR) has become ubiquitous: Aside from web search as its most obvious use case, IR powers the backbones of a multitude of downstream tasks that require specific information from a large corpus of unstructured data at some point within their pipeline; notable examples are *question answering*, as used, for example, by personal assistants, or *recommender systems*, which are prevalent in e-commerce and streaming services. Naturally, many of the services that employ IR are backed by large companies, and for many of these companies, their IR system directly influences the quality of the services they offer, creating financial interests in its improvement. As such, research in the field of IR is being driven by industry and academia alike, which makes for rapid advances and frequent innovations.

[1] `http://archive.google.com/zeitgeist/2012/` (accessed on April 4th, 2023)
[2] `https://live-counter.com/google-searches/` (accessed on April 4th, 2023)

Query: *What is the top speed of a Jaguar F-Type?*

Document 1
The Jaguar F-Type is a series of vehicles manufactured by Jaguar Land Rover [...] The car reaches a top speed of 322 km/h [...]

Document 2
The jaguar is a species of cats [...] Jaguars can reach a top speed of up to 80 km/h [...]

Figure 1.1: An example that demonstrates the limitations of lexical retrievers. The query term "*Jaguar*" is ambiguous in the sense that both cars and animals have a top speed. The words that match in both documents are highlighted. As a consequence, a lexical retriever would likely assign high relevance scores to both documents. However, from the query it is clear that only the first document is relevant. Neural semantic models are able to perform better due to their natural language understanding capabilities.

1.1 Ad-hoc Document Retrieval

At the heart of services that exploit textual resources from large unordered corpora lies the *ad-hoc document retrieval* task. A *ranking model* returns a list of documents (or passages), which are ordered by their respective relevance to an *input query*. Machine learning-based *learning to rank* [129] models may subsequently be used to take additional features, such as user interactions, into account.

Traditional retrieval algorithms, such as BM25 [169, 168] or the query likelihood model [102], perform *lexical matching* (or *term matching*) to estimate the relevance of a document in a corpus with respect to a query and require no training data. These approaches are typically called *sparse retrievers*, as they represent documents using high-dimensional sparse vectors, where the elements usually correspond to some vocabulary obtained from the document corpus.

Although sparse retrieval has prevailed as the state of the art for a long time, it has inherent limitations that render it less and less adequate; since matching is performed solely based on the query and document terms, sparse models are highly susceptible to the likes of spelling errors or synonyms. We refer to this as the *vocabulary mismatch problem*. Moreover, they do not possess any *semantic* capabilities, which would allow them to perform reasoning (albeit limited) within documents. This limitation can be alleviated only to an extent by using pseudo relevance feedback techniques like RM3 [1]. An example of the shortcomings of lexical matching is illustrated in Fig. 1.1. For this reason, alternative approaches have emerged, where a sparse retriever is used to perform high-recall candidate selection (*retrieval stage*), followed by a *re-ranking stage* using a computationally more expensive semantic model. These approaches are referred to as *telescoping* [144] or *retrieve-and-re-rank* [179, 64]. The semantic re-ranking models are usually implemented using neural networks and are hence called *neural rankers*.

1.2 Neural Ranking Models

Recently, the advent of large pre-trained neural language models (LLMs) based on the Transformer architecture [197] ushered in a new era, when BERT [40] achieved state-of-the-art performance on a variety of language tasks, including re-ranking [155], where it significantly outperformed previous semantic rankers. This is largely due to two important aspects of LLMs: First, their self-attention layers allow for vastly improved *natural language understanding*, resulting in the capability of determining the intent of ambiguous queries or even resolving synonyms. Second, the size of the models (in terms of parameters), along with the large amounts of pre-training data, allows them to scale almost arbitrarily [152]. Consequently, neural rankers based on LLMs can overcome the limitations of lexical models (cf. Fig. 1.1).

Shortly thereafter, the first *dense retrieval* methods emerged [89], which allow for the usage of neural semantic LLMs directly for retrieval instead of re-ranking. Nowadays, commercial search engines employ LLMs to rank documents. For example, Google has been using BERT for query processing since 2019.[3] Nonetheless, sparse retrievers remain relevant, mainly for their simplicity and efficiency, both in terms of query latency and resource consumption, and are often used in retrieve-and-re-rank or hybrid retrieval settings.

Neural models, specifically large language models, have achieved appreciable performance improvements in IR-related tasks, most notably, passage or document ranking. However, they aren't "silver bullets"; in fact, neural rankers exhibit two inherent drawbacks: First, LLMs are notoriously inefficient and require powerful hardware. Second, decisions made by complex neural models are difficult to understand and explain. In this thesis, we focus on these challenges; our goal is to make neural IR more efficient and explainable.

1.2.1 The *Efficiency Challenge*

When it comes to web search, query latency, that is, the time it takes between the user issuing a query and the search engine displaying the results, is essential. In fact, according to Google,

[3] https://blog.google/products/search/search-language-understanding-bert/ (accessed on April 4th, 2023)

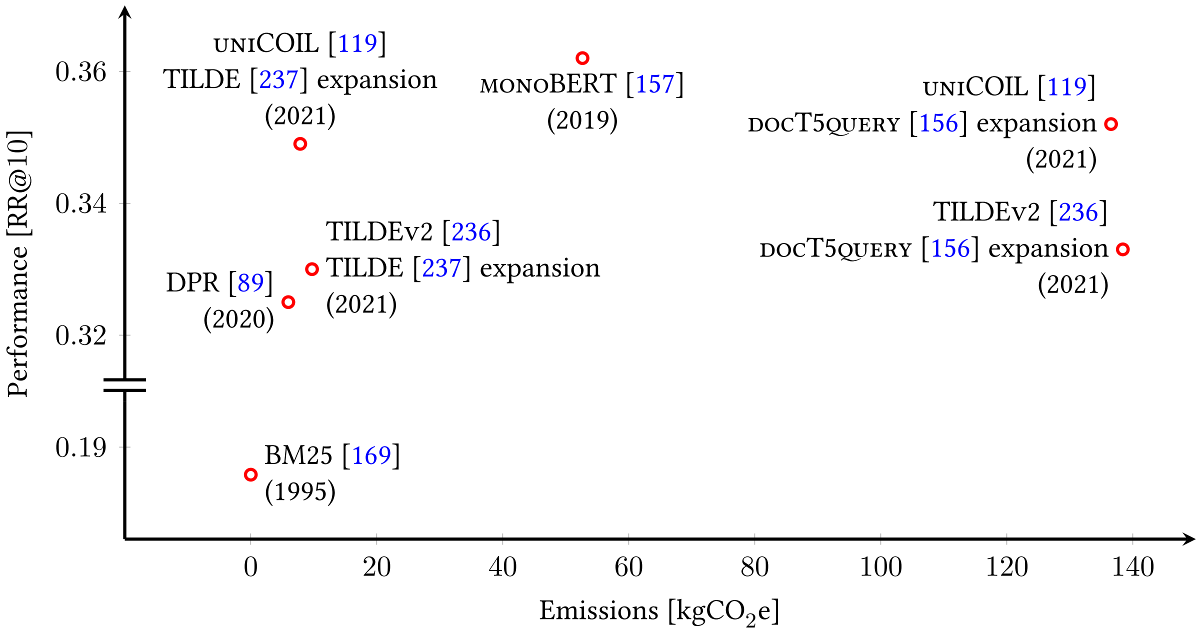


Figure 1.2: The carbon emissions caused by retrieval and ranking models [175]. The newer, more effective approaches exhibit substantial growth in terms of emissions.

a delay of 400 milliseconds causes a drop of 0.44% in search volume.[4] The same holds true for downstream tasks, where the retrieval step is only one part of a pipeline of operations.

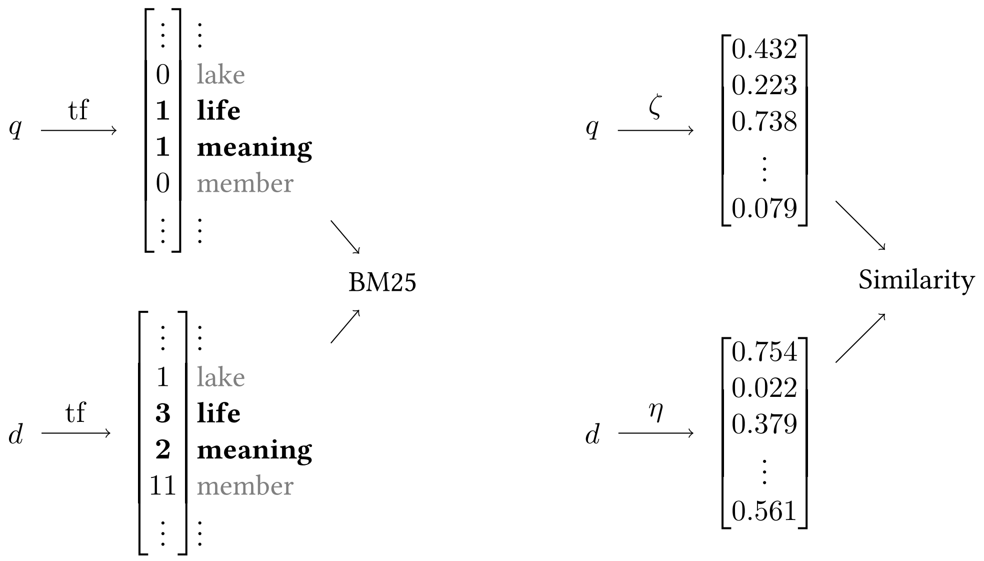
The LLMs used in neural IR are very large and computationally expensive. Consequently, in order to keep the latency at an acceptable level, hardware acceleration and approximative techniques, such as product quantization, are used [86]. Although hardware acceleration using specialized devices, such as GPUs or TPUs, improves the latency substantially, it has negative implications on the cost (in the sense of both initial outlay and power draw) and, more importantly, the environmental footprint (cf. Fig. 1.2). Despite the importance of efficiency, most of the (recent) work on neural IR seems to focus on effectiveness. Notable exceptions are SPLADE [49, 99], TILDE [237, 236], and the *ReNeuIR* workshop [18].

1.2.2 The *Explainability* Challenge

Neural models, especially LLMs, are essentially black boxes, i.e., inherently obscure and uninterpretable [171].[5] In the context of neural ranking, this means that it is usually hard to

[4] https://www.thinkwithgoogle.com/future-of-marketing/digital-transformation/the-google-gospel-of-speed-urs-hoelzle/ (accessed on April 4th, 2023)

[5] In this thesis, we use the terms "*explainability*" and "*interpretability*" interchangeably; in the literature, one of them often refers to human-understandable justifications of model outputs (i.e., *why* a prediction was



(a) Sparse retrieval

(b) Dense retrieval

Figure 1.3: Sparse and dense retrieval side-by-side for the query "*What is the meaning of life?*". Sparse retrievers (Fig. 1.3a) allow for interpretability by design using term frequencies (denoted by tf). Dense retrievers (Fig. 1.3b) utilize internal low-dimensional representations computed by neural encoders (denoted by ζ and η), which are not easily interpretable.

tell *why* a model assigned a high (or low) relevance to a specific document given a query. In other words, the ranking decisions cannot easily be *explained*. This is in contrast to other non-neural models, such as BM25, which are interpretable by design, in that they explicitly assign weights to specific terms in the documents to be ranked (cf. Fig. 1.3).

However, interpretability is deemed an important property of machine learning models.[6] On the one hand, it helps during the development stage to discover potential bugs, for example, within the implementation, training data, or evaluation. An example for this is *label leakage*, where a model input encodes the relevance of the document to the query in some unintended way, even though this information should obviously not be available to the ranker. Those cases are often called *right for the wrong reasons*. On the other hand, explanations are used to uncover *biased ranking decisions* the model might have learned from flawed training data. Consider an example, where a model has been trained using data that was annotated by humans who introduced bias, for example, by sharing common beliefs, or, even worse, racism or sexism. As a result, a search engine might rank documents based on the race or

made), whereas the other focuses on understanding the model itself (i.e., *how* a prediction was made), but we found that these definitions are not always consistent [58, 126].

[6] For example, the proposed EU AI law ("The AI Act"), available at https://artificialintelligenceact.eu/the-act/ (accessed on April 4th, 2023), stipulates transparency and interpretability as a requirement for high-risk AI systems.

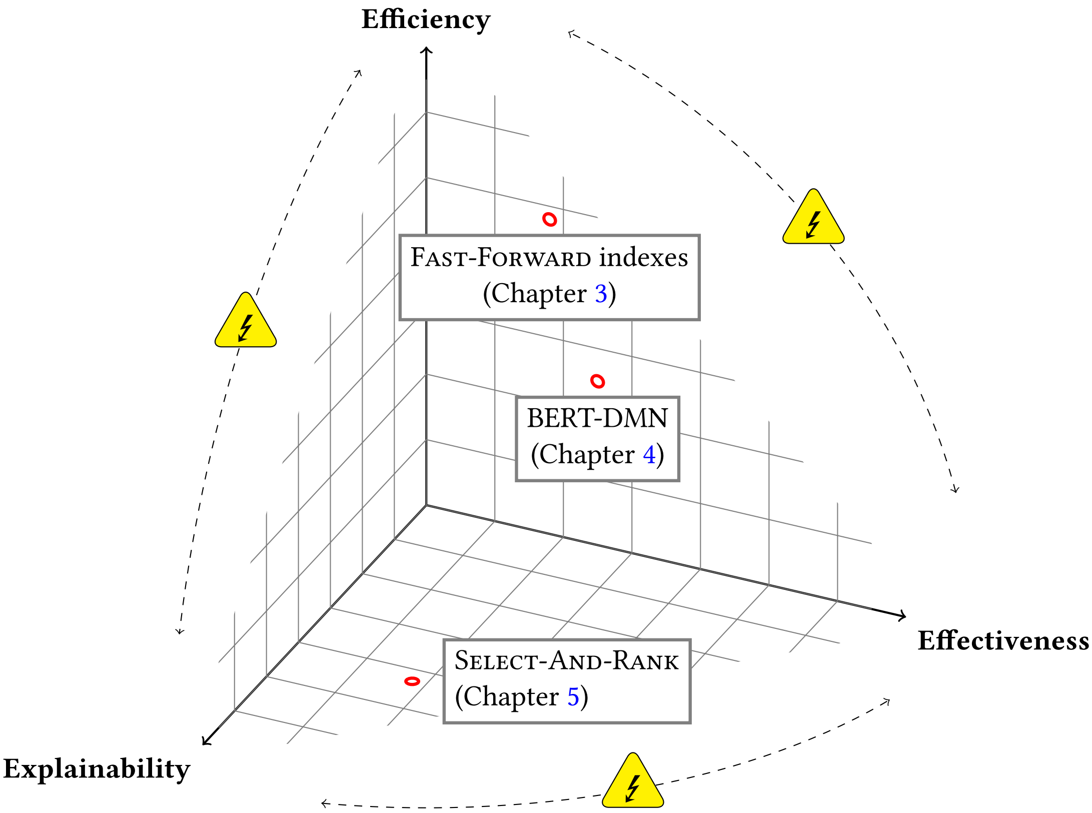


Figure 1.4: An overview of contributions in this thesis. The three properties of ranking models, *effectiveness*, *efficiency*, and *explainability*, conflict each other in the sense that optimizing a model for one of them is usually rather straight-forward, as long as the others are sacrificed; combining two or even all three of them in a single approach, however, is much more difficult. This figure illustrates the goal of the contributions in this thesis with respect to these properties. Note that BoilerNet (Chapter 6) is omitted here, as it is not a ranking model.

gender of their authors. If the specific reasons behind a ranking decision are known, such cases can be detected and mitigated.

1.3 Outline and Contributions

There is an inherent trade-off between effectiveness, efficiency, and explainability of neural ranking models. Figure 1.4 visualizes this trade-off and illustrates the focus of each corresponding contribution. In this section, we give an outline of the thesis and highlight the contributions.

1.3.1 Efficiency in Neural IR

The first part of this thesis focuses on the **efficiency challenge** of neural information retrieval.

Chapter 3 introduces our main contribution, Fast-Forward indexes. We show that dual-encoder models, which are commonly used for dense retrieval, are perfectly suitable for use as re-rankers; we find that, in the retrieve-and-re-rank setting, these models achieve performance similar to cross-attention rankers, despite their lack of query-document attention, when the relevance scores of the sparse first-stage retriever are combined with the scores of the re-ranking model using simple linear interpolation. At the same time, the dual-encoder architecture allows the pre-computation of document representations, which would otherwise be the most expensive component of the re-ranking pipeline, and leaves only the query representation to be computed online. We show that our approach, run on CPUs only, is faster than cross-attention re-rankers run with GPU acceleration, and still achieves comparable performance. Furthermore, we propose several techniques and extensions to improve the efficiency of Fast-Forward indexes even more: In Section 3.3, we propose more efficient and lightweight encoder models. In Section 3.2.2, we propose an *early stopping* technique, which dynamically limits the number of documents to be re-ranked for a query based on estimated limits of the scores. Finally, in Section 3.2.1, we propose *sequential coalescing*, an index compression technique for dual-encoder-based indexes.

Furthermore, in Chapter 4, we propose BERT-DMN, an extension of BERT that takes sentence-level representations into account. The light version of this model, BERT-DMN$_{\text{lite}}$, trains only a smaller network instead of fine-tuning BERT itself; this approach improves the *training efficiency* by caching the representations to avoid expensive forward passes, as we show in Section 4.4.3.

1.3.2 Explainability in Neural IR

The second part of this thesis corresponds to the **explainability challenge** of neural information retrieval.

In Chapter 5, we propose Select-And-Rank, a paradigm for neural re-ranking models that are *interpretable by design*. Specifically, Select-And-Rank models have two components: A *selector* selects an extractive, query-dependent explanation as a subset of sentences from the input document. Afterwards, a *ranker* computes the relevance score based on the query and only the selected sentences. We conduct a study in Section 5.4.3 to show that the explanations output by our models are useful to human users. Furthermore, we show that how model successfully uncovers a case of label leakage in Section 5.6.1.

1.3.3 The Big Picture

The final part of the thesis deals with the **IR process** as a whole, introducing methods that are not used directly in the ranking process, but rather serve as utilities in the IR pipeline.

In Chapter 6, we propose BoilerNet, a web content extraction method based on a simple deep recurrent neural network architecture. We show that the model requires very little training data to be effective in removing *boilerplate*, i.e., any HTML tags in the DOM tree that do not correspond to actual content, from web pages. In the context of IR, BoilerNet can be used to index websites in order do create document corpora.

Finally, we conclude the thesis in Chapter 7, giving a summary of contributions in terms of both research conducted and software released. In Section 7.2, we present possible limitations of the work conducted in this thesis and discuss promising directions for future work, and in Section 7.3, we outline a vision on how this work could be used to advance the field of neural IR.

2

Background

This chapter introduces core concepts that are essential to the work in this thesis. Section 2.1 describes text ranking techniques, ranging from classical term-matching to modern neural approaches. Section 2.2 deals with the fundamentals of large language models.

2.1 Text Ranking

The process of *text ranking* entails creating an *ordered list* of text documents (for example, single sentences, passages, or complete web pages), such that the order of the list reflects the relevance of each item with respect to an *input query* (i.e., a short sentence or some keywords). The collection of documents that are to be ranked is commonly referred to as *corpus*, the size of which varies greatly depending on the task, domain, and other factors.

More formally, we express queries and documents as sequences of terms (or tokens), i.e.,

$$q = \left(t_1^q, \ldots, t_{|q|}^q\right) \quad \text{and} \quad d = \left(t_1^d, \ldots, t_{|d|}^d\right) . \tag{2.1}$$

Let the corpus $\mathcal{D}$ be a set of documents d_i. A *ranking*

$$R = \left(d_1^R, \ldots, d_{|R|}^R\right) , \quad \text{where} \quad d_i^R \in \mathcal{D}, \tag{2.2}$$

defines an order over the documents in the corpus with respect to the query q, such that

$$\phi\left(q, d_i^R\right) \geq \phi\left(q, d_j^R\right) \Leftrightarrow i < j \tag{2.3}$$

for all distinct pairs of integers $1 \leq i, j \leq |R|$.[1] The function $\phi(q, d)$ is used to estimate the relevance of a document d given a query q. Note that Eq. (2.3) describes the *pointwise* approach, where a ranking model estimates the relevance independently for each document; *pairwise* and *listwise* approaches also exist, predominantly in the *learning-to-rank* setting [129], but they are omitted here.

The estimation of the relevance of a document to a query (i.e., the function $\phi(q, d)$ in Eq. (2.3)) is often referred to as *matching*. This section introduces various matching approaches, loosely following their chronological order: Section 2.1.1 focuses on *lexical* approaches, which employ term matching in order to estimate relevance. Section 2.1.2 focuses on *semantic* models and representation learning. Finally, Section 2.1.3 shows how different matching approaches are combined in practice to be used in text retrieval systems.

2.1.1 Lexical Matching

Retrieval methods based on term matching (or *lexical* matching) date back as far as the 1980s [120]. Arguably the most popular one, Okapi BM25, was originally proposed by Robertson et al. [169], but has since been slightly refined by omitting unused parameters [120, 168]. It computes the relevance of a query-document pair (q, d) as

$$\phi_{\text{BM25}}(q, d) = \sum_{t \in q \cap d} \log \frac{N - \mathrm{df}(t) + 0.5}{\mathrm{df}(t) + 0.5} \cdot \frac{\mathrm{tf}(t, d) \cdot (k_1 + 1)}{\mathrm{tf}(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{l_d}{L}\right)}, \tag{2.4}$$

making use of *document frequencies* $\mathrm{df}(t)$, i.e., the number of documents the term t appears in, and *term frequencies* $\mathrm{tf}(t, d)$, i.e., the number of occurrences of t in d. Furthermore, N is the total number of documents, l_d is the length of d, L is the average document length, and b and k_1 are parameters. Due to its exploitation of term frequencies (tf) along with inverse document frequencies (idf—computed in the first factor within the sum in Eq. (2.4)), BM25 belongs to the family of *tf-idf*-based rankers.

The implementation of term frequency and document frequency look-ups for a given term is commonly realized using *inverted indexes* [141], a structure that maps each term in the vocabulary to a list of documents it appears in along with the corresponding number of occurrences. Inverted indexes allow for very efficient implementations of BM25 or tf-idf methods in general, the most popular one being Apache Lucene.[2] However, the order in which words appear in the documents is lost as they are indexed.

[1] Note that, usually, only a very small fraction of all documents in the corpus is relevant for a given query, and, thus, the number of documents in a ranking is chosen to be relatively small, i.e., $|R| \ll |\mathcal{D}|$.

[2] https://lucene.apache.org/

Another important limitation is the fact that BM25 performs so-called *exact matching*; this becomes apparent in Eq. (2.4), as the sum is computed only over the intersection of query and document terms ($q \cap d$). Thus, any document term that does not appear in the query (and vice versa) is not taken into account. As a result, the use of synonyms, paraphrasing, or even simple spelling mistakes directly negatively affect the performance of exact matching algorithms. This limitation is referred to as the *vocabulary mismatch problem.* To some extent, it can be mitigated; for example, pre-processing often involves stemming operations, i.e., removing suffixes and thus only keeping the root of each term. Other approaches employ query expansion techniques, such as the widely used RM3 [1], which expands the query using pseudo-relevance feedback. Nonetheless, despite those efforts, the vocabulary mismatch remains an issue of tf-idf-based methods,[3] prompting the need for models capable of semantic matching and natural language understanding.

2.1.2 Semantic Matching

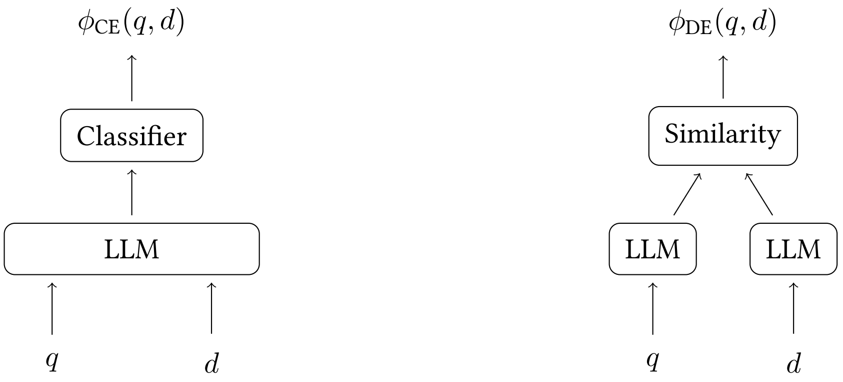
Contrarily to lexical term matching-based models (cf. Section 2.1.1), the estimation of *semantic* relevance scores enables rankers to capture the meaning of queries and documents through natural language understanding (NLU). Semantic matching thus goes beyond term matching and the limitation of vocabulary mismatch.

Although the learning-to-rank era already saw frequent use of neural models for ranking, an early example being RankNet [19], these approaches were mainly characterized by their usage of a large number of hand-crafted features [120].[4] As time progressed, subsequent approaches began to move away from feature engineering and towards operating directly on query and document tokens instead, setting them apart from learning-to-rank models. Common approaches include *learning representations* of queries and documents, often making use of pre-trained word embeddings like Word2Vec [146] or GloVe [162], or performing *query-document interactions* and subsequent pooling operations [120]. We refer this class of models as *semantic matching*-based, with popular examples being DSSM [77] and DESM [148], K-NRM [220] and Conv-KNRM [37], PACRR [79] and Co-PACRR [78], and DUET [147].

Shortly thereafter, the Transformer architecture [197] was introduced, which exploits self-attention layers in order to compute contextual token representations. Nogueira and Cho [155] showed that BERT [40], a large pre-trained language model based on the Transformer, achieved considerable performance improvements on the MS MARCO passage ranking task

[3] or any method based on term matching for that matter, another popular example being the query likelihood model [102]

[4] As learning-to-rank is not relevant for this thesis, we do not introduce it in detail here.



(a) Cross-encoder **(b)** Dual-encoders

Figure 2.1: Two common model architectures for LLM-based semantic rankers. Cross-encoders operate on a concatenation of the query and document, while dual-encoders compute independent representations.

compared to existing approaches. Since then, most state-of-the-art neural ranking models have been based on BERT or similar LLMs; they exhibit a multitude of semantic capabilities, such as *semantic clustering* or *entity matching* [3], making them much more robust to queries containing paraphrasing. Recently, Zhuang and Zuccon [235] showed that Transformer-based rankers can also be made more robust to queries with spelling errors.

In the remainder of this section, we focus on these LLM-based neural ranking models; we introduce the cross-encoder and dual-encoder architectures and show how they are trained.

Cross-Encoder Models

Nogueira and Cho [155] used a large pre-trained language model, more specifically, BERT, for passage ranking. The term *cross-encoder* refers, on the one hand, to the model architecture and, on the other hand, the way the model inputs and outputs are handled. Figure 2.1a shows an illustration of the cross-encoder architecture; its main characteristic is that the query and document are concatenated before being fed through the LLM. In the case of BERT models, the input for a query-document pair (q, d) is commonly constructed as

$$\text{BERT}(q, d) \equiv \text{BERT}\left(\texttt{[CLS]}, q_1, \dots, q_{|q|}, \texttt{[SEP]}, d_1, \dots, d_{|d|}, \texttt{[SEP]}\right). \tag{2.5}$$

As BERT uses WordPiece tokenization [215], q_i and d_i are subwords, and [CLS] and [SEP] are special tokens. Even though Transformer-based models output a vector for each input token, the most common approach is to only use the output vector corresponding to the *classification token* [CLS] and discard the rest. The relevance thus computes as

$$\phi_{\text{CE}}(q,d) = \mathbf{W} \cdot \text{BERT}_{\texttt{[CLS]}}(q,d) + b, \tag{2.6}$$

where $\mathbf{W} \in \mathbb{R}^{1 \times d_{\text{BERT}}}$ and $b \in \mathbb{R}^1$ are the trainable parameters of a linear layer that outputs a scalar value.

Cross-encoder models are typically trained using *pointwise* or *pairwise* objectives. The pointwise loss function used by Nogueira and Cho [155],

$$\mathcal{L}_{\texttt{pointwise}}(q,d,y) = -y \log \phi(q,d) - (1-y) \log \left(1 - \phi(q,d)\right), \tag{2.7}$$

minimizes the binary cross-entropy, where $y \in \{0,1\}$ is a ground-truth relevance label. In contrast, for pairwise training, a hinge loss can be used [39], i.e.,

$$\mathcal{L}_{\texttt{pairwise}}\left(q,d^+,d^-\right) = \max\left\{0, m - \phi\left(q,d^+\right) + \phi\left(q,d^-\right)\right\}, \tag{2.8}$$

where m is a *margin* and d^+ is *more relevant* to q than d^-.

Dual-Encoder Models

The *dual-encoder* (also referred to as *two-tower*) architecture [89] (cf. Fig. 2.1b) employs two models to compute dense vector representations of queries and documents, respectively. Specifically, a *query encoder* ζ and a *document encoder* η map queries and documents to representations in a common a-dimensional vector space. The relevance score $\phi_{\text{DE}}(q,d)$ of a query-document pair is then computed as the similarity of their vector representations. A common choice for the similarity function is the dot product, such that

$$\phi_{\text{DE}}(q,d) = \zeta(q) \cdot \eta(d), \tag{2.9}$$

where $\zeta(q), \eta(d) \in \mathbb{R}^a$. The *maxP* approach [36] splits long documents into passages, and the score of a document is then computed as the maximum of its passage scores, i.e.,

$$\phi(q,d) = \max_{p_i \in d} \phi(q,p_i). \tag{2.10}$$

Dual-encoders encode the query and document independently, i.e., no query-document attention exists. Typically, they are trained using a *contrastive* loss function [89],

$$\mathcal{L}_{\texttt{contrastive}}\left(q, d^+, D^-\right) = -\log\left(\frac{\exp\left(\phi(q, d^+)/\tau\right)}{\sum_{d \in D^- \cup \{d^+\}} \exp\left(\phi(q, d)/\tau\right)}\right), \tag{2.11}$$

where a training instance consists of a query q, a positive (relevant) document d^+, and a set D^- of negative (irrelevant) documents. The temperature τ is a hyperparameter. Since it is usually infeasible to include all negative documents for a query in D^-, there are various negative sampling approaches, such as in-batch strategies [89], asynchronous indexes [221], or negative caches [127]. Zhan et al. [230] showed that the negative sampling technique has a considerable impact on the ranking performance of the model.

2.1.3 Retrieval Approaches

The previous section focused on *matching*, i.e., the estimation of query-document relevance. This section introduces several *retrieval* paradigms that employ matching techniques in order to generate a ranking of documents from a large corpus based on their relevance to a query.

Sparse Retrieval

Lexical matching methods based on tf-idf and inverted indexes, for example, BM25 (cf. Section 2.1.1), can directly be used to perform document retrieval efficiently on large corpora. The term *sparse retrieval* is often used to denote such retrievers and set them apart from other approaches [131], such as dense retrieval (cf. Section 2.1.3); it originates from the *vector space model* [174], where a document (or a query) is represented by a vector. Each of the vector's elements corresponds to a specific term in the vocabulary and holds a weight based on its frequency in the document. Consequently, the dimension of the vectors is equal to the size of the vocabulary, and most of each vector's elements are zero, making them *sparse*. The relevance of a document w.r.t. a query can then be computed as the similarity of the two vector representations.

In sparse retrieval, we denote the top-k_S documents retrieved from the inverted index for a query q by K^q_S. The sparse score of a query-document pair (q, d) is denoted by $\phi_S(q, d)$.

Retrieve-and-Re-Rank

Sparse retrievers are popular for their efficiency, allowing for the scoring of all documents in the corpus in acceptable time. However, due to their lack of semantic capabilities, they are limited (see Fig. 1.1). The idea of the *retrieve-and-re-rank* approach [179] is to perform retrieval in two stages: In the first stage, a term frequency-based (sparse) retrieval method (such as BM25 [169]) retrieves a set of documents from a large corpus. In the second stage, another model *re-ranks* the retrieved documents again. As the re-ranking model only needs to score a small amount of query-document pairs (e.g., $k_S = 1000$), the use of computationally expensive semantic models becomes feasible. The re-ranking step is deemed very important for tasks that require high performance for small retrieval depths, such as question answering.

As before, we denote the documents retrieved by a sparse retriever for a query q in the first stage by K_S^q. For each retrieved document $d \in K_S^q$, the corresponding re-ranking score $\phi_D(q, d)$ is computed.[5] This score is then used to re-rank the retrieved set to obtain the final ranking. It has been shown that the scores of the sparse retriever ϕ_S can be beneficial for re-ranking as well [4, 10]. To that end, an interpolation-based approach can be employed, where the final score of a query-document pair is computed as

$$\phi(q, d) = \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \phi_D(q, d). \tag{2.12}$$

Setting $\alpha = 0$ recovers standard re-ranking. The linear interpolation in Eq. (2.12) is referred to as *convex combination* in the literature [17].

Dense Retrieval

Dual-encoder models (cf. Section 2.1.2) are commonly utilized to perform *dense retrieval* [89]. Compared to sparse retrieval (cf. Section 2.1.3), queries and documents are represented by low-dimensional[6] dense vectors (i.e., vectors that do not contain zeros). A dense index contains pre-computed vector representations $\eta(d)$ for all documents d in the corpus $\mathcal{D}$. To retrieve a set of documents K_D^q for a query q, a k-nearest neighbor (kNN) search is performed to find the documents whose representations are most similar to the query:

$$K_D^q = \underset{1 \leq i \leq |\mathcal{D}|}{k\text{-argmax}}(\zeta(q) \cdot \eta(d_i)) \tag{2.13}$$

In order to make dense retrieval more efficient, *approximate nearest neighbor* (ANN) search is commonly employed [86, 139]. ANN search can be further accelerated using special hardware, such as GPUs [86].

[5] We choose this notation because re-rankers usually compute and utilize (internal) dense representations.

[6] Karpukhin et al. [89] use 768-dimensional representation vectors in the original DPR approach.

Hybrid Retrieval

An approach similar to interpolation-based re-ranking (cf. Section 2.1.3) is *hybrid retrieval* [55, 123]. The key difference is that the re-ranking scores $\phi_D(q, d)$ are not computed for all query-document pairs. Instead, ϕ_D is a dense retrieval model (cf. Section 2.1.3), which retrieves documents d_i and their scores $\phi_D(q, d_i)$ using nearest neighbor search given a query q. A hybrid retriever combines the retrieved sets of a sparse and a dense retriever.[7]

For a query q, two sets of documents, K_S^q and K_D^q, are retrieved using the sparse and dense retriever, respectively. Since K_S^q and K_D^q usually do not contain exactly the same documents, the combination of both sets is not trivial; possible strategies are

1. ranking all documents in $K_S^q \cup K_D^q$, computing [17] or approximating [123] missing scores,
2. ranking documents in $K_S^q \cap K_D^q$ only, discarding the rest, or
3. ranking documents in either K_S^q or K_D^q only, discarding the rest and approximating missing scores.

An example for the last option is the following, where only documents from K_S^q are considered for the final ranking and the rest is discarded. In this case, the score may be computed as

$$\phi_H(q, d) = \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \begin{cases} \phi_D(q, d) & d \in K_D^q \\ \phi_S(q, d) & d \notin K_D^q \end{cases}. \tag{2.14}$$

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

2.1.4 Metrics

In this section, we introduce the ranking metrics we use in the following chapters. The definitions and notations are mostly taken from Lin, Nogueira, and Yates [120].

Let a ranking R be as in Eq. (2.2). *Precision* measures how many of the documents in a retrieved set are relevant for the query q and is computed as

$$\mathrm{P}(R, q) = \frac{\sum_{i=1}^{|R|} \operatorname{rel}\left(q, d_i^R\right)}{|R|}, \tag{2.15}$$

where $\operatorname{rel}(q, d) \in \{0, 1\}$ denotes the binary relevance of the document d w.r.t. q.

[7] Of course, other combinations are possible, for example, two dense retrievers or two sparse retrievers.

Similarly, *recall* is defined as the fraction of all relevant documents (from the complete corpus $\mathcal{D}$) that are in the retrieved set, i.e.,

$$\mathrm{R}(R, q) = \frac{\sum_{i=1}^{|R|} \mathrm{rel}\left(q, d_i^R\right)}{\sum_{d \in \mathcal{D}} \mathrm{rel}\left(q, d\right)}. \tag{2.16}$$

In the retrieve-and-re-rank setting (cf. Section 2.1.3), high recall is deemed important for the initial retrieval stage, as the re-ranking model is only applied to those documents rather than to the complete corpus.

We denote metrics computed at a particular depth using the $@\,k$ notation, where k is referred to as the *cut-off*; in this case, only the k highest ranked documents in R are considered in the computation (as if $|R| = k$). The *average precision* takes the precision values at depths where a relevant document appears in the ranking into account, i.e.,

$$\mathrm{AP}(R, q) = \frac{\sum_{i=1}^{|R|} \mathrm{P@}\,i\left(R, q\right) \cdot \mathrm{rel}\left(q, d_i^R\right)}{\sum_{d \in \mathcal{D}} \mathrm{rel}\left(q, d\right)}. \tag{2.17}$$

The *discounted cumulative gain*, unlike the previous metrics, takes *graded* relevance judgments into account, i.e., $\mathrm{rel}(q, d) \in \mathbb{N}_0$ can be any integer greater than or equal to zero. It is computed as

$$\mathrm{DCG}(R, q) = \sum_{i=1}^{|R|} \frac{2^{\mathrm{rel}\left(q, d_i^R\right)} - 1}{\log_2(i + 1)} \tag{2.18}$$

Normalizing the DCG using an ideal ranking (i.e., the best ranking possible), denoted by IDCG, yields

$$\mathrm{nDCG}(R, q) = \frac{\mathrm{DCG}(R, q)}{\mathrm{IDCG}(R, q)}. \tag{2.19}$$

The nDCG metric is often used to evaluate methods related to web search [120].

Finally, the *reciprocal rank* is computed as

$$\mathrm{RR}(R, q) = \frac{1}{\mathrm{rank}_i}, \tag{2.20}$$

where rank_i corresponds to the highest rank (i.e., the smallest absolute number) of any relevant document in R. Consequently, only the first relevant document in the ranking is taken into account, and any subsequent relevant documents are insignificant. RR is a popular choice for the evaluation of rankers used in question answering pipelines.

2.2 Large Language Models

Large pre-trained language models (LLMs) are at the core of many recent semantic ranking models (cf. Section 2.1.2). This section gives an overview of the architecture of BERT [40] as a representative of LLMs based on Transformers [197].

LLMs are often characterized by their self-attention components, a large number of parameters, and extensive pre-training, all of which enable superior natural language understanding capabilities. While the original pre-training of BERT was limited to masked language modeling and next sentence prediction, several other objectives have been proposed that are specifically aimed at IR [133, 52, 53, 100]. Even though the original Transformer implements an encoder-decoder architecture, BERT is composed only of a sequence of encoders.

2.2.1 Input Representations

We express the input to the model as a sequence of tokens $T = \left(t_1, \dots, t_{|T|}\right)$. Each token t_i is a subword,[8] allowing the model to deal with unknown words; the WordPiece algorithm [215] is used for tokenization. For example, the sentence

```
How does WordPiece handle unknown words?
```

yields the tokens

```
how   does   word   ##piece   handle   unknown   words   ?
```

using the tokenizer corresponding to uncased BERT$_{\text{base}}$ models.

Prior to encoding, the tokens in T are fed through a trainable embedding layer, representing them as H-dimensional vectors. We denote the embedding operation by $E : \mathbb{N} \mapsto \mathbb{R}^H$. In addition to the embedding, each token is assigned a *positional encoding* $P : \mathbb{N} \mapsto \mathbb{R}^H$. It is computed as

$$\begin{aligned} P\left(t_i\right)_{2j} &= \sin\left(i/10000^{2j/H}\right), \\ P\left(t_i\right)_{2j+1} &= \cos\left(i/10000^{2j/H}\right), \end{aligned} \tag{2.21}$$

where j corresponds to the dimension of the resulting positional encoding vector [197]. Without the positional encodings, the order of the tokens in T would be lost. Thus, the final input representation of a token t_i is $E(t_i) + P(t_i)$.

[8] or a special token (cf. Section 2.1.2)

2.2.2 Encoder Layers

Each encoder layer in Transformer models [197] is composed of two sub-layers: multi-head self-attention and a feed-forward layer. The sub-layers implement residual connections and layer normalization: Let X be the input of sub-layer L_{sub}, then its output is computed as

$$\mathrm{LayerNorm}(X + L_{\text{sub}}(X)). \tag{2.22}$$

We briefly describe the two sub-layers below.

Self-Attention

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

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.23}$$

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

$$\begin{aligned} \mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [h_1, \ldots, h_A]\,\mathbf{W}^O, \\ \text{where} \quad h_i &= \mathrm{Attn}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right). \end{aligned} \tag{2.24}$$

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

As Transformer encoders compute *self-attention*, the three inputs $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ originate from the same place, i.e., they are projections of the output of the previous encoder layer. The inputs to the first encoder layer are the original token representations.

Feed-Forward

The feed-forward component of encoders is a simple fully-connected layer. The output is computed as

$$\mathrm{FFN}(X) = \mathrm{ReLU}\left(X\mathbf{W}_1 + b_1\right)\mathbf{W}_2 + b_2, \tag{2.25}$$

where $\mathbf{W}_i$ and b_i are trainable parameters.

3

Forward Indexes for Efficient Neural Ranking

Neural rankers are typically based on large pre-trained language models, the most popular example being BERT [40]. Due to their architectural inductive bias (like self-attention units) and complexity, these models are able to capture the semantics of documents very well, mitigating the limitations of lexical retrievers. However, their capabilities come at a price, as the models commonly used often have upwards of hundreds of millions of parameters. This makes training and even inference without specialized hardware infeasible, and it is impossible to rank all documents in a large corpus in reasonable time. Furthermore, the resources required to run these models produce a considerable amount of emissions, creating a negative impact on the environment [175].

There are two predominant approaches to deal with the inefficiency of neural ranking models. The first one, referred to as *retrieve-and-re-rank* (cf. Section 2.1.3), uses an efficient lexical retriever to obtain a candidate set of documents for the given query. The idea is to maximize the recall, i.e., capture most of the relevant documents, in the first stage. Afterwards, the second stage employs a complex neural ranker, which *re-ranks* the documents in the candidate set, in order to promote the relevant documents to higher ranks. However, the retrieve-and-re-rank approach typically employs cross-attention re-rankers, which are expensive to compute, even for a small set of candidate documents. This limits the first-stage retrieval depth, as low latency is essential for many applications (e.g., search engines).

The second approach skips the lexical retrieval step entirely and uses neural models for retrieval. The *dual-encoder* architecture employs a *query encoder* and a *document encoder*, both of which are neural models which map their string inputs to dense representations in a common vector space. Retrieval is then performed as a k-nearest-neighbor (kNN) search operation to find the documents whose representations are most similar to the query. This is referred to as *dense retrieval* (cf. Section 2.1.3). Representing queries and documents independently means that most of the computationally expensive processing happens during the indexing stage, where document representations are pre-computed. However, dense retrieval is still slower than lexical retrieval and benefits from GPU acceleration, because the query needs to be encoded during the query-processing phase. Furthermore, we find that dense retrievers generally have lower recall than term-matching-based models at higher retrieval depths.

In this chapter, we argue that neither of the two approaches is ideal. Instead, our first key idea is to explore the utility of dual-encoders in the re-ranking phase instead of the retrieval phase. Using dual-encoders in the re-ranking phase allows for a drastic reduction in query processing times and resource utilization (i.e., GPUs) due to pre-computed document representations. Towards this, we first show that simple interpolation-based re-ranking, that combines the benefits of lexical (computed using sparse retrieval) and semantic (computed using dual-encoders) similarity, can result in competitive and, sometimes, better performance than using cross-attention. We propose a novel index structure called Fast-Forward indexes, which exploits the ability of dual-encoders to pre-compute document representations, in order to substantially improve the runtime efficiency of re-ranking. We empirically establish that dual-encoder models show great performance as re-rankers, even though they do not use cross-attention.

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

Lastly, we focus on the aspects of *index footprint* and *index maintenance.* Since dense indexes store the pre-computed representations of documents in the corpus, they exhibit much higher storage and memory requirements compared to sparse indexes [68]. At the same time, maintaining the index, i.e., adding new documents, requires expensive forward passes of the document encoder. We propose two means of reducing the memory footprint: On the one hand, we propose *sequential coalescing* to compress an index by reducing the number of vectors that need to be stored; on the other hand, we experiment with choosing a smaller number of dimensions, which reduces the size of each vector. Finally, we propose efficient document encoders, which dynamically drop irrelevant tokens prior to indexing using a very simple technique.

Our research questions are as follows:

RQ1.1 How suitable are dual-encoder models for interpolation-based re-ranking in terms of performance and efficiency (Section 3.5.1)?

RQ1.2 Can the re-ranking efficiency be improved by limiting the number of Fast-Forward look-ups (Section 3.5.2)?

RQ1.3 To what extent does query encoder complexity affect re-ranking performance (Section 3.5.3)?

RQ1.4 What is the trade-off between Fast-Forward index size and ranking performance (Section 3.5.4)?

RQ1.5 Can the indexing efficiency be improved by removing irrelevant document tokens (Section 3.5.5)?

We conduct extensive experimentation on existing ranking benchmarks and find that dual-encoder models are very suitable for interpolation-based re-ranking and exhibit highly desirable performance and efficiency trade-offs. We show that, with further optimizations (*early stopping*—cf. Section 3.2.2), re-ranking efficiency can be greatly improved by limiting the number of Fast-Forward look-ups. Additionally, we report a good trade-off between Fast-Forward index size and ranking performance by using our novel *sequential coalescing* algorithm (cf. Section 3.2.1). Our experiments show that we can indeed train extremely lightweight query encoders without adversely affecting ranking performance. Specifically, our most lightweight query encoders are orders of magnitude faster than BERT$_{\text{base}}$ models with little performance degradation. More importantly, we can migrate query processing to CPUs instead of relying on GPUs, improving on the environmental impact. Finally, we show that we can reduce index maintenance costs by around 50% by dynamically removing irrelevant document tokens.

3.1 Related Work

Classical ranking approaches, such as BM25 [169] or the query likelihood model [102], rely on the inverted index, which stores term-level statistics like term frequency, inverse document frequency, and positional information. We refer to this style of methods as *sparse*, since they assume sparse document representations. The recent success of large pre-trained language models (e.g., BERT) shows that *semantic* or contextualized information is essential for many language tasks. In order to incorporate such information in the relevance measurement, Dai and Callan [34, 35] proposed DEEP-CT, which stores contextualized scores for terms in the inverted index for text ranking. SPLADE [49] aims to enrich sparse document representations using a trained contextual Transformer model and sparsity regularization on the term weights. Similarly, DeepImpact [140] enriches the document collection with expansion terms to learn improved term impacts. In this chapter, we employ efficient sparse models for high-recall first-stage retrieval and perform re-ranking using semantic models in a subsequent step.

The ability to accurately determine semantic similarity is essential in order to alleviate the vocabulary mismatch problem [149, 33, 35, 135, 136]. Computing the semantic similarity of a document given a query has been heavily researched in IR using smoothing methods [97], topic models [211], embeddings [148], personalized models [132], etc. In these classical approaches, ranking is performed by interpolating the semantic similarity scores with the lexical matching scores from the first-stage retrieval. More recently, *dense* neural ranking methods, which employ large pre-trained language models, have become increasingly popular. Dense rankers do not explicitly model terms, but rather compute low-dimensional dense vector representations through self-attention mechanisms in order to estimate relevance; this allows them to perform semantic matching. However, the inherent complexity of dense ranking models usually has a negative impact on latency and cost, especially with large corpora. Therefore, besides performance, efficiency has been another major concern in developing neural ranking models.

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

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

3.1.1 Cross-Attention Models

The majority of cross-attention approaches have been dominated by large contextual models [36, 134, 4, 67, 69, 113]. The input to these ranking models is a concatenation of the query and document. This combined input results in higher query processing times, since each document has to be processed in conjugation with the query string. Thereby, cross-attention models usually re-rank a relatively small number of potentially relevant candidates retrieved in the first stage by efficient sparse methods. The expensive re-ranking computation cost is then proportional to the retrieval depth (e.g., 1000 documents).

Another key limitation of using cross-attention models for document ranking is the maximum acceptable number of input tokens for Transformer models, which exhibit quadratic complexity w.r.t. input length. Some strategies address this limitation by document truncation [134] or chunking documents into passages [36, 172]. However, the performance of chunking-based strategies depends on the chunking properties, i.e., passage length or overlap among consecutive passages [173]. Recent proposals include a two-stage approach, where a query-specific summary is generated by selecting relevant parts of the document, followed by re-ranking strategies over the query and summarized document [115, 70, 109, 117]. Due to the efficiency concerns, we do not consider cross-attention methods in this chapter, but focus on dual-encoders instead.

3.1.2 Dual-Encoders

Dual-encoders learn dense vector representations for queries and documents using contextual models [89, 90]. The dense vectors are then indexed in an offline phase [86], where retrieval is akin to performing an approximate nearest neighbor (ANN) search given a vectorized query. This allows dual-encoders to be used for both retrieval and re-ranking. Consequently, there has been a large number of follow-up works that boost the performance of dual-encoder models by improving pre-training [23, 52, 53, 100, 205], optimization [55], and negative sampling [165, 221, 230] techniques or employing distillation approaches [123, 234, 128]. Lindgren et al. [127] propose a *negative cache* that allows for efficient training of dual-encoder models. LED [231] uses a SPLADE model to enrich a dense encoder with lexical information. Lin, Li, and Lin [122] propose Aggretriever, a dual-encoder model which aggregates and exploits all token representations (instead of only the classification token). In this chapter, we use dual-encoders for computing semantic similarity between queries and passages. Some approaches have also proposed architectural modifications to the aggregations between the query and passage embeddings [26, 84, 69]. Nogueira and Lin [156] propose a simple document expansion model. We use dual-encoder models to perform efficient semantic re-ranking in our approach.

Efficiency improvements of dual-encoder-based ranking and retrieval focus mostly on either inference efficiency of the encoders or memory footprint of the indexes. TILDEv1 [237] and TILDEv2 [236] efficiently re-rank documents using a deep query and document likelihood model instead of a query encoder. The SpaDE model [28] employs a *dual document encoder* that has a *term weighting* and *term expansion* component; it improves inference efficiency by using a vastly simplified query representation. Li et al. [116] employ *dynamic lexical routing* in order to reduce the number of dot products in the late interaction step. Cohen et al. [29] use auto-encoders to compress document representations into fewer dimensions in order to reduce the overall size. Dong, Goldstein, and Yang [43] propose an approach to split documents into variable-length segments and dynamically merge them based on similarity, such that each document has the same number of segments prior to indexing. Hofstätter et al. [71] introduce ColBERTer, an extension of ColBERT [90], which removes irrelevant word representations in order to reduce the number of stored vectors. In a similar fashion, Lassance et al. [101] propose a *learned token pruning* approach, which is also used to reduce the size of ColBERT indexes by dropping tokens that are deemed irrelevant. Yang, Qiao, and Yang [224] propose a *contextual quantization* approach for pre-computed document representations (such as the ones used by ColBERT) by compressing document-specific representations of terms.

In most of the previous work, dual-encoders are used in a *homogeneous* or *symmetric* fashion, meaning that both the query and document encoder have the same architecture or even share weights (*Siamese* encoders). Jung, Choi, and Rhee [87] show that the characteristics of queries and documents are different and employ *light fine-tuning* in order to adapt each encoder to its specific role. Kim et al. [92] use model distillation for asymmetric dual-encoders, where the query encoder has fewer parameters than the document encoder. Lassance and Clinchant [99] separate the query and document encoder of SPLADE models in order to improve efficiency. In this chapter, we explore the use of light-weight query encoders for more efficient re-ranking.

3.1.3 Hybrid Models

Hybrid models combine sparse and dense retrieval. The most common approach is a simple linear combination of both scores [123]. CLEAR [55] takes the relevance of the lexical retriever into account in the loss function of the dense retriever. COIL [54] performs contextualized exact matching using pre-computed document token representations. COILcr [45] extends this approach by factorizing token representations and approximating them using canonical representations in order to make retrieval more efficient.

Unlike classical methods, where score interpolation is the norm, semantic similarity from neural contextual models (e.g., cross-attention or dual-encoders) is not consistently combined with the matching score. Recently, Wang, Zhuang, and Zuccon [207] showed that the interpolation of BERT-based models and lexical retrieval methods can boost the performance. Furthermore, they analyzed the role of interpolation in BERT-based dense retrieval strategies and found that dense retrieval alone is not enough, but interpolation with BM25 scores is necessary. Similarly, Askari et al. [10] found that even providing the BM25 score as part of the input text improves the re-ranking performance of BERT models.

3.1.4 Inference Efficiency

Several methods have been proposed to improve the inference efficiency of large Transformer-based models, which have quadratic time complexity w.r.t. the input length. PoWER-BERT [59] progressively eliminates word vectors in the subsequent encoder layers in order to reduce the input size. DeeBERT [218] implements an *early-exit* mechanism, which may stop the computation after any Transformer layer based on the entropy of its output distribution. SkipBERT [203] uses a technique where intermediate Transformer layers can be skipped dynamically using pre-computed look-up tables. We use a simple Selective-BERT approach, which dynamically removes irrelevant document tokens in order to make document encoding more efficient.

3.2 Fast-Forward Indexes

Hybrid retrieval, as described in Section 2.1.3, has two distinct disadvantages. First, in order to retrieve K_D^q, an (approximate) nearest neighbor search has to be performed, which is time consuming. Second, some of the query-document scores are expected to be missed, leading to an incomplete interpolation, where the score of one of the retrievers needs to be approximated [123] for a number of query-document pairs.

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

$$\phi_D(q, d) = \zeta(q) \cdot \eta(d), \tag{3.1}$$

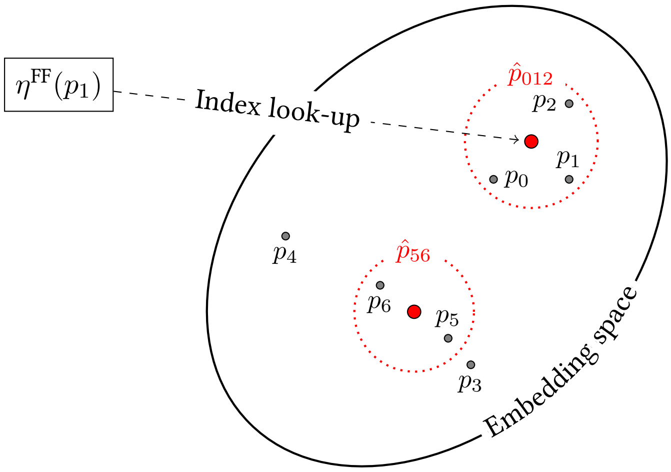


Figure 3.1: Sequential coalescing combines the representations of similar consecutive passages as their average. Note that p_3 and p_5 are not combined, as they are not consecutive passages.

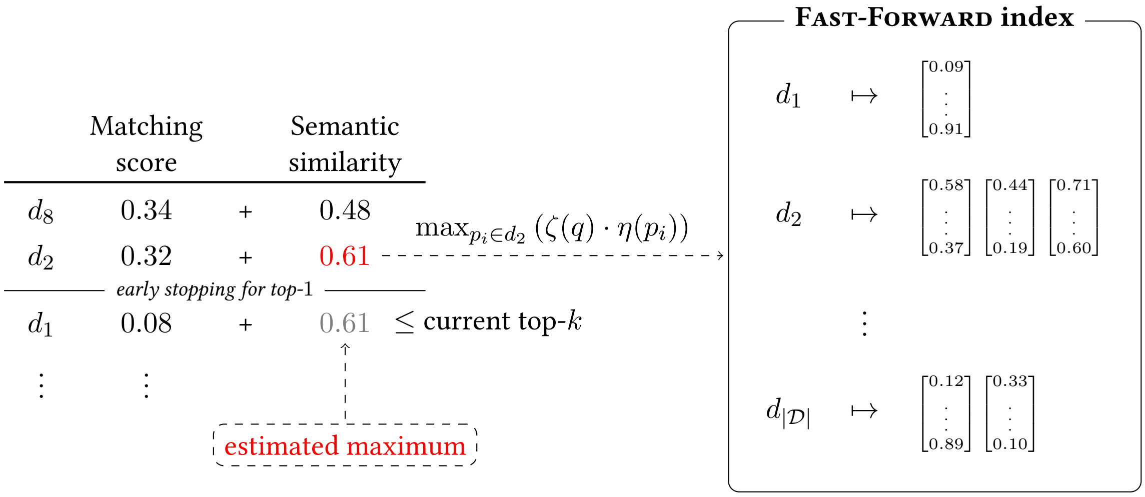

Matching score
Semantic similarity
d_8 0.34 + 0.48
d_2 0.32 + 0.61
$\max_{p_i \in d_2} (\zeta(q) \cdot \eta(p_i))$
early stopping for top-1
d_1 0.08 + 0.61 $\leq$ current top-k
estimated maximum
Fast-Forward index
$d_1 \mapsto$ 0.09 0.91
$d_2 \mapsto$ 0.58 0.37 0.44 0.19 0.71 0.60
$d_{|\mathcal{D}|} \mapsto$ 0.12 0.89 0.33 0.10

Figure 3.2: Early stopping reduces the number of interpolation steps by computing an approximate upper bound for the dense scores. This example depicts the most extreme case, where only the top-1 document is required.

where ζ and η are the query and document encoders, respectively. Examples of such models are ANCE [221] and TCT-ColBERT [123]. Since the query and document representations are independent for dual-encoder models, we can pre-compute the document representations $\eta(d)$ for each document d in the corpus. These document representations are then stored in an efficient hash map, allowing for look-ups in constant time. After the index is created, the score of a query-document pair can be computed as

$$\phi_D^{\text{FF}}(q, d) = \zeta(q) \cdot \eta^{\text{FF}}(d), \tag{3.2}$$

where the superscript "FF" indicates the look-up of a pre-computed document representation in the Fast-Forward index. At retrieval time, only $\zeta(q)$ needs to be computed once for each query. As queries are usually short, this can be done on CPUs. The main benefit of this method is that the number of documents to be re-ranked can be much higher than with cross-attention models; the scoring operation is a simple look-up and dot product computation. We compute the final query-document score using interpolation of sparse retrieval and dense re-ranking scores as in Eq. (2.12).

Note that the use of large Transformer-based query encoders still remains a bottleneck in terms of latency (or, if it is run on GPUs, cost). In Section 3.3, we focus on lightweight encoder models.

3.2.1 Index Compression via Sequential Coalescing

A major disadvantage of dense indexes and dense retrieval in general is the size of the final index. This is caused by two factors: First, in contrast to sparse indexes, the dense representations cannot be stored as efficiently as sparse vectors. Second, the dense encoders are typically Transformer-based, imposing a (soft) limit on their input lengths due to their quadratic time complexity with respect to the inputs. Thus, long documents are split into passages prior to indexing (*maxP* indexes).

As an increase in the index size has a negative effect on efficiency, both for nearest neighbor search and Fast-Forward indexing as used by our approach, we exploit a *sequential coalescing* approach as a way of dynamically combining the representations of consecutive passages within a single document in maxP indexes. The idea is to reduce the number of passage representations in the index for a single document. This is achieved by exploiting the *topical locality* that is inherent to documents [105]. For example, a single document might contain information regarding multiple topics; due to the way human readers naturally ingest information, we expect documents to be authored such that a single topic appears mostly

Algorithm 1: Compression of dense maxP indexes by sequential coalescing

```
Input: list of passage vectors $P$ (original order) of a document, distance threshold $\delta$
Output: coalesced passage vectors $P'$
$P' \leftarrow$ empty list
$\mathcal{A} \leftarrow \emptyset$
foreach $v$ in $P$ do
    if first iteration then
        // do nothing
    else if cosine_distance($v, \overline{\mathcal{A}}$) $\geq \delta$ then
        append $\overline{\mathcal{A}}$ to $P'$
        $\mathcal{A} \leftarrow \emptyset$
    add $v$ to $\mathcal{A}$
    $\overline{\mathcal{A}} \leftarrow$ mean($\mathcal{A}$)
end
append $\overline{\mathcal{A}}$ to $P'$
return $P'$
```

in consecutive passages, rather than spread throughout the whole document. Our approach aims to combine consecutive passage representations that encode similar information. To that end, we employ the cosine distance function and a *threshold* parameter δ that controls the degree of coalescing. Within a single document, we iterate over its passage vectors in their original order and maintain a set $\mathcal{A}$, which contains the representations of the already processed passages, and continuously compute $\overline{\mathcal{A}}$ as the average of all vectors in $\mathcal{A}$. For each new passage vector v, we compute its cosine distance to $\overline{\mathcal{A}}$. If it exceeds the distance threshold δ, the current passages in $\mathcal{A}$ are combined as their average representation $\overline{\mathcal{A}}$. Afterwards, the combined passages are removed from $\mathcal{A}$ and $\overline{\mathcal{A}}$ is recomputed. This approach is illustrated in Algorithm 1. Fig. 3.1 shows an example index after coalescing. To the best of our knowledge, there are no other forward index compression techniques proposed in the literature so far.

3.2.2 Accelerating Interpolation by Stopping Early

As described in Section 2.1.3, by interpolating the scores of sparse and dense retrieval models, we perform implicit re-ranking, where the dense representations are pre-computed and can be looked up in a Fast-Forward index at retrieval time. Furthermore, increasing the sparse retrieval depth k_S, such that $k_S > k$, where k is the final number of documents, improves the performance. A drawback of this is that an increase in the number of retrieved documents also results in an increase in the number of index look-ups.

Common term pruning mechanisms for term-at-a-time retrieval, such as MaxScore [196]

Algorithm 2: Interpolation with early stopping

```
Input: query $q$, sparse retrieval depth $k_S$, cut-off depth $k$, interpolation parameter $\alpha$
Output: approximated top-$k$ scores $Q$
$Q \leftarrow$ priority queue of size $k$
$s_D \leftarrow -\infty$
$s_{min} \leftarrow -\infty$
foreach $d$ in sparse$(q, k_S)$ do
    if $Q$ is full then
        $s_{min} \leftarrow$ remove smallest item from Q
        $s_{best} \leftarrow \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot s_D$
        if $s_{best} \leq s_{min}$ then
            // early stopping
            put $s_{min}$ into $Q$
            break
    // approximate max. dense score
    $s_D \leftarrow$ max$(\phi_D(q, d), s_D)$
    $s \leftarrow \alpha \cdot \phi_S(q, d) + (1 - \alpha) \cdot \phi_D(q, d)$
    put max$(s, s_{min})$ into $Q$
end
return $Q$
```

or WAND [16], accelerate query processing for inverted-index-based retrievers; however, these techniques are not compatible with neural ranking models based on contextual query and document representations. Our use case is more similar to *top-k query evaluation*, with algorithms such as the *threshold algorithm* [44] or probabilistic approximations [192], but these approaches usually require sorted access, which is not available for the dense re-ranking scores in our case.

In this section, we propose an extension to Fast-Forward indexes that allows for *early stopping*, i.e., avoiding a number of unnecessary look-ups, for cases where $k_S > k$ by approximating the maximum possible dense score. The early stopping approach takes advantage of the fact that documents are ordered by their sparse scores $\phi_S(q, d)$. Since the number of retrieved documents, k_S, is finite, there exists an upper limit s_D for the corresponding dense scores such that $\phi_D(q, d) \leq s_D \forall d \in K_S^q$. Since the retrieved documents K_S^q are ordered by their sparse scores, we can simultaneously perform interpolation and re-ranking by iterating over the ordered list of documents: Let d_i be the ith highest ranked document by the sparse retriever. Recall that we compute the final score as

$$\phi(q, d_i) = \alpha \cdot \phi_S(q, d_i) + (1 - \alpha) \cdot \phi_D(q, d_i). \qquad (3.3)$$

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

$$s_{best} = \alpha \cdot \phi_S(q, d_{i-1}) + (1 - \alpha) \cdot s_D. \tag{3.4}$$

In turn, this allows us to stop the interpolation and re-ranking if $s_{best} \leq s_{min}$, where s_{min} denotes the score of the kth document in the current ranking (i.e., the currently lowest ranked document). Intuitively, this means that we stop the computation once the *highest possible* interpolated score $\phi(q, d_i)$ is too low to make a difference. The approach is illustrated in Algorithm 2 and Fig. 3.2. Since the dense scores ϕ_D are usually not normalized, the upper limit s_D is unknown in practice. We thus approximate it by using the highest observed dense score at any given step.

Theoretical Analysis

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

Theorem 3.2.1. *Let s_D, as used in Algorithm 2, be the true maximum of the dense scores. Then the returned scores are the actual top-k scores.*

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

Next, we show that a good approximation of the top-k scores can be achieved by using the sample maximum. To prove our claim, we use the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [143].

Lemma 3.2.2. *Let $X_1, X_2, ..., X_n$ be n real-valued independent and identically distributed random variables with the cumulative distribution function $F(\cdot)$. Let $F_n(\cdot)$ denote the empirical cumulative distributive function, i.e.,*

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \leq x\}}, \quad x \in \mathbb{R}. \tag{3.5}$$

According to the DKW inequality, the following estimate holds:

$$\Pr\left(\sup_{x\in\mathbb{R}}\left(F_n(x)-F(x)\right)>\epsilon\right)\leq e^{-2n\epsilon^2}\forall\epsilon\geq\sqrt{\frac{1}{2n}\ln 2}. \tag{3.6}$$

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

Theorem 3.2.3. *Let $x_1, x_2, ..., x_n$ be a real-valued independent and identically distributed random sample drawn from the distribution of the dense scores with the cumulative distribution function $F(\cdot)$. Let $z = \max\left(x_1, x_2, ..., x_n\right)$. Then, for every $\epsilon > \frac{1}{\sqrt{2n}}\ln 2$, we obtain*

$$\Pr\left(F(z)<1-\epsilon\right)\leq e^{-2n\epsilon^2}. \tag{3.7}$$

Proof. Let $F_n(\cdot)$ denote the empirical cumulative distribution function as above. Specifically, $F_n(x)$ is equal to the fraction of variables less than or equal to x. We then have $F_n(z) = 1$. By Theorem 3.2.2, we infer

$$\Pr\left(F_n(z)-F(z)>\epsilon\right)\leq e^{-2n\epsilon^2}. \tag{3.8}$$

Substituting $F_n(z) = 1$, we obtain Eq. (3.7). □

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

$$\Pr\left(P_D\left(X\leq s_D\right)\geq 1-\epsilon\right)\geq 1-e^{-2n\epsilon^2}, \tag{3.9}$$

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

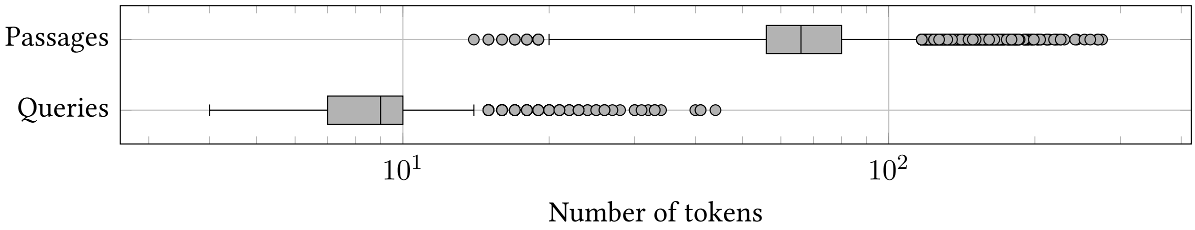


Figure 3.3: The distribution of query and passage lengths in the MS MARCO corpus. The statistics are computed based on the development set queries and the first 10 000 passages from the corpus using a BERT$_{\text{base}}$ tokenizer.

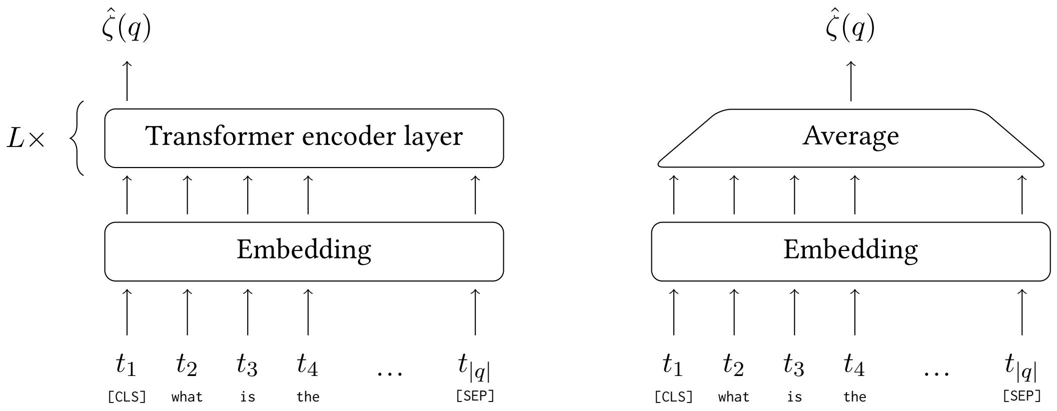
3.3 Efficient Encoders

BERT models are the de facto standard for both query and document encoders [89, 123, 221]. The encoders are often *homogeneous*, meaning that the architectures of both models are identical, or even *Siamese*, i.e., the same encoder weights are used for both queries and documents. Other approaches are *semi-Siamese* models [87], where *light fine-tuning* is used to adapt each encoder to its input characteristics, or TILDEv1 [237] and TILDEv2 [236], which do not require dense query representations. However, the most common choice remains the use of BERT$_{\text{base}}$ for both encoders.

We argue that the homogeneous structure is not ideal for dual-encoder IR models w.r.t. query processing efficiency, since the characteristics of queries and documents differ [87]. Those characteristics w.r.t. the average number of tokens in are illustrated in Fig. 3.3. This section focuses on model architectures for both query and document encoding that aim to improve the overall efficiency of the ranking process.

3.3.1 Lightweight Query Encoders

Query encoders need to be run online during query processing, i.e., the representations cannot be pre-computed. Consequently, query encoding latency is essential for many downstream applications, such as search engines. Our experiments reveal that even encoding a large batch of 256 queries using a BERT$_{\text{base}}$ model on CPU takes more than 3 seconds (cf. Fig. 3.7a), resulting in roughly 12 milliseconds per query (smaller batch sizes or even single queries lead to even slower encoding). Since queries are typically short and concise, we argue that query encoders require lower complexity (e.g., in terms of the number of parameters) than document encoders. Our proposed query encoders are considerably more lightweight than standard BERT$_{\text{base}}$ models and thus more efficient in terms of latency and resources.



(a) Attention-based

(b) Embedding-based

Figure 3.4: The query encoder types used in this chapter. Note that the positional encoding that is added to BERT input tokens has been omitted in this figure.

Attention-based

Attention-based query encoders (such as models based on BERT [40]) use Transformer encoder layers [197] to compute query representations. Each of these layers has two main components—*multi-head attention* and a feed-forward sub-layer—both of which include residual connections and layer normalization operations (see Section 2.2 for a more detailed description).

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

$$\hat{\zeta}_{\text{Attn}}(q) = \text{BERT-CLS}\left(\texttt{[CLS]}, t_1, \dots, t_{|q|}, \texttt{[SEP]}\right), \tag{3.10}$$

where BERT-CLS indicates that the output vector corresponding to the *classification token*, denoted by [CLS], is used. Figure 3.4a shows attention-based query encoders.

The usual choice for query encoders, BERT$_{\text{base}}$, has $L = 12$ layers, $H = 768$ dimensions for hidden representations and $A = 12$ attention heads. In this chapter, we investigate how less complex query encoders impact the re-ranking performance. Specifically, we vary three hyperparameters, namely the number of Transformer layers L, hidden dimensions H and attention heads A. The pre-trained BERT models we use are provided by Turc et al. [195].

Embedding-based

Embedding-based query encoders can be seen as a special case of BERT-based query encoders (cf. Section 3.3.1). Setting $L = 0$, we obtain a model without any Transformer encoder layers; what's left is only the token embedding layer E (cf. Section 2.2.1).

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

$$\hat{\zeta}_{\text{Emb}}(q) = \frac{\sum_{t_i \in q} E(t_i)}{|q|}. \tag{3.11}$$

Embedding-based query encoders are illustrated in Fig. 3.4b.

3.3.2 Selective Document Encoders

Document encoders are not run during query processing time, since document representations are pre-computed and indexed. However, the computation of document representations still requires a substantial amount of time and resources. This is particularly important for applications like web search, where *index maintenance* plays an important role, usually due to large amounts of new documents constantly needing to be added to the index. The effect is further amplified by the maxP approach (cf. Eq. (2.10)), where long documents require more than one encoding step. Since documents tend to be much longer and more complex than queries, lightweight document encoders would likely negatively affect performance, and recent research suggests that larger document encoders lead to better results [152]. However, due to the nature of documents obtained from web pages, we expect a considerable number of document tokens to be irrelevant for the encoding step; examples for this are stop words or redundant (repeated) information. Similar observations have been made in other approaches [71]. Furthermore, recent research [166] has shown that certain aspects, such as the position of tokens, are not essential for large language models to perform well. Our proposed document encoders assign a *relevance score* to each input token and dynamically drop low-scoring tokens before computing self-attention in order to make the document encoding step more efficient.

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

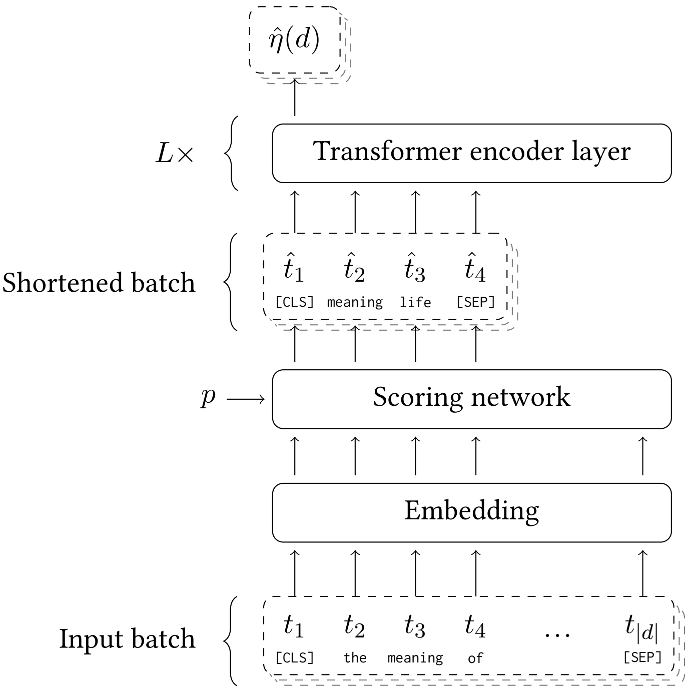


Figure 3.5: The fine-tuning and inference phase of Selective-BERT document encoders. In the given example, the documents in the input batch are dynamically shortened to four tokens each, based on the corresponding relevance scores. Note that the positional encoding that is added to BERT input tokens has been omitted in this figure.

Pre-Training

The **first step** pre-trains the scoring network. θ_Ψ is initialized using the weights of a pre-trained BERT model (e.g., BERT$_{\text{base}}$), and θ_Φ is initialized randomly. The complete model is then trained for a single epoch using the same data as during the unsupervised BERT pre-training step [40]. The scoring network Φ is taken into account by multiplying the embedding of an input token t_i by its corresponding score, i.e.,

$$x_i = E(t_i) \cdot \Phi(t_i) + P(t_i), \tag{3.12}$$

where $E(t_i)$ is the token embedding and $P(t_i)$ is the positional encoding (cf. Section 2.2.1). The resulting representation x_i is then used to compute self-attention in the first encoder layer.

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

$$\ell_1 = \sum_{i=0}^{n} \Phi(t_i). \tag{3.13}$$

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

$$\min_{\theta_\Psi, \theta_\Phi} \left[\mathcal{L}(\theta_\Psi, \theta_\Phi) + \lambda \cdot \ell_1(\theta_\Phi) \right]. \tag{3.14}$$

Fine-Tuning and Inference

The **second step**, referred to as *fine-tuning*, only trains the BERT model Ψ, while the scoring network Φ remains frozen for the remainder of the training process. Furthermore, the weights of the BERT model obtained in the previous step, θ_Ψ, are discarded and replaced by the same pre-trained model as before. The training objective during this stage is identical to that of other dual-encoder models (cf. Eq. (2.11)).

During fine-tuning and inference (i.e., document encoding), we only retain the tokens with the highest scores; we set a ratio $p \in [0; 1]$ of the original input length to retain. As a result, the length of the input batch is shortened by $1 - p$. This is achieved by removing the lowest scoring tokens from the input. Since individual documents within a batch are usually padded, p always corresponds to the longest sequence in the batch. Consequently, padding tokens are always removed first before the scores of the other tokens are taken into account. The process is illustrated in Fig. 3.5.

3.4 Experimental Setup

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

3.4.1 Baselines

We consider the following baselines:

1. **Sparse retrievers** rely on term-based matching between queries and documents. We consider BM25, which uses term-based retrieval signals. DEEP-CT [34], SPLADE [49], and SpaDE [28] use sparse representations, but contextualize terms in some fashion.

2. **Dense retrievers** retrieve documents that are semantically similar to the query in a common embedding space. We consider TCT-ColBERT [123], ANCE [221], and the more recent Aggretriever [122]. All three approaches are based on BERT encoders.

Large documents are split into passages before indexing (maxP). These dense retrievers use exact (brute-force) nearest neighbor search as opposed to approximate nearest neighbor (ANN) search. We evaluate these methods in both the retrieval and re-ranking setting.

3. **Hybrid retrievers** interpolate sparse and dense retriever scores. We consider CLEAR [55], a retrieval model that complements lexical models with semantic matching. Additionally, we consider the hybrid strategy described in Eq. (2.14) as a baseline, using the dense retrievers above.

4. **Re-rankers** operate on the documents retrieved by a sparse retriever (e.g., BM25). Each query-document pair is input into the re-ranker, which outputs a corresponding score. In this chapter, we use a BERT-CLS re-ranker, where the output corresponding to the classification token is used as the score. Note that re-ranking is performed using the full documents (i.e., documents are not split into passages). If an input exceeds 512 tokens, it is truncated. Furthermore, we consider TILDEv2 [236] with TILDE expansion.

3.4.2 Datasets

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

1. The **TREC Deep Learning track** [31] provides test sets and relevance judgments for retrieval and ranking evaluation on the MS MARCO corpora [151]. We use both the passage and document ranking test sets from the years 2019 and 2020 for our experiments. In addition, we use the MS MARCO development sets to determine the optimal values for hyperparameters.

2. The **BEIR benchmark** [191] is a collection of various IR datasets, which are commonly evaluated in a *zero-shot* fashion, i.e., without using any of the data for training the model. We evaluate our models on a subset of the BEIR datasets, including tasks such as passage retrieval, question answering, and fact checking.

3.4.3 Evaluation Details

Our ranking experiments are performed on a single machine using an Intel Xeon Silver 4210 CPU and an NVIDIA Tesla V100 GPU. In our initial experiments (Tables 3.3 and 3.4), we measured the per-query latency by performing each experiment four times and reporting

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

Table 3.1: The pre-trained dense encoders and corresponding indexes we used in our experiments. In each cell, the first line corresponds to a pre-trained encoder (to be obtained from the HuggingFace Hub), and the second line is a pre-built index provided by Pyserini.

the average latency, excluding the first measurement. In subsequent experiments (Table 3.5 and Figs. 3.7a and 3.10a), we adjusted our way of measuring; we perform multiple runs of each experiment, where each run contains multiple latency measurements. We then report the average over all measurements of the fastest run. In Tables 3.3 and 3.4, latency is reported as the sum of scoring,[1] interpolation (cf. Eq. (2.12)), and sorting cost. Any pre-processing or tokenization cost is ignored. Where applicable, dense models use a batch size of 256. The first-stage (sparse) retrieval step is not included, as it is constant for all methods. The Fast-Forward indexes are loaded into the main memory entirely before they are accessed. In Table 3.5, we report end-to-end latency, which includes retrieval, re-ranking, and tokenization cost.

We use the Pyserini [121] toolkit, which provides a number of pre-trained encoders (available on the *HuggingFace Hub*[2]) and corresponding indexes (see Table 3.1), for our retrieval experiments. Dense encoders (ANCE, TCT-ColBERT, and Aggretriever) output 768-dimensional representations. The sparse BM25 retriever is provided by Pyserini as well. We use the pre-built indexes `msmarco-passage` ($k_1 = 0.82$, $b = 0.68$) and `msmarco-doc` ($k_1 = 4.46$, $b = 0.82$). Furthermore, we use Pyserini to run SPLADE with the provided `msmarco-passage-distill-splade-max` index and the pre-trained DistilSPLADE-max model.

We use the MS MARCO development set to determine the interpolation parameter α. We set $\alpha = 0.2$ for TCT-ColBERT, $\alpha = 0.5$ for ANCE and $\alpha = 0.7$ for BERT-CLS (Section 3.5.1). For Aggretriever, we set $\alpha = 0.3$ for BM25 re-ranking and $\alpha = 0.1$ for SPLADE re-ranking. For the dual-encoder models we trained ourselves (Sections 3.5.3 to 3.5.5), the value for α is

[1] This includes operations like encoding queries and documents, obtaining representations from a Fast-Forward index, computing the scores as dot-products, and so on.

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

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

3.4.4 Training Details

Our dual-encoder models are trained using the contrastive loss in Eq. (2.11). For each training instance, we sample 8 hard negative documents using BM25. Additionally, we use in-batch negatives and a batch size of 4, resulting in $|D^-| = 32$ negatives for each query. Each model is trained on four NVIDIA A100 GPUs. We set the learning rate to 1×10^{-5} and use gradient accumulation of 32 batches (this results in an effective batch size of $4 \times 4 \times 32 = 512$). During training, we perform validation on the MS MARCO development set. Our models are trained until the average precision stops improving for five consecutive iterations. We exclusively train on the MS MARCO passage ranking corpus; the resulting models are then evaluated on multiple datasets (i.e., for BEIR, we do zero-shot evaluation). Our Selective-BERT model (cf. Section 3.3.2) uses $\lambda = 10^{-6}$ during pre-training. We implemented our models and training pipeline using PyTorch,[3] PyTorch-Lightning,[4] and Transformers.[5]

Dual-Encoder Architecture

Our dual-encoder rankers consist of a query encoder ζ and a document encoder η (cf. Section 2.1.2):

$$\zeta(q) = ||\mathbf{W}_\zeta \hat{\zeta}(q) + b_\zeta||_2, \tag{3.15}$$

$$\eta(d) = ||\mathbf{W}_\eta \hat{\eta}(d) + b_\eta||_2. \tag{3.16}$$

The models $\hat{\zeta}$ and $\hat{\eta}$ map queries and documents to arbitrary vector representations; examples for these models are pre-trained Transformers or the encoders described in Section 3.3. We include optional trainable linear layers (with corresponding weights $\mathbf{W}_\zeta \in \mathbb{R}^{a \times d_\zeta}$, $\mathbf{W}_\eta \in \mathbb{R}^{a \times d_\eta}$, $b_\zeta \in \mathbb{R}^a$, and $b_\eta \in \mathbb{R}^a$) for heterogeneous encoders, where the dimensions of the representation vectors, d_ζ and d_η, do not match. We further L_2-normalize the representations during training and indexing; we do not normalize the query representations during ranking, as this would only scale the scores, but not change the final ranking.

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

3.5 Results

In this section, we perform experiments to show the effectiveness and efficiency of Fast-Forward indexes.

3.5.1 Dual-Encoders for Interpolation-based Re-ranking

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

Interpolation-based Re-Ranking Performance of Dual-Encoder Models

In Table 3.2, we report the performance of sparse, dense, and hybrid retrievers, re-rankers, and interpolation.

First, we observe that dense retrieval strategies perform better than sparse ones in terms of nDCG, but have poor recall except on TREC-DL-Psg'19. The contextual weights learned by DEEP-CT are better than tf-idf-based retrieval (BM25), but fall short of dense semantic retrieval strategies (TCT-ColBERT and ANCE) with differences upwards of 0.1 in nDCG. However, the overlap among retrieved documents is rather low, reflecting that dense retrieval cannot match query and document terms well.

Second, dual-encoder-based (TCT-ColBERT and ANCE) perform better than contextual (BERT-CLS) re-rankers. In this setup, we first retrieve $k_S = 1000$ documents using a sparse retriever and re-rank them. This approach benefits from high recall in the first stage and promotes the relevant documents to the top of the list through the dense semantic re-ranker. However, re-ranking is typically time-consuming and requires GPU acceleration. The improvements of TCT-ColBERT and ANCE over BERT-CLS (e.g., 0.1 in nDCG) also suggest that dual-encoder-based re-ranking strategies are better than cross-interaction-based methods. However, the difference could also be attributed to the fact that BERT-CLS does not follow the maxP approach (cf. Eq. (2.10)).

Finally, interpolation-based re-ranking, which combines the benefits of sparse and dense scores, significantly outperforms the BERT-CLS re-ranker and dense retrievers. Recall that dense re-rankers operate solely based on the dense scores and discard the sparse BM25 scores of the query-document pairs. The superiority of interpolation-based methods is also supported by evidence from recent studies [23, 26, 55, 54].

	TREC-DL-Doc'19			TREC-DL-Doc'20			TREC-DL-Psg'19		
	AP@1000	R@1000	nDCG@10	AP@1000	R@1000	nDCG@10	AP@1000	R@1000	nDCG@10
Sparse Retrieval									
BM25	0.331	0.697	0.519[abc]	0.404	0.809	0.527[abc]	0.301	0.750	0.506[abc]
DEEP-CT	-	-	0.544	-	-	-	0.422	0.756	0.551
Dense Retrieval									
TCT-ColBERT	0.279	0.576	0.612[a]	0.372	0.728	0.586[ab]	0.391	0.792	0.670
ANCE	0.254	0.510	0.633[a]	0.401	0.681	0.633	0.371	0.755	0.645
Hybrid Retrieval									
CLEAR	-	-	-	-	-	-	0.511	0.812	0.699
Re-Ranking									
TCT-ColBERT	0.370	0.697	0.685	0.414	0.809	0.617	0.423	0.750	0.694
ANCE	0.336	0.697	0.654	0.426	0.809	0.630	0.389	0.750	0.679
BERT-CLS	0.283	0.697	0.520[abc]	0.329	0.809	0.522[abc]	0.353	0.750	0.578[ab]
Interpolation									
[a] TCT-ColBERT	0.406	0.697	0.696	0.469	0.809	0.637	0.438	0.750	0.708
[b] ANCE	0.387	0.697	0.673	0.490	0.809	0.655	0.417	0.750	0.680
[c] BERT-CLS	0.365	0.697	0.612	0.460	0.809	0.626	0.378	0.750	0.617

Table 3.2: Ranking performance. Retrievers use depths $k_S = 1000$ (sparse) and $k_D = 10000$ (dense). Dense retrievers retrieve passages and perform maxP aggregation for documents. Scores for CLEAR and DEEP-CT are taken from the corresponding papers [55, 54]. Superscripts indicate statistically significant improvements using two-paired tests with a sig. level of 95% [158].

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

Table 3.3: Document ranking performance. Latency is reported per query for $k_S = 5000$ as the sum of GPU and CPU time. The coalesced Fast-Forward indexes are compressed to approximately 25% of their original size. Hybrid retrievers use a dense retrieval depth of $k_D = 1000$. Superscripts indicate statistically significant improvements using two-paired tests with a sig. level of 95% [158].

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

Table 3.4: Ranking performance on TREC-DL-Psg'19. Latency is reported per query for $k_S = 5000$ as the sum of GPU and CPU time. Hybrid retrievers use a dense retrieval depth of $k_D = 1000$.

Efficient Re-Ranking at Higher Retrieval Depths

Tables 3.3 and 3.4 show results of re-ranking, hybrid retrieval, and interpolation on document and passage datasets, respectively. The metrics are computed for two sparse retrieval depths, $k_S = 1000$ and $k_S = 5000$.

We observe that additionally taking the sparse component into account in the score computation (as is done by the interpolation and hybrid methods) causes performance to improve with retrieval depth. Specifically, some queries receive a considerable recall boost, capturing more relevant documents with large retrieval depths. Interpolation based on Fast-Forward indexes achieves substantially lower latency compared to other methods. Pre-computing the document representations allows for fast look-ups during retrieval time. As only the query needs to be encoded by the dense model, both retrieval and re-ranking can be performed on the CPU while still offering considerable improvements in query processing time. Note that, for BERT-CLS, the input length is limited, causing documents to be truncated, similarly to the *firstP* approach. As a result, the latency is much lower, but in turn the performance suffers. It is important to note here, that, in principle, Fast-Forward indexes can also be used in combination with firstP models.

The hybrid retrieval strategy, as described in Eq. (2.14), shows good performance. However, as the dense indexes require nearest neighbor search for retrieval, the query processing latency is higher than for interpolation using Fast-Forward indexes.

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

Varying the First-Stage Retrieval Model

We perform additional passage ranking experiments in Table 3.5, where we compare various first-stage retrieval methods in combination with re-rankers. The idea is to show how Fast-Forward indexes perform in combination with modern sparse retrievers and how they compare with other re-rankers. Additionally, these experiments give an idea of the *end-to-end* efficiency, as we report the latency as the sum of retrieval, re-ranking, and tokenization. The Aggretriever model [122] we use in combination with Fast-Forward indexes is a recent single-vector dual-encoder model based on coCondenser [53].

Both SpaDE and SPLADE, unsurprisingly, perform substantially better than BM25, as these models use contextualized learnt representations. This boost in performance comes with a large increase in latency, in terms of both indexing and query processing. However, it becomes evident that re-ranking BM25 results comes very close to these models in terms of performance, and sometimes even surpasses them, even though the overall latency remains lower. At the same time, Fast-Forward indexes manage to improve the performance of SPLADE by re-ranking (although the improvements are not as big). Interestingly, TILDEv2 does not exhibit this behavior, but rather performs worse when a SPLADE first-stage retriever is used. We assume that the reason for this is that the model was not optimized for this scenario.

3.5.2 Early Stopping for more Efficient Re-ranking

We evaluate the utility of the early stopping approach described in Section 3.2.2 on the TREC-DL-Psg'19 dataset. Figure 3.6 shows the average number of look-ups performed in the Fast-Forward index during interpolation w.r.t. the cut-off depth k. We observe that, for $k = 100$, early stopping already leads to a reduction of almost 20% in the number of look-ups. Decreasing k further leads to a significant reduction of look-ups, resulting in improved query processing latency. As lower cut-off depths (i.e., $k < 100$) are typically used in downstream

	Latency	MSM-Psg-Dev		TREC-DL-Psg'19			TREC-DL-Psg'20		
	ms	AP@1000	RR@10	AP@1000	RR@10	nDCG@10	AP@1000	RR@10	nDCG@10
BM25	14	0.196	0.187	0.301	0.702	0.506	0.288	0.655	0.488
↳ TILDEv2	104	0.338	0.342	0.437	0.836	0.680	0.459	0.868	0.679
↳ Aggretriever									
↳ Fast-Forward	150	0.373	0.369	0.465	0.877	0.700	0.486	0.825	0.717
SpaDE ($k = 5$)	-	-	0.355	0.437	-	0.682	0.453	-	0.677
SPLADE	302	0.375	0.368	0.485	0.901	0.728	0.490	0.830	0.711
↳ TILDEv2	374	0.337	0.342	0.412	0.808	0.654	0.433	0.858	0.648
↳ Aggretriever									
↳ Fast-Forward	420	0.383	0.378	0.489	0.899	0.726	0.500	0.856	0.716

Table 3.5: Passage ranking performance using various first-stage retrieval models as well as re-rankers. Aggretriever models are used for interpolation-based re-ranking using Fast-Forward indexes. Re-ranking is done with $k_S = 5000$ passages. SpaDE results are taken from the corresponding paper [28]. For SPLADE, we use the DistilSPLADE-max model. Latency is reported per query on CPU. For retrieval models (BM25 and SPLADE), latency is reported at retrieval depth $k_S = 1000$. For re-ranking (TILDEv2 and Fast-Forward), latency is reported as the sum of retrieval and re-ranking, both at depth $k_S = 5000$.

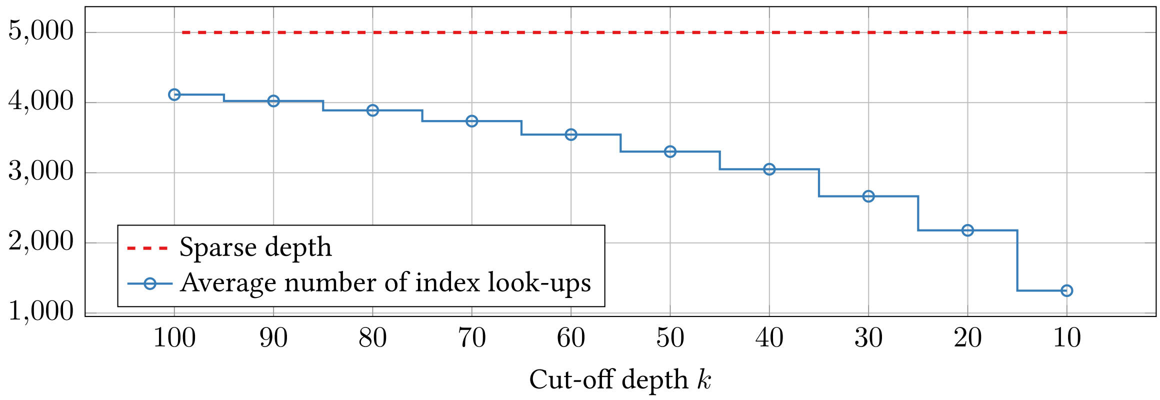


Figure 3.6: The average number of Fast-Forward index look-ups per query for interpolation with early stopping at varying cut-off depths k on TREC-DL-Psg'19 with $k_S = 5000$ using ANCE.

tasks, such as question answering, the early stopping approach for low values of k turns out to be particularly helpful.

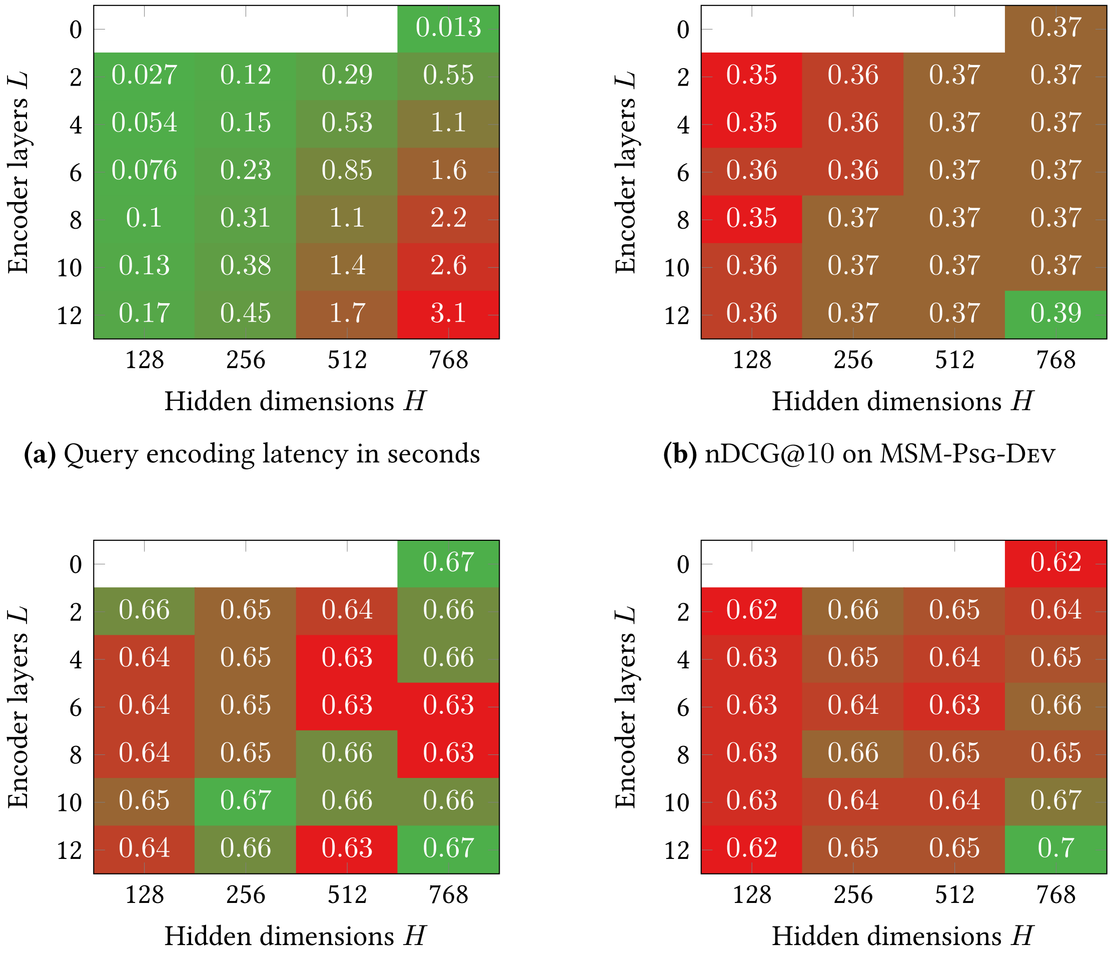
Table 3.4 shows early stopping applied to the passage dataset to retrieve the top-10 passages and compute reciprocal rank. It is evident that, even though the algorithm approximates the maximum dense score (cf. Section 3.2.2), the resulting performance is identical, which means that the approximation was accurate in both cases and did not incur any performance hit. Furthermore, the query processing time is decreased by up to a half compared to standard interpolation. This means that presenting a small number top results (as is common in many downstream tasks) can yield substantial speed-ups. Note that early stopping depends on the value of α, hence the latency varies between TCT-ColBERT and ANCE.

3.5.3 Query Encoder Complexity

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

The Role of Self-Attention

First, we train a large number of dual-encoder models (as described in Section 3.4.4) and successively reduce the complexity of the query encoder. At the same time, we monitor the effects on performance and latency. The query encoders we analyze correspond to the *attention-based query encoders* and the *embedding-based query encoders* described in Section 3.3.1. Since the embedding-based encoders are, technically speaking, a special case of the attention-based ones, we plot the results together in Fig. 3.7. The document encoder we



(a) Query encoding latency in seconds

(b) nDCG@10 on MSM-Psg-Dev

(c) nDCG@10 on TREC-DL-Psg'19

(d) nDCG@10 on TREC-DL-Psg'20

Figure 3.7: Query encoding latency and Fast-Forward ranking performance of dual-encoders with various query encoder models. The sparse retrieval depth is $k_S = 5000$. L and H correspond to the number of Transformer layers and dimensions of the hidden representations, respectively. $L = 0$ corresponds to embedding-based query encoders, which are initialized with pre-trained token embeddings from BERT$_{\text{base}}$, and $L > 0$ corresponds to attention-based query encoders, where the number of attention heads is $A = \frac{H}{64}$. The document encoder is a BERT model with 12 layers and 768-dimensional representations in all cases. Query encoding latency is measured on CPU with a batch size of 256 queries from MSM-Psg-Dev (tokenization cost is excluded, as it is identical for all models).

use is a BERT$_{\text{base}}$ model, which has $L = 12$ layers and $H = 768$ hidden dimensions; it is the same across all experiments. For the query encoder, we start with BERT$_{\text{base}}$ as well and reduce both the number of layers and hidden dimensions. All pre-trained BERT models we use for this experiment are provided by Turc et al. [195]. If the output dimensions of the encoders do not match, we add a single linear layer to the query encoder (cf. Section 3.4.4).

Figure 3.7a illustrates the time each encoder requires to encode a batch of queries on a CPU; as expected, a reduction in either the number of layers or hidden dimensions has a positive impact on encoding latency, and the most lightweight attention-based model ($L = 2$, $H = 128$) is significantly faster than BERT$_{\text{base}}$ (27 milliseconds vs. 3.1 seconds). Furthermore, the complete omission of self-attention in the embedding-based encoder ($L = 0$, $H = 768$) results in even faster encoding (13 milliseconds).

Next, we analyze to what extent the drastic reduction of complexity affects the ranking performance. Figures 3.7b to 3.7d show the corresponding Fast-Forward re-ranking performance on passage development and test sets. It is evident that the absolute difference in performance between the encoders is relatively low; this is especially true on MSM-Psg-Dev and TREC-DL-Psg'19. In fact, the embedding-based query encoder does not always yield worse performance than the attention-based encoders, specifically on TREC-DL-Psg'19. On TREC-DL-Psg'20, the highest absolute difference of 0.05 is the largest among the three datasets.

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

3.5.4 Trade-off Between Index Size and Ranking Performance

This research question investigates how index size influences ranking performance and latency. In detail, we reduce index size in two different ways: First, we apply sequential coalescing (cf. Section 3.2.1) in order to reduce the *number of vector representations* in the index. Second, we train query and encoders to output *lower-dimensional vector representations.* Note that these methods are not mutually exclusive, but rather complementary.

Sequential Coalescing

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

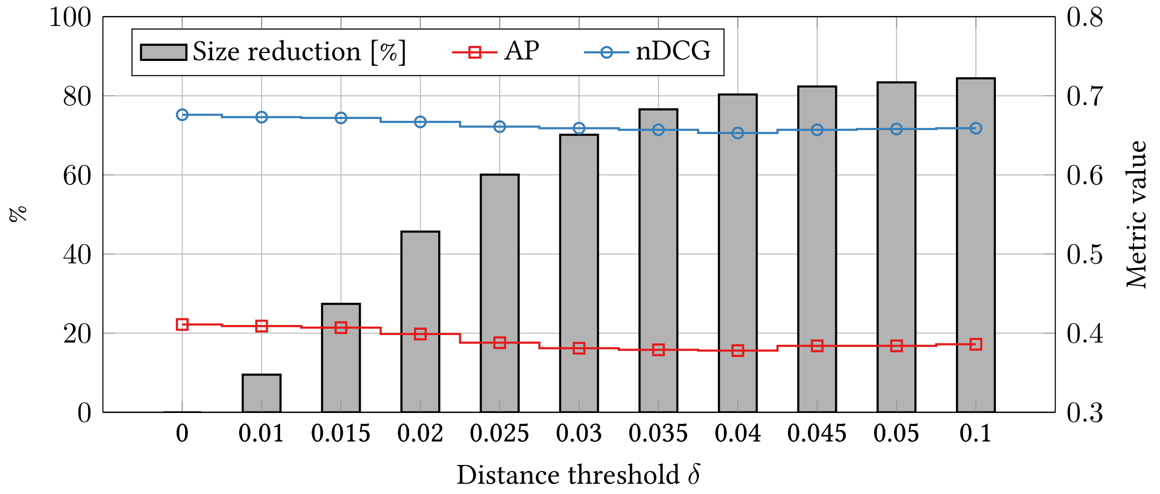


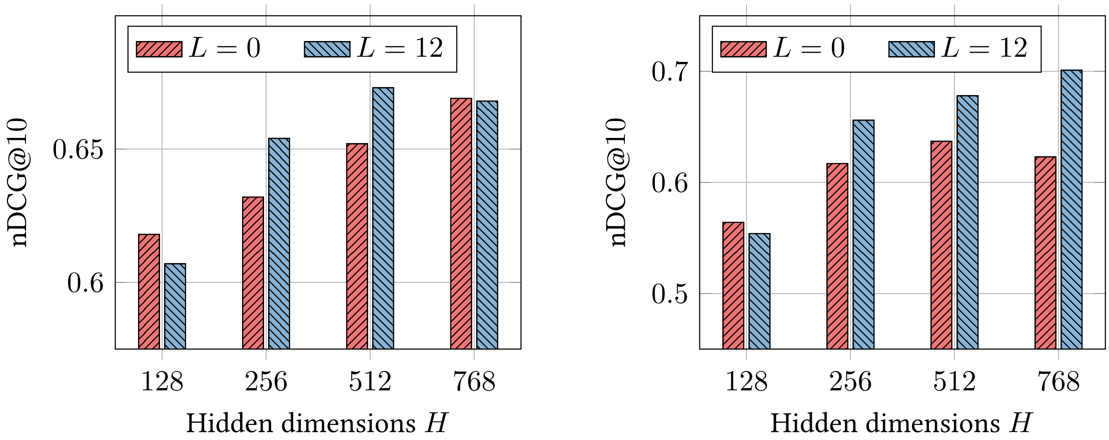
Figure 3.8: Sequential coalescing applied to TREC-DL-Doc'19. The plot shows the index size reduction in terms of the number of passages and the corresponding metric values for Fast-Forward interpolation with TCT-ColBERT.

trated in Fig. 3.8. It is evident that, by combining the passage representations, the number of vectors in the index can be reduced by more than 80% in the most extreme case, where only a single vector per document remains. At the same time, the performance is correlated with the granularity of the representations. However, the drops are relatively small. For example, for $\delta = 0.025$, the index size is reduced by more than half, while the nDCG decreases by roughly 0.015 (3%).

Additionally, Table 3.3 shows the detailed performance of coalesced Fast-Forward indexes on the document datasets. We chose the indexes corresponding to $\delta = 0.035$ (TCT-ColBERT) and $\delta = 0.003$ (ANCE), both of which are compressed to approximately 25% of their original size. This is reflected in the query processing latency, which is reduced by more than half. The overall performance drops to some extent, as expected, however, these drops are not statistically significant in all but one case. The trade-off between latency (index size) and performance can be controlled by varying the threshold δ.

The Effect of Representation Size

In this experiment, we investigate the degree to which the dimension of the query and document representations influences the final ranking performance of the models. The idea is motivated by recent research [152], which suggests that the representation vectors are not the bottleneck of dual-encoder models, but rather the document encoder complexity is. Since the dimensionality of the representations directly influences the index size, it is desirable to keep it as low as possible.



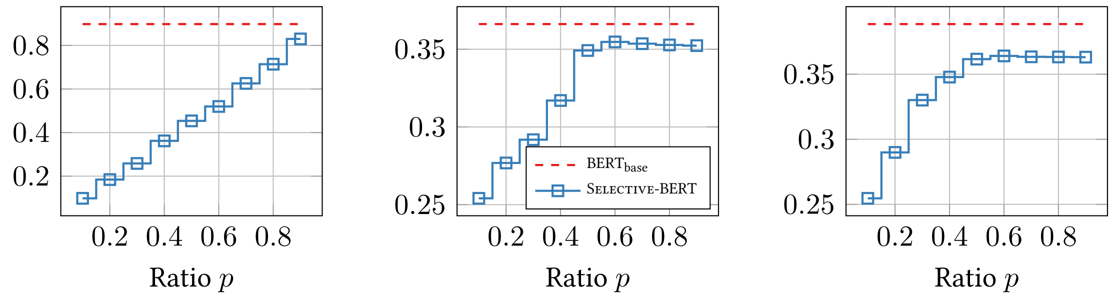
(a) Performance on TREC-DL-Psg'19 **(b)** Performance on TREC-DL-Psg'20

Figure 3.9: Fast-Forward ranking results for $k_S = 5000$ of embedding-based ($L = 0$) and attention-based ($L = 12$) query encoders. The representation dimension H is always the same for both encoders. The document encoders use $L = 12$ layers and $A = \frac{H}{64}$ attention heads in all cases.

In order to analyze the effect, we train a number of dual-encoder models (cf. Eq. (2.11)), where all hyperparameters, except the hidden dimension H and number of attention heads A, are kept the same. We show results for embedding-based ($L = 0$) and attention-based ($L = 12$) query encoders in Fig. 3.9. There is a trade-off between the dimensionality of representations and ranking performance, which is expected; this trade-off is exhibited by both embedding-based and attention-based query encoders. Overall, the results show that the performance reduction is rather small for $H = 512$ and even $H = 256$ (compared to $H = 768$), considering that it goes hand in hand with a reduction in index size of approximately 33% and 67%, respectively.

3.5.5 Efficient Indexing by Removing Irrelevant Document Tokens

In this experiment, we focus on the Selective-BERT document encoders proposed in Section 3.3.2. In order to analyze the index efficiency and ranking performance, we train two dual-encoders (cf. Section 3.4.4) with Selective-BERT document encoders, where $L = 12$ and $H = 768$. The query encoders have $L = 0$ (embedding-based) and $L = 12$ (attention-based), respectively, and $H = 768$. During fine-tuning (cf. Section 3.3.2), we fix the hyperparameter $p = 0.75$, which controls the ratio of tokens to be removed from the documents; afterwards, we create a number of indexes, where we vary p between 0.1 and 0.9, and compute the corresponding indexing time (using GPUs) and ranking performance. The results are plotted in Fig. 3.10.



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

Figure 3.10: Evaluation of FAST-FORWARD indexes created using SELECTIVE-BERT models. The document encoders are BERT$_{\text{base}}$ models with $L = 12$ and $H = 768$. During fine-tuning, we set the parameter $p = 0.75$ (percentage of tokens to keep). We then vary $p \in [0, 1]$ during the indexing stage, resulting in progressively higher indexing efficiency (Fig. 3.10a). The corresponding FAST-FORWARD ranking performance on MSM-PSG-DEV is shown in Fig. 3.10b for an embedding-based query encoder ($L = 0$) and in Fig. 3.10c for an attention-based query encoder ($L = 12$). Document encoding latency is measured on GPU with a batch size of 256 passages from the MS MARCO corpus (tokenization cost is excluded, as it is identical for all models).

The document encoding latency (Fig. 3.10a) increases nearly linearly with the ratio of tokens to keep (p). Even though the BERT model has a quadratic complexity w.r.t. input length, this is expected, as there is a certain amount of overhead introduced by the scoring network and the reconstruction of the batches. More interestingly, the ranking performance (Figs. 3.10b and 3.10c) is mostly unchanged for $p \geq 0.5$ in both cases, however, neither of the models manages to match the performance of their respective baselines (the same configuration with a standard BERT model instead of SELECTIVE-BERT). We hypothesize that the reason for this could be the choice of $p = 0.75$ during the fine-tuning step.

Overall, our results show that up to 50% of document tokens can be removed without much of a performance reduction. Encoding half of the number of tokens results in approximately halving the time required to encode documents. This has a large impact on efficient index maintenance in the context of dynamically increasing document collections. For future work, the SELECTIVE-BERT architecture can be further refined, for example, by introducing improved (contextualized) scoring networks.

3.6 Discussion

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

	TREC-DL-Psg'19			TREC-DL-Doc'19		
	AP@1000	R@1000	nDCG@10	AP@1000	R@1000	nDCG@10
Sparse Retrieval						
BM25	0.301	0.750	0.506	0.331	0.697	0.519
Dense Retrieval						
ANCE	-	-	0.648	-	-	0.628
TCT-ColBERT	-	-	0.670	-	-	-
Our Models						
$L = 0$, $H = 768$	0.198	0.486	0.424	0.100	0.263	0.342
$L = 12$, $H = 768$	0.318	0.691	0.545	0.201	0.457	0.504

Table 3.6: Retrieval results of dual-encoder models using lightweight query encoders and some baselines. For TREC-DL-Doc'19, the dense retrieval depth is set to $k_D = 10000$ and maxP aggregation is applied (cf. Eq. (2.10)). Our model with $L = 0$ uses an embedding-based query-encoder, and the one with $L = 12$ uses an attention-based query encoder. The document encoder is a BERT$_{\text{base}}$ model ($L = 12$, $H = 768$) in both cases.

3.6.1 Efficient Encoders for Dense Retrieval

Our research questions and experiments have focused exclusively on interpolation-based re-ranking using dual-encoders and Fast-Forward indexes. However, the most common application of dual-encoders in the field of IR is the use as dense retrieval models; a natural question that occurs is, whether the encoders proposed in Section 3.3 can be used for more efficient dense retrieval.

In Table 3.6, we present passage and document retrieval results on the MS MARCO corpus. Dense retrievers use a FAISS [86] vector index; no interpolation or re-ranking is performed. It is immediately obvious that our models do not achieve competitive results; on the contrary, the embedding-based encoder yields far worse performance than dense retrievers and even BM25, and even the attention-based encoder fails to improve over sparse retrieval.

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

- We use a simple in-batch negative sampling strategy [89], which has been shown to be less effective than more involved strategies [230, 221, 123, 127].
- The hardware we use for training the models is limiting w.r.t. the batch size and thus the number of negative samples, i.e., we cannot use a batch size greater than 4.
- We perform validation and early stopping based on re-ranking.

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

Table 3.7: Zero-shot ranking results on BEIR datasets (nDCG@10) using embedding-based ($L = 0$) and attention-based ($L = 12$) query encoders. The document encoder is a BERT model with 12 layers and 768-dimensional representations. The sparse retrieval depth is $k_S = 5000$.

Considering the points above, we expect that our dual-encoder models, including ones with lightweight encoders, could also be used in retrieval settings, if the shortcomings of the training setup are addressed, for example, by using more powerful hardware and state-of-the-art training approaches. On the other hand, we argue that the fact that our models perform well in the re-ranking setting (see Section 3.5) shows that it is both easier and more efficient (in terms of time and resources) to train models to be used with Fast-Forward indexes instead of for dense retrieval.

3.6.2 Out-of-Domain Performance

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

In order to ascertain the out-of-domain capabilities of our models, we evaluate them on a number of test sets from the BEIR benchmark. The evaluation happens in a zero-shot fashion, meaning that we use the same models as before and do not re-train them on the respective datasets. The results are shown in Table 3.7. It is apparent that the attention-based query encoder yields better results than the embedding-based one in all cases, but the difference varies across datasets. Since both models were trained on MS MARCO, they perform well on the BEIR version of that dataset, as expected; notable differences in performance are observed on Fever and DBpedia-Entity, however, both models manage to improve the BM25 results. Finally, on Quora, SciFact, and NFCorpus, re-ranking does not lead to a performance improvement, but rather fails to improve or even degrades the results. We assume that the corresponding tasks either require specific in-domain knowledge of the model or would benefit greatly from query-document attention (cross-attention).

3.6.3 Threats to Validity

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

Performance of BERT-CLS

In Tables 3.3 and 3.4, we report the performance of dual-encoder ranking models, along with a cross-attention model (BERT-CLS). We found that BERT-CLS performed notably worse, especially when the sparse retrieval depth k_S is increased. This result is unexpected, especially considering the fact that the cross-attention architecture allows for query-document attention.

In addition to the architecture itself, the models differ in the way they are trained: ANCE and TCT-ColBERT use complex distillation and negative sampling approaches, along with contrastive loss functions (cf. Eq. (2.11)), while BERT-CLS is trained using simple pairwise loss (cf. Eq. (2.8)). It is thus reasonable to assume that the negative sampling approach has a positive impact on the performance. Specifically, the contrastive loss trains the models to identify relevant documents among a very large number of irrelevant documents, while the pairwise loss focuses on re-ranking mostly related documents, which could explain the performance drop for higher retrieval depths.

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

Latency Measurements

As Fast-Forward indexes aim at improving ranking efficiency, we mainly focus on the query processing latency, which is reported in Tables 3.3 to 3.5 and Fig. 3.7. As the experiments in this chapter have been performed over a longer period of time, there have been slight changes with respect to, for example, hardware or implementations. Consequently, the numbers in latency might not be directly comparable **across experiments**. Thus, we made sure to make each experiment self-contained, such that these comparisons are not necessary; rather, our results highlight relative latency improvements **within** each experiment, where all measurements are comparable. In general, one should also keep in mind that latency can be heavily influenced by the way a method is implemented.

Hybrid Retrieval Baselines

In Tables 3.3 and 3.4, we presented, along with the results of our own method, some hybrid retrieval baselines. Table 3.1 shows the corresponding indexes that we used for the dense retrievers. It is important to note that those are *brute-force* indexes, i.e., they perform exact kNN retrieval. It is thus to be expected that the latency of hybrid retrieval can be further reduced by employing approximate dense retrieval instead; this would likely go hand in hand with a reduction in performance though.

3.7 Conclusion

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

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

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

4

Sentence-Level Representations for Passage Ranking

Fine-tuning of large pre-trained language models has been shown to be effective for natural language processing tasks. BERT [40] offers pre-trained deep bidirectional representations, which are conditioned on both left and right context. In combination with the aforementioned task-specific discriminative fine-tuning, it obtained new state-of-the-art results on a broad spectrum of diverse tasks, including passage re-ranking for open domain question answering [155].

There are two limitations of using fine-tuned BERT models for re-ranking passages in QA. First, passages are of variable lengths, which affects the quality of BERT-based representations. Specifically, in the fine-tuning regime of BERT for open domain QA and passage re-ranking, a representation is learnt for the entire passage given a question. While this is desirable for small passages or questions that have short and easy answers, it isn't for instances where the passage answers a question using multiple more complex statements. Second, the passage re-ranking task is unlike other QA tasks, like *factoid QA* or *reading comprehension*, in that the answers are not limited to a word, phrase, or sentence. Potential answers can have varying granularity, and passages are judged by annotators based on the likelihood of containing the relevant answer. Therefore, the applicability of "vanilla" BERT models to answering queries that span multiple sentences or require reasoning across distant sentences within the same passage is limited.

In this chapter, we deal with the above problems by extending the BERT model to explicitly model *sentence representations*. This is realized by distilling the sentence representations from the output of BERT and aggregating the representations of the tokens that form a sentence. Once we have obtained the sentence representations, we apply a *Dynamic Memory Network* [95, 219] to model sentence-wise relations for relevance estimation. We are interested in the following research questions:

RQ2.1 Can the re-ranking performance be improved by aggregating BERT outputs on a sentence level and then reasoning over the resulting sentence representations (Sections 4.4.1 and 4.4.2)?

RQ2.2 Can the training efficiency be improved by performing lightweight reasoning instead of fine-tuning all parameters of BERT (Section 4.4.3)?

We perform experimentation on three diverse open-domain QA datasets and show that the sentence-level representations improve the model's re-ranking performance. We find that explicit sentence modeling using a DMN enables us to reason about the answers that spread across sentences. Additionally, we find that our approach, BERT-DMN, although being an extension of BERT, can be used without expensive fine-tuning of the BERT model, resulting in faster training.

4.1 Related Work

Recent practices in open-domain question answering (QA) can be traced to the Text Retrieval Conferences (TRECs) in the late 1990s. Voorhees [200] defines the task of *textual open-domain question answering* as answering a question using a small text snippet, which is usually an excerpt from a document (e.g., a web page [96]) that is part of a large collection. In the last decade, the focus on open-domain question answering has shifted to the re-ranking stage, where answer identification from candidate documents is performed using learning strategies based on richer and better language understanding models [190, 194, 208, 209, 125]. Our approach also proposes models that improve the re-ranking part of the QA pipeline, though it is different from approaches that perform *end-to-end* question answering, which requires some type of term matching-based retrieval technique to restrict the input text under consideration [25, 210, 94].

Multiple approaches have been proposed to improve re-ranking in open-domain QA. Tan et al. [190] use LSTMs to encode questions and answers and then perform attention- and CNN-based pooling in order to perform question-answer-matching; Tran and Niedereée [194] follow a similar idea, but produce multiple vector representations for each question and answer, which can then focus on different aspects. Lin et al. [125] aim at improving the answer selection process by filtering out noisy, irrelevant paragraphs; afterwards, the answer is selected from the remaining, relevant paragraphs. Some works have used evidence aggregation to re-rank passages based on information from multiple other passages [208] or reinforcement learning to jointly train a *ranking model* to rank the passages and a *reading model* to extract the answer from a passage [209]. Xu et al. [223] use weak supervision to train a BERT-based passage ranking model without any ground-truth labels.

The most recent improvement in the re-ranking stage of open-domain QA comes from BERT models that have been shown to improve natural language understanding. Recent works have used BERT-based ranking models, dealing with efficiency [61] and analyzing the attention mechanism [229]. Peters, Ruder, and Smith [163] compare the performance of BERT with and without fine-tuning on various NLP tasks. MacAvaney et al. [134] combine traditional Ranking models with BERT token representations.

Neural architectures for document ranking can be roughly categorized into *representation-based* models for learning semantic representations of the text [176, 77, 177], *interaction-based* models for learning salient interaction patterns from the local interactions between the query and document [220, 60], or a combination of both [147]. Other works [159, 153, 154] try to capture hierarchical matching patterns based on *n-gram* matches from the local interaction matrix of the query-document. More recent approaches [79, 78, 145] have tried to exploit positional information and context of the query terms. Other approaches include query modeling techniques [42, 228] with a query expansion based language model (QLM) that uses word embeddings.

4.2 BERT-DMN

The usual question answering process consists of multiple stages. Given a query, a simple method (like BM25) is used to rank a number of passages with respect to the query. Next, the top-n of these passages are **re-ranked** using a more expensive model (cf. Section 2.1.3). Finally, the top-k ($k < n$) of the re-ranked passages are used to answer the query. This work deals with the passage re-ranking step.

BERT-based models have achieved high performance in passage re-ranking tasks. We find, however, that these models are limited: First, most variants tend to rely solely on BERT's dedicated classification output, operating under the assumption that its internal capabilities of compressing all query and passage representations into a single output are optimal. Second, BERT models are very large, which results in slow training.

In this chapter, we introduce a re-ranking approach that leverages the representations obtained from BERT and aggregates them using a Dynamic Memory Network. We describe the DMN model and outline how it can be combined with BERT, such that, in addition to the classification output, the query and passage representations are taken into account. Moreover, we investigate how our model can reduce training time by introducing a *lite* version.

4.2.1 Dynamic Memory Networks

In this section, we briefly introduce Dynamic Memory Networks [95, 219], which we use to aggregate BERT outputs. A DMN takes as input a sequence of words $w = (w_1, \ldots, w_T)$, which usually represent multiple sentences, such as a document or a passage, and a question $q = (q_1, \ldots, q_N)$. It is composed of four *modules*:

The **input module** encodes the input words as a sequence of vector representations. The input text is represented by pre-trained word embeddings and fed into a word-level many-to-many GRU. The outputs $h_t^w = \mathrm{GRU}(w_t, h_{t-1}^w)$ are then used as inputs in other modules. If the input consists of a single sentence, each of the GRU outputs is used; however, if the input consists of multiple sentences, only those GRU outputs h_t are used for which t corresponds to an end-of-sentence token (for example, periods or question marks), while the rest is discarded. We denote the final sequence of vectors produced by the input module by $s = (s_1, \ldots, s_M)$.

The **question module** is similar to the input module, as it is used to encode the query (or question) as a fixed-size vector representation. The word embeddings are fed into a many-to-one GRU, which outputs the query representation Q at the end, i.e., $h_t^q = \mathrm{GRU}(q_t, h_{t-1}^q)$ and $Q = h_N^q$.

The **episodic memory module** maintains a number of *episodes*. An episode e^i produces a *memory* m^i by iterating a GRU over the fact representations from the input module, taking the previous memory m^{i-1} into account. For this, the GRU's update gate is replaced by a special attention gate at each time step,

$$\mathrm{AttGRU}^i(x_t, h_{t-1}) = g_t^i \circ h_t' + (1 - g_t^i) \circ h_{t-1}, \tag{4.1}$$

where h_t' is the candidate hidden state for the GRU's internal update at time step t and $\circ$ denotes element-wise multiplication. The *attention gate* g_t^i is a function of the input x_t and the memory and question vectors, encoding their similarities (details can be found in [95]). The initial memory is initialized as $m^0 = Q$. The hidden state of an episode e^i is then computed as $e_t^i = \mathrm{AttGRU}^i(c_t, e_{t-1})$, where $c_t = [s_t, m^{i-1}]$ is a *candidate fact* and $[\cdot,\cdot]$ denotes concatenation. The new memory value is then simply set to the last hidden state

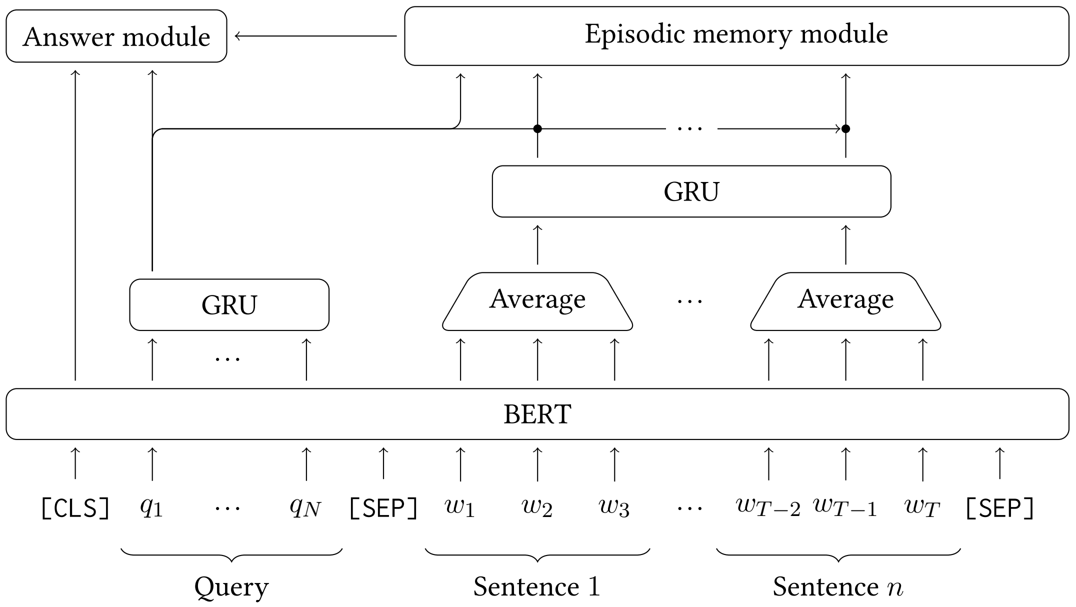


Figure 4.1: The BERT-DMN model architecture. Note that the padding tokens are omitted here.

of the episode, i.e., $m^i = e^i_M$. Finally, the output of the episodic memory module is the last output of a GRU that iterates over all memories m^i.

The **answer module** generates the final output of the model and is therefore highly dependent on the task. In our case, it is a simple feed-forward layer that predicts a score to rank passages given the output of the episodic memory module.

4.2.2 Combining BERT and DMN

Dynamic Memory Networks have proven to be effective in QA tasks such as reading comprehension. In this chapter, we combine a DMN with contextualized representations, specifically, the outputs of BERT, by modifying the input, question, and answer module. The resulting model, BERT-DMN, takes all outputs of BERT into account (including the classification token). It processes the token-level outputs by creating query and sentence representations and reasons over them. In the final step, everything is combined to produce the final query-passage score, which is then used to rank the documents. Figure 4.1 shows the architecture of our approach.

Input and Question Module

Let the query and passage again be denoted by $q = (q_1, \dots, q_N)$ and $w = (w_1, \dots, w_T)$. We first construct the input for BERT as

$$\texttt{[CLS]}, q_1, \dots, q_N, \texttt{[SEP]}, w_1, \dots, w_T, \texttt{[SEP]}. \tag{4.2}$$

Note that q_i and w_i are not necessarily words, as BERT uses subword tokenization. This input format is identical to the usual way BERT is used, where the first input is a classification token, followed by two text inputs, which are separated by separator tokens.

We split the BERT output $o = (o_1, \dots, o_L)$ back into two chunks, where one corresponds to the query and the other one to the passage. The outputs corresponding to the `[SEP]` tokens are discarded. We then use the token representations output by BERT as a replacement for the word embeddings in the DMN. In practice, instead of simply using the vector corresponding to the end-of-sentence token to represent the whole sentence, we take the vectors of all tokens in this sentence and average them.

Answer Module

Since the original DMN model was used for reading comprehension tasks, the answer module consisted of a sequence generation network. For the re-ranking task, we are only interested in predicting a score, therefore we modify the answer module: Let the final memory be a vector $m \in \mathbb{R}^{d \times 1}$ and the BERT output $c \in \mathbb{R}^{b \times 1}$ correspond to the `[CLS]` token. We concatenate these vectors along with the query representation Q and compute the final score a using a feed-forward layer, i.e.,

$$a = \sigma(\mathbf{W}_a[c, Q, m] + b_a), \tag{4.3}$$

where σ is the sigmoid function and $\mathbf{W}_a \in \mathbb{R}^{1 \times (b+2d)}$.

4.3 Experimental Setup

In this section, we describe the datasets we use for our experiments and the baselines. We further outline the training process.

4.3.1 Datasets

We conduct experiments on three diverse passage ranking datasets:

	ANTIQUE	InsuranceQA	TREC-DL-Psg'19
No. of train queries	2406	12 889	808 731
No. of test queries	200	2000	200
Avg. query length	10.55	8.42	6.53
No. of passages	33 642	27 413	8 841 823
Avg. passage length	47.83	103.59	64.63
Avg. no. of passages per query	32.95	500	1000
Avg. no. of relevant passages	9.6	1.66	1.69

Table 4.1: Dataset statistics. InsuranceQA and TREC-DL-Psg'19 have dedicated development sets; for ANTIQUE, we use a small fraction of the training set for validation. Query and passage lengths are measured in words.

1. **ANTIQUE** [65] is a non-factoid question answering benchmark based on the questions and answers of *Yahoo! Webscope L6*. The questions were filtered to remove ones that are too short or duplicate. A resulting sample of question-answer pairs was then judged by crowd workers who assigned one of four relevance labels to each pair. All questions have well-formed correct grammar. For the evaluation, we follow the authors' recommendation and treat the two higher labels as relevant and the lower two labels as irrelevant.

2. **InsuranceQA** [46] is a dataset from the insurance domain released in 2015. For this work, we use the second version, which comes with predefined training, development, and test sets. The development and test sets include, for each question, the relevant answers as well as a pool of $n \in \{100, 500, 1000, 1500\}$ irrelevant candidate answers. For our experiments, we choose $n = 500$. All queries and passages in this dataset consist of grammatically well-formed sentences.

3. **TREC-DL-Psg'19** is the passage ranking dataset from the TREC deep learning track. It uses MS MARCO [151], a collection of large *machine reading comprehension* datasets released by Microsoft in 2016. This dataset was created using real, anonymized queries from the Bing search engine. The authors automatically identified queries that represented questions and extracted passages from the top-10 search results. These passages then manually received relevance labels from human annotators. The result is a very large dataset with over 8 000 000 passages and 1 000 000 queries. However, some queries have no associated relevant passages. Because of the nature of this dataset, queries and passages are not guaranteed to be grammatically or structurally correct or even made of complete sentences.

Table 4.1 outlines some dataset statistics. The evaluation (except for the ANTIQUE test set) follows the telescoping setting [144], where a first round of retrieval has already been performed to select candidate passages that are relevant to the queries, followed by a re-ranking step by our models.

4.3.2 Baselines

Since we are mainly interested in improving the effectiveness and training efficiency of BERT-based models, the most important baseline is a vanilla BERT ranker [155]. The ranking is solely based on the output corresponding to the classification token, which is transformed into a scalar score using a feed-forward classification layer. Additionally, we implement other neural baselines:

1. **QA-LSTM** [190] is based on bidirectional LSTMs and attention. Both query and document are encoded using a shared bidirectional many-to-many LSTM and a pooling operation (maximum or average pooling) to the LSTM outputs. Attention scores are computed using the hidden LSTM states of the document and the pooled query representation. The resulting vectors are then compared using cosine similarity after applying dropout. We set the batch size to 32 and the number of LSTM hidden units to 256. We feed 300-dimensional pre-trained GloVe [162] embeddings to the shared LSTM and use a dropout rate of 0.5.

2. **K-NRM** [220] is a neural ranking model that works via kernel pooling. Starting from pre-trained word embeddings, it builds a *translation matrix*, where each row contains the cosine similarities of a query word to all document words. Each row is then fed into K kernel functions, and the results are pooled by summation. Finally, a single transformation with $\tanh$ activation is applied to output a score. The model is trained with a pairwise ranking loss and uses RBF kernels. We use 300-dimensional pre-trained GloVe embeddings to build the translation matrix. The hyperparameters are adopted from [220]: We set $K = 11$ and use one kernel for exact matches, i.e., $\mu_0 = 1$ and $\sigma_0 = 10^{-3}$. The remaining kernels are spaced evenly in $[-1, 1]$ with $\mu_1 = 0.9, \mu_2 = 0.7, \dots, \mu_{10} = -0.9$ and $\sigma_1 = \dots = \sigma_{10} = 0.1$. We use the Adam optimizer with a leaning rate of 0.001 and $\epsilon = 10^{-5}$ and a batch size of 16.

3. **DMN** [95, 219] serves (in a slightly modified fashion) as the aggregation part of our model, which transforms sentence-level BERT outputs into a relevance score. We also train this model using pre-trained 300-dimensional word vectors in order to analyze if

and how much BERT representations improve the performance. For these experiments, we use the same DMN hyperparameters as in our experiments with BERT-DMN to make the results more comparable.

4.3.3 Training Efficiency

As previously mentioned, a drawback of BERT-based models is their training inefficiency, as the time required for even a single training epoch can be substantial, albeit a one-time cost. In order to mitigate this, we propose BERT-DMN$_{\text{lite}}$. While the model architecture remains identical, the BERT layer is excluded from backpropagation, such that its weights remain frozen. This reduces the training time in two ways: The time required to complete the first epoch will be slightly lower, as the majority of the weights are excluded from the backward pass; the second and all subsequent epochs can be sped up significantly, as the BERT outputs can be cached and re-used.

4.3.4 Training Details

Our models are implemented using PyTorch. We use a pre-trained, uncased BERT$_{\text{base}}$ model with 12 encoder layers, 12 attention heads, and 768-dimensional vector representations. The training is done as follows: We feed all query-passage pairs through the BERT layer to obtain the token representations. We then compute the average of all vectors for each sentence to obtain the inputs for the GRU, which in turn produces representations that serve as the inputs of the episodic memory module. Similarly, we use another GRU to encode the query as a single vector. In the case of BERT-DMN, the fine-tuning of BERT and training of the DMN happens jointly. For BERT-DMN$_{\text{lite}}$, all weights corresponding to BERT are frozen, i.e., they remain unchanged during the optimization. BERT inputs are truncated if they exceed 512 tokens.

The models are trained using the AdamW optimizer [130] with the learning rate set to 3×10^{-5}, following [155], and a pairwise max-margin loss as in Eq. (2.8) with a margin of $m = 0.2$ and linear warm-up over the first 1000 steps ($10\,000$ on TREC-DL-Psg'19). The DMN hyperparameters are set to 4 episodes, 256-dimensional hidden representations, and a dropout rate of 0.1. Dropout is applied at the DMN input, over the attention gates, and before the output layer. We use a batch size of 32 throughout our experiments. Validation is performed based on AP on the development set. We use the same fixed random seed and thus identical training data for all experiments.

	ANTIQUE		InsuranceQA		TREC-DL-Psg'19	
	AP	RR	AP	RR	AP	RR
QA-LSTM	0.488	0.619	0.185	0.231	0.193	0.519
K-NRM	0.511	0.654	0.176	0.215	0.237	0.567
DMN	0.491	0.613	0.092	0.118	0.136	0.274
BERT$_{\text{lite}}$	0.593	0.774	0.259	0.314	0.327	0.739
BERT-DMN$_{\text{lite}}$	0.675	0.851	0.374	0.449	0.418	0.859
BERT	0.697	0.849	0.399	0.476	**0.428**	0.831
BERT-DMN	**0.700**	**0.866**	**0.406**	**0.484**	0.408	**0.889**

Table 4.2: Passage re-ranking performance. BERT$_{\text{lite}}$ and BERT-DMN$_{\text{lite}}$ are architecturally identical to BERT and BERT-DMN, respectively, but only the classification layer is trained, while all other weights remain frozen. The baselines use pre-trained GloVe embeddings.

4.4 Results

In this section, we present and discuss our results.

4.4.1 Passage Re-Ranking Performance

Table 4.2 outlines the passage re-ranking performance of our methods and the baselines on three datasets. It is evident that the BERT-based methods vastly outperform the other baselines on all datasets. BERT$_{\text{lite}}$ performs noticeably worse, but still shows improvements over the non-contextual baselines. Finally, BERT-DMN improves the performance of BERT in all but one case. These results yield the following insights:

1. As expected, the contextual token representations obtained from BERT trump non-contextual word embeddings. Even without any fine-tuning, the BERT representations perform well.

2. The contextual sentence representations do in fact hold valuable information. This information is discarded by models which only use the output corresponding to the classification token. End-to-end training further improves the performance.

As a result, the DMN profits vastly from BERT representations (BERT-DMN$_{\text{lite}}$), and the performance improves even more when the model is trained end-to-end (BERT-DMN).

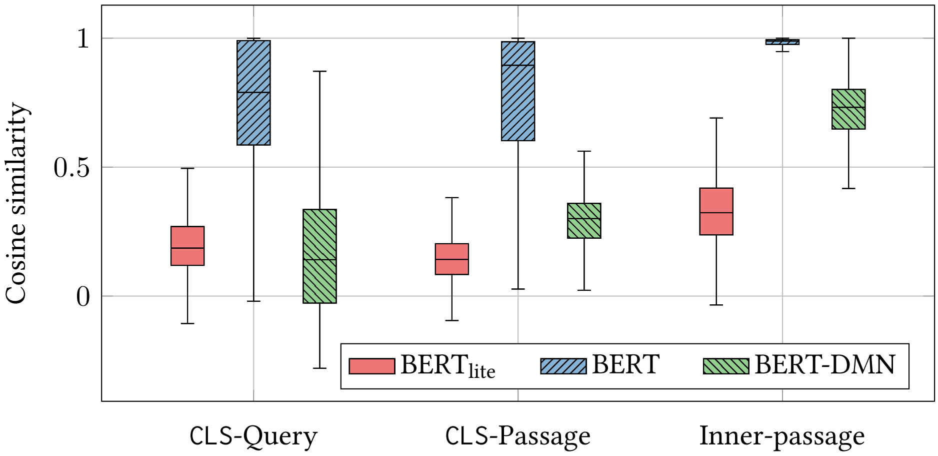


Figure 4.2: The diffusion of information within BERT representations on TREC-DL-Psg'19, illustrated by cosine similarities between classification token, query tokens, and passage tokens. Outliers omitted.

4.4.2 The Effect of Fine-Tuning

In order to analyze the effect of fine-tuning the parameters of BERT, we conduct additional experiments using *lite* versions of BERT and BERT-DMN. The architectures and hyperparameters of these models are unchanged, however, the number of trainable parameters is reduced roughly from 110×10^6 to 3×10^6 (BERT-DMN$_{\text{lite}}$) or 1×10^3 (BERT$_{\text{lite}}$) by freezing the BERT model itself. Table 4.2 shows slight performance drops of BERT-DMN$_{\text{lite}}$ in all but one case. However, comparing it to the fine-tuned vanilla BERT model shows even smaller differences, and, in some cases, the performance increases. Conversely, BERT$_{\text{lite}}$ exhibits a much higher loss of performance over BERT. This indicates that most of the information required for the task is already inherent to the pre-trained BERT model, and fine-tuning its parameters is merely required to direct it towards the desired output (usually, the classification token). In order to confirm this hypothesis, we adopt a method proposed by Goyal et al. [59] to measure the *diffusion of information* within the contextual representations output by BERT: Given a query-passage pair, we use a BERT model to obtain a representation (in our case, a 768-dimensional vector) of each token, corresponding to either the query or the passage. We then compute diffusion of information in three ways:

1. **CLS-Query**: Cosine similarity between the classification token and each query token.

2. **CLS-Passage**: Cosine similarity between the classification token and each passage token.

3. **Inner-passage**: Cosine similarity between each possible pair of two passage tokens.

	ANTIQUE	InsuranceQA
BERT	1.71	1.69
BERT-DMN$_{\text{lite}}$	$2.26 \rightarrow 5.32$	$2.55 \rightarrow 5.67$

Table 4.3: The average number of training batches (size 16) per second (higher is better). For BERT-DMN$_{\text{lite}}$, we report one number for the first epoch and one number for all subsequent epochs.

The results are illustrated in Fig. 4.2 for three BERT models, one without any fine-tuning (BERT$_{\text{lite}}$), one with standard fine-tuning using only the classification output (BERT), and one fine-tuned as part of our approach (BERT-DMN). These measurements were performed on roughly 10% of the TREC-DL-Psg'19 test set (20 000 query-passage pairs). We observe that, without any fine-tuning, the outputs are rather dissimilar; with standard fine-tuning, however, the similarity of all representations vastly increases, especially within the passages. The same trend is exhibited by the model fine-tuned with BERT-DMN, but to a much lesser extend. This shows that discarding all but one output during fine-tuning leads to very high diffusion, in that all output vectors become very similar, and taking all outputs into account during fine-tuning alleviates this issue, allowing for a slight performance gain. It further suggests that BERT-DMN$_{\text{lite}}$ is able to combine the classification output and the sentence representations, performing closely to a fine-tuned BERT model.

4.4.3 Training Efficiency

Since the performances of BERT-DMN$_{\text{lite}}$ and BERT are comparable (cf. Table 4.2), BERT-DMN$_{\text{lite}}$ can be seen as an alternative to the usual fine-tuning of a BERT model. Since the DMN layer has very few parameters compared to BERT (roughly 3×10^6 vs. 100×10^6), the size of the model itself does not change a lot. However, BERT-DMN$_{\text{lite}}$ exhibits noticeable improvements in training efficiency compared to fine-tuning BERT. In order to show this, we measure the number of batches per seconds for both models in Table 4.3. For BERT-DMN$_{\text{lite}}$, the first epoch is already slightly faster, as the majority of the weights are excluded from the backward pass; the second and all subsequent epochs are sped up significantly, as the BERT outputs can be cached re-used for the remainder of the training. The measurements were performed on a single non-shared NVIDIA GTX 1080Ti GPU.

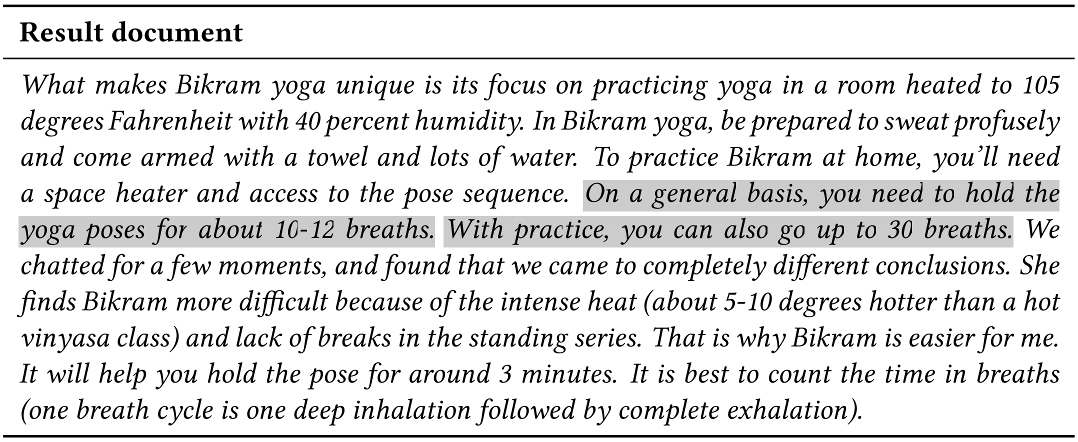
4.5 Conclusion

The exponential growth in the searchable web [76] has resulted in the proliferation of numerous knowledge-intensive tasks [75, 184], of which question answering tasks are prominent [151, 8]. In this chapter, we introduced BERT-DMN and BERT-DMN$_{\text{lite}}$, extensions of BERT, which utilize dynamic memory networks to perform passage re-ranking. We have shown that our model improves the performance of BERT on three datasets. Moreover, BERT-DMN$_{\text{lite}}$ performs well even without a fine-tuned BERT model, reducing the training time while incurring only a small performance hit. Our findings demonstrate that fine-tuning BERT-based models is not always necessary, as nearly the same result can be achieved using sentence-level representations.

5

Extractive Explanations for Interpretable Text Ranking

Information prioritization is an essential and important problem to reduce information overload in a large multitude of web-based tasks like *question-answering* [170], *fact verification* [193], and *conversational search* [8]. Prioritizing information relies on retrieving a small set of highly relevant knowledge units from a large source of world knowledge contained in unstructured text collections with web and textual knowledge bases like Wikipedia that contain documents and articles. The most common way to address information prioritization is to cast it as a ranking problem, i.e., inducing a ranking over all documents in the collection and inspecting only the top-ranked document(s) to satisfy the information need. This is called the *document ranking problem* and is a central task in web search and information retrieval. The objective of the document ranking task is to rank documents relevant to a user-specified query. Consequently, tasks that require access to world knowledge rely on effective document ranking techniques for superior performance. This makes document ranking one of the primitive operations for a large number of knowledge-intensive tasks. Recent advances in document ranking have been dominated by approaches based on tuning large pre-trained contextual models like BERT [36, 134, 4, 111], indicating that better language understanding [3] leads to better document understanding. However, such models are inherently non-interpretable, as they automatically extract latent and complex query-document features from large training sets, leading to opaque decision-making that limits understanding in case of failures or undesirable results. In this chapter, we propose interpretable models for document ranking that have a wide utility in numerous ranking tasks, ranging from web search and question answering over fact-checking to argument and entity



Result document

What makes Bikram yoga unique is its focus on practicing yoga in a room heated to 105 degrees Fahrenheit with 40 percent humidity. In Bikram yoga, be prepared to sweat profusely and come armed with a towel and lots of water. To practice Bikram at home, you'll need a space heater and access to the pose sequence. On a general basis, you need to hold the yoga poses for about 10-12 breaths. With practice, you can also go up to 30 breaths. We chatted for a few moments, and found that we came to completely different conclusions. She finds Bikram more difficult because of the intense heat (about 5-10 degrees hotter than a hot vinyasa class) and lack of breaks in the standing series. That is why Bikram is easier for me. It will help you hold the pose for around 3 minutes. It is best to count the time in breaths (one breath cycle is one deep inhalation followed by complete exhalation).

Figure 5.1: Sentence selection for the query "*how long to hold bow in yoga*", taken from a document in the MS MARCO corpus marked as relevant by human annotators.

retrieval.

Although the interpretability of machine learning models has been popular, there are only few approaches for document ranking, most of which focus on post-hoc interpretability of rankers [182, 180]. However, post-hoc approaches are limited in the sense that their explanations might not accurately reflect the true underlying rationale of the model decisions [171]. Furthermore, the collection of ground-truth data for the evaluation of explanations is often hindered by human bias [98]. Due to this, post-hoc methods are unreliable, and one cannot be sure about the correctness of the explanations.

Unlike post-hoc approaches, we are specifically interested in ranking models that are interpretable by design. We argue that an interpretable ranking model should help us understand which sentences or passages in the document are used for the relevance estimation. In this chapter, we present a document ranking paradigm, where each prediction can be *unambiguously attributed* to a reason or rationale that is both accurate and human-understandable. We define explanations as extractive pieces of text from the input document. An example explanation is shown in Fig. 5.1, where the highlighted sentences serve as an explanation for the relevance of the document. For additional examples, we refer the reader to Tables 5.5 and 5.6.

This chapter proposes a two-stage approach, which we refer to as Select-And-Rank, for modeling long documents that addresses the above limitations. In the **selection phase**, we extract relevant sentences given a query. In the **ranking phase**, we perform the relevance estimation only on the extracted evidence. Our idea is based on the observation that not all sentences in a document are relevant; instead, the document's relevance signals are typically sparse (refer to Fig. 5.2). The selection phase essentially acts as a noise removal mecha-

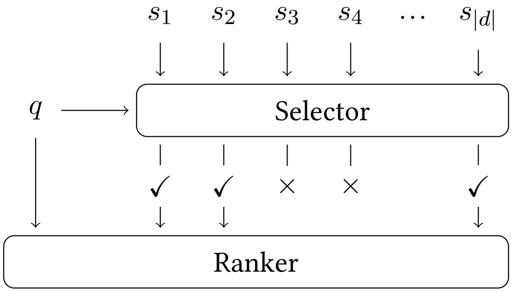


Figure 5.2: The Select-And-Rank paradigm. The document d is split into sentences s_i. The *selector* assigns a score to each sentence with respect to the query q. The scores determine which of the sentences are selected as the input for the *ranker*.

nism, resulting in a succinct, query-based document representation. As an added advantage, our sentence selection allows for choosing a concise query-based document representation as input into size limited models like BERT, in contrast to other heuristic truncation approaches [36].

Within our modular framework, we consider *joint models* that are trained end-to-end with gradient descent. Specifically, we allow the user to regulate the sparsity by setting the number of sentences k to be selected. The selection is akin to sampling from a latent distribution over sentences in a document. We use a parameterized model to output such a distribution and apply the *Gumbel-max* trick. Finally, we use *relaxed subset sampling* to enforce the user-specified sparsity k, i.e., the number of sentences to be selected for the summary or explanation. This allows us to approximate *hard masking*, i.e., the multiplication of the input with a boolean mask in order to remove certain parts, by using *soft masking* (or *continuous masks*), where a similar result is achieved, but the process remains fully differentiable and thus trainable end-to-end.

We conduct extensive empirical evaluation over three document ranking datasets—TREC-DL-Doc'19, Core17, and ClueWeb09. Our intention is not to achieve the best performance in document ranking. Instead, we aim at presenting a ranking model that is interpretable without compromising ranking performance. First, we find that query-specific sparse document representation by sentence selection can improve the task performance over heuristic sentence selection approaches [36]. Second, and more striking, our Select-And-Rank models (with 20 selected sentences) perform on par with and sometimes outperform other document modeling approaches that model the entire document. Furthermore, we show how Select-And-Rank can be used to explain the decisions of BERT rankers that operate only on small parts of the input document.

Additionally, we conduct experiments on twelve diverse datasets provided by the BEIR benchmark [191], including tasks such as passage ranking, fact-checking, and argument retrieval. Our experiments show that Select-And-Rank models also perform well on corpora consisting of shorter documents and provide more interpretable results compared to standard approaches like BERT.

Finally, we highlight the utility of Select-And-Rank to human users through a study and present real-world applications by showing how the extractive explanations can be used to uncover model bias or bugs and illustrating their utility in search engines.

Our experiments aim at answering the following research questions:

RQ3.1 How well do Select-And-Rank models perform in document and passage ranking tasks (Sections 5.4.1 and 5.4.2)?

RQ3.2 How comprehensive are explanations from Select-And-Rank models, i.e., how important are the selected sentences for the model decision (Section 5.4.3)?

RQ3.3 How faithful are Select-And-Rank explanations, and what is their utility to human users (Section 5.4.4)?

RQ3.4 Does sentence selection lead to sparsification of the input documents, resulting in more interpretable ranking decisions (Sections 5.4.5 and 5.4.6)?

RQ3.5 Can Select-And-Rank models be used to explain rankers that focus only on the head of the documents due to limitations, such as BERT (Section 5.4.6)?

5.1 Related Work

We divide the related work into two major categories—*text ranking models* and *interpretability approaches in IR*.

5.1.1 Ranking Models for Text

Classical approaches in information retrieval for ad-hoc document retrieval are probabilistic query likelihood (QL) [102] and term frequency-based models such as BM25 [169] and BM25P [150] models. More recently, neural models have entered the field of IR. Common approaches include semantic representation learning [176, 77, 177], query-document cross-interactions [220, 60, 159, 153, 154] or the exploitation of positional information [79, 78, 145]. Mitra, Diaz, and Craswell [147] employ a combination of the aforementioned approaches.

Nowadays, contextual self-attention-based models, such as BERT [40], achieve state-of-the-art performance in ranking tasks. MacAvaney et al. [134] replace static word embeddings in existing document retrieval models with contextualized token embeddings output by BERT.

Since self-attention models have quadratic time complexity with respect to the input length, work has been done to address this limitation by splitting the input documents into either passages [36, 216, 172] or sentences [4] and subsequently labeling those. Doc-Labeled [36] uses passage-level relevance scores from a fine-tuned BERT model to obtain relevance scores. However, this approach assumes that all passages inherit their relevance from the corresponding document, which might be problematic. BERT-3S [4] works similarly, but on a per-sentence level using a cross-domain transfer model. This leads to substantially slower inference.

Recently, researchers also focused on the efficiency aspect of document and passage retrieval, along with the performance aspect. The major bottleneck of existing language model-based ranking models is the processing time required during the inference phase. Some of the works use dual-encoder based models to alleviate the need of document processing during inference [90, 123, 70, 5]. Zhuang and Zuccon [237] proposed a term-independent likelihood model for passage ranking that relies on both query and document likelihood to rank the documents. This approach pre-computes and stores the likelihood of terms and thus removes the requirement of running deep language models during query processing. In recent times, some studies have focused on the mitigation of positional bias in passage ranking [73] and robustness against misspellings in document retrieval [178]. However, these studies do not focus on the interpretability aspects of the ranking models. The selection model in our Select-And-Rank approach is modular and can be used in combination with other text rankers.

5.1.2 Interpretability of Ranking Models

Interpretability of ranking models focuses on building models that either can be *analyzed for interpretability in a post-hoc fashion* or are *interpretable by design*. However, post-hoc approaches suffer from the limitation that their explanations might not accurately reflect the true rationale underlying the model decisions [171].

Different from classical feature selection, our aim is at selecting features from a document given a query, that is, we want to dynamically select sentences from a document based on the input query. Such instance-wise feature selection has been explored in the machine learning literature [227], however, their applicability to modeling documents is limited.

In NLP, similar models have been studied for ensuring interpretability by design [104, 103]. Li et al. [118] use sentence selection to mimic human reading behavior in order to estimate the relevance of a document to a query. These works mainly differ in how they perform end-to-end training. Training has been done using REINFORCE [104, 118], actor-critic methods [227], pipeline approaches [232], or reparameterization tricks [14]. Lehman et al. [103] use a decoupled rationale generator and predictor. Zhong, Shao, and McKeown [233] use additional human annotations for task supervision. We do not have any explicit training data to train the selector network and rather use the task supervision signal to update its parameters. Finally, [115] pursues an idea similar to ours, however, the authors only use non-trainable selectors and do not consider an end-to-end trainable model.

There exists lots of work on post-hoc analysis of trained neural models on different tasks. Such kinds of analyses use different methods like probing tasks [202], attention weights [11, 27, 142, 32, 222, 225], or state activation [66, 88, 114]. In previous works, researchers tried to learn the attention weights of different tokens to judge their contribution toward a task prediction and mark the tokens that got higher attention scores as rationales or explanations. However, recent studies showed that attention weights are not explanations [83, 212] and models are able to maintain the same prediction accuracy even in the absence of those tokens. Sometimes, tokens that get high attention scores do not correlate well with the human annotated rationales. As recent language models are contextual, it is very difficult to disentangle the importance of token inputs. Finally, there has been recent work on devising decoy datasets to measure the utility of explanation methods for NLP models [80]. Recent approaches also tried to de-bias masked language models with automated bias prompts [62]. A major bottleneck of interpretability studies is the availability of annotated benchmark datasets. In recent times, many interpretability evaluation benchmark datasets have been introduced for neural NLP tasks [41, 206]. In this chapter, our objective is to extend this interpretability aspect toward the document ranking task.

For the ranking task, most of the work has focused on post-hoc interpretability of text rankers [180, 181, 47, 199] and learning-to-rank models [182, 185]. In contrast, our Select-And-Rank models are interpretable by design. The closest to our work is [70], where the authors use cascading rankers after retrieval. However, cascading rankers differ from our approach in the style of optimization and the type of interpretability they provide.

5.2 Select-And-Rank

In this section, we formally define the problem of document ranking (Section 5.2.1). We then give a high-level overview of our Select-And-Rank framework that aims at generating an *extractive sentence-level summary* of the document prior to ranking (cf. Fig. 5.2). Finally, we present our algorithmic contribution that aims at training the *selector* and *ranker* models using gradient-based optimization in a joint manner (Section 5.2.3).

5.2.1 Problem Statement

The retrieve-and-re-rank pipeline (cf. Section 2.1.3) consists of two stages: First, given a query, an inexpensive term-frequency based retriever retrieves a set of documents from the complete, usually very large, collection. Afterward, a more involved, expensive model re-ranks the result of the first-stage retrieval.

Our objective is to learn a parameterized model for document re-ranking. Specifically, given a training set of triples $\left\{q^{(i)}, d^{(i)}, y^{(i)}\right\}_{i=1}^{N}$, where $q^{(i)}$ is a query, $d^{(i)}$ is a document, and $y^{(i)}$ is a relevance label, our goal is to learn a model that predicts relevance scores $\hat{y} \in \mathbb{R}$ for query-document pairs (q, d). We denote the set of documents retrieved in the first stage for the query q by D_q. The resulting predictions are then used to obtain a ranking of all documents $d \in D_q$. Finally, the rankings corresponding to all queries are evaluated using appropriate ranking metrics (cf. Section 2.1.4).

We model each document as a sequence of sentences, i.e., $d = \left(s_1, s_2, \ldots, s_{|d|}\right)$. Our Select-And-Rank approach assumes that only a subset of the constituent sentences actually contribute toward the relevance estimation. Based on this assumption, the model consists of two components: The **selector** Ψ defines a distribution $p\left(s \mid q, d\right)$ over sentences in d given the input query q, encoding the relevance of the sentence given the query. This distribution is used to select an extractive, query-dependent summary $\hat{d} \subseteq d$. The **ranker** Φ is a relatively involved relevance estimation model that generates a relevance label $\hat{y}$ given the query and an extractive document summary $\hat{d}$, thus taking only parts of the document into account.

The selector Ψ is a parameterized model that takes the query and sentences as input and outputs a score or weight w_i for each sentence, representing its relevance to the query, i.e.,

$$\left(w_1, \ldots, w_{|d|}\right) = \Psi\left(q, d\right). \tag{5.1}$$

The logit weights w_i are normalized using the $\mathrm{softmax}$ function, defining a distribution over the sentences as

$$p\left(s_i \mid q, d\right) = \frac{\exp\left(w_i\right)}{\sum_{j=1}^{|d|} \exp\left(w_j\right)}. \tag{5.2}$$

Using this distribution, a document summary $\hat{d} \subseteq d$ is created as a subset of the document's sentences based on the selector's scores, i.e., by dropping some of the lower scoring sentences. The ranker takes as input the query and the document summary to compute the query-document relevance $\hat{y} = \Phi\left(q, \hat{d}\right)$.

Since the selector and ranker are, in principle, independent models, it is possible to train them in one of two ways:

1. Both models are trained separately; the selector is trained to extract a summary from a document with respect to a query, while the ranker is trained on a ranking dataset. The models are then applied consecutively to a query-document pair. We refer to this family of approaches as *pipeline approaches*.

2. The models are trained jointly in an end-to-end fashion, where the gradients are propagated directly from the final outputs back to the selector network. Since this approach includes a non-differentiable selection operation (argmax), it requires approximated differentiable subset sampling.

In this chapter, we analyze and compare the approaches above; furthermore, we implement different selector models and compare them. Section 5.2.2 describes the pipeline approach, and Section 5.2.3 describes the end-to-end approach.

5.2.2 Pipeline Approach

In this section, we apply the Select-And-Rank framework in the aforementioned pipeline setting. Concretely, this means that the selector and ranker are trained independently of each other. For sentence selection, we consider multiple approaches from simple term matching to rather complex language models:

1. **Term matching-based selectors**: We use tf-idf scores between the query q and sentences s_i to determine the best sentences.

2. **Embedding-based selectors**: We use semantic similarity scores between the query q and sentences s_i to determine the best sentences. Both the query and sentence are represented as average over the constituent word embeddings.

3. **Neural non-contextual selectors**: We build a neural network to define a distribution over the sentences s_i.

4. **Contextual selectors**: We use BERT to define a distribution over the sentences s_i.

Term- and embedding-based selectors are non-parameterized. As the other selectors (neural and contextualized models) are parameterized and need to be trained, we follow a transfer learning approach and use the MS MARCO passage re-ranking dataset [151] to train each selector on a passage ranking task. Specifically, the models learn to predict a relevance score given a query and a passage (or sentence). This task in itself is very similar to document summarization, supported by the fact that the passages in this particular dataset were created by splitting documents. We do not consider summarized documents in the training phase of the ranker. During inference, the pipeline approach may be described as follows: The selector is applied to the query and document, outputting a score for each sentence in the document. Along with the query, the k highest scoring sentences then form the input to the ranker, maintaining their original order as in the source document. The ranker outputs the final score $\hat{y}$ that is used to rank the document.

5.2.3 End-to-End Approach

Existing approaches rely on sampling from a stochastic distribution using the REINFORCE algorithm, resulting in a boolean mask over the sentences. An alternative way to achieve end-to-end training instead is by allowing a continuous mask over the sentences. This is akin to using a soft-attention mechanism that is arguably easier to train. However, this approach does not allow for a reduction of the input sequence length, which can be problematic, especially with Transformer-based rankers. Additionally, during inference, one would still need a selection of k sentences given a soft-selection model. This, in particular, is ineffective, given that soft-selection models still rely on all sentences for more effective predictions. We, therefore, propose an approach based on the Gumbel-max trick [137], which enables gradient flow in models where discrete variables must be sampled.

Feature Attribution and Masking

In interpretability, explaining the model output in terms of the input features is called *feature attribution* [9]. Feature attribution (or *saliency*) methods create explanations in terms of input feature importance for individual predictions. In our case of text ranking, a *feature* refers to a subset of the input, such as a sentence or a passage in the document. Feature attributions can be *soft* or *hard*. Soft attributions are scalar values, representing importance, that are assigned to each input feature. The output of an attribution method is typically a vector of the same dimension as the input, with either scalar or boolean values, called a *mask*. A soft mask is an output of soft attributions that can be viewed as a distribution of word-level or sentence-level relevance over the document text. Hard masks are binary and thus have no ambiguity or uncertainty in terms of the presence or absence of a word or sentence in an explanation, which can be preferable for humans [81].

The Gumbel-Max Trick

The *Gumbel-max trick* [137] provides a simple and efficient way to parameterize a discrete distribution and draw samples from it. Let X be a random variable. We wish to parameterize a categorical distribution, such that $P\left(X=i\right) \propto w_i$, where w_i is a weight associated to the ith category. Using the Gumbel-max trick, we can simply draw a sample as

$$X = \underset{i}{\operatorname{argmax}} \left(\log w_i + g_i\right), \tag{5.3}$$

where $g_i = -\log\left(-\log u_i\right)$ is called a *Gumbel random variable* and $u_i \sim \operatorname{Uniform}\left(0,1\right)$. The resulting sample is parameterized by the weights w. In order to completely relax the sampling process and allow for the propagation of gradients (i.e., end-to-end training), the trick is commonly extended, replacing argmax with $\operatorname{softmax}$ (*Gumbel-softmax trick*) [138]. In detail, the Gumbel-softmax estimator gives an approximate one-hot sample y with

$$y_i = \frac{\exp\left(\left(\log w_i + g_i\right)/\tau\right)}{\sum_{j=1}^{k} \exp\left(\left(\log w_j + g_j\right)/\tau\right)} \quad \text{for } i = 1, \ldots, k, \tag{5.4}$$

where τ is a temperature. By using the Gumbel-softmax estimator, one can generate samples $y = \left(y_1, \ldots, y_k\right)$ to approximate the categorical distribution. Furthermore, as the randomness g is independent of w, which is usually defined by a set of parameters, the reparameterization trick can be used to optimize the model's parameters using standard backpropagation algorithms.

Relaxed Subset Sampling

Since we are interested in sampling a subset, i.e., drawing a number of samples (in our case sentences) without replacement, we employ a relaxed subset sampling algorithm proposed in [217] that makes use of the aforementioned Gumbel-max trick. Let a set of items $x_1, \ldots, x_n$ have associated weights w_i and Gumbel variables g_i as above. In order to sample a subset, a *Gumbel-max key*

$$\hat{r}_i = \log w_i + g_i \tag{5.5}$$

is computed for each item. Since $\hat{r}_i$ is a monotonic transformation of w_i (fixing u_i), a relaxed subset sample of the items can be drawn by applying a relaxed top-k procedure directly on

$\hat{r}$. The procedure proposed in [164] defines

$$\alpha_i^1 := \hat{r}_i, \tag{5.6}$$

$$\alpha_i^{j+1} := \alpha_i^j + \log\left(1 - p\left(a_i^j = 1\right)\right), \tag{5.7}$$

where $p\left(a_i^j = 1\right)$ is the expectation of the distribution

$$p\left(a_i^j = 1\right) = \frac{\exp\left(\alpha_i^j / \tau\right)}{\sum_{m=1}^n \exp\left(\alpha_m^j / \tau\right)}, \tag{5.8}$$

and τ is a temperature. Finally, a relaxed k-hot vector is computed as

$$v = \left(v_1, \dots, v_n\right), \tag{5.9}$$

$$v_i = \sum_{j=1}^k p\left(a_i^j = 1\right). \tag{5.10}$$

Training and Inference

In order to train both selector and ranker jointly, we make use of the relaxed subset sampling, as described above. We start by obtaining query and document representations, q^{emb} and d^{emb}, from a shared embedding E as

$$q^{\text{emb}} = E\left(q\right), \tag{5.11}$$

$$d^{\text{emb}} = E\left(d\right). \tag{5.12}$$

The selector then operates on these representations and computes a weight w_i for each sentence s_i, i.e.,

$$\left(w_1, \dots, w_{|d|}\right) = \Psi\left(q^{\text{emb}}, d^{\text{emb}}\right). \tag{5.13}$$

We now draw a relaxed k-hot sample $\hat{w}$ (cf. Section 5.2.3) from the set of sentences using the weights w and a temperature τ as

$$\left(\hat{w}_1, \dots, \hat{w}_{|d|}\right) = \text{SubsetSample}\left(w, k, \tau\right). \tag{5.14}$$

Finally, the document summary $\hat{d}$ is selected as the k highest scoring sentences according to $\hat{w}$. The ranker only operates on the document summary $\hat{d}$ and discards all other sentences. This means that the ranker needs to assemble its new inputs during the training process. The ordering of the sentences is maintained irrespectively of their scores. Since our goal is to train both models jointly, we have to preserve the gradients of the selector (i.e., $\hat{w}$) by combining them with the ranker inputs in a differentiable way. Let $t_1, \dots, t_n$ denote the embedded tokens corresponding to some sentence $s_i \in \hat{d}$. We compute the actual input tokens for the ranker as

$$\hat{t}_j = t_j \odot \hat{w}_i. \tag{5.15}$$

Note that t_j is a vector and $\hat{w}_i$ is a scalar. We use $\odot$ to denote the multiplication of each element in the vector with the scalar. This multiplication changes the input representations, which is undesirable. We mitigate this by making use of the *straight-through* estimator [15]. The idea is to use $\hat{t}_j$ **only** during the backward pass, i.e., when computing gradients. The forward pass simply ignores $\hat{w}_i$ and considers just t_j.

During inference, we do not use relaxed subset sampling. Instead, we simply select the k highest scoring sentences.

Selectors

In this section, we present the selector networks we use in the end-to-end approach. Figure 5.3 illustrates the two selectors.

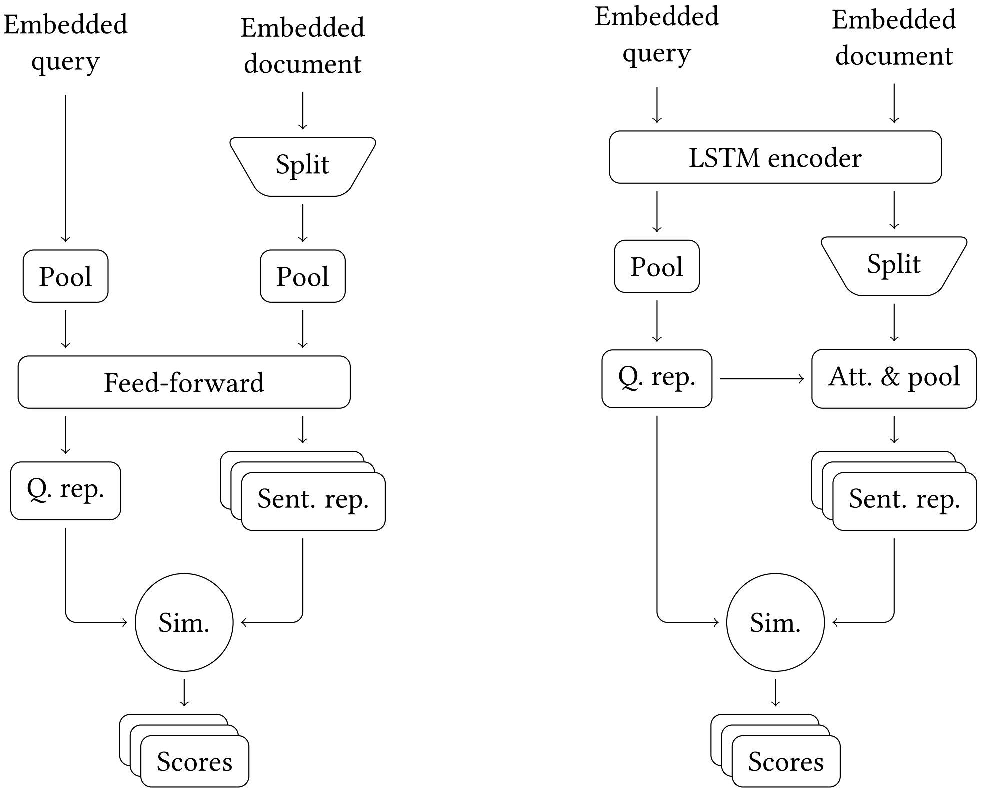
The **linear selector** (Fig. 5.3a) simply represents a sequence as the average of its token embeddings. Query and sentence representation are fed through a single feed-forward layer. The score is computed as the dot product.

The **attention-based LSTM selector** (Fig. 5.3b) is inspired by the QA-LSTM model [190]. Query and document are passed through a shared, bidirectional many-to-many LSTM. On the query side, we obtain the representation $\hat{q}$ by max-pooling over all LSTM outputs. On the document side, we split the LSTM outputs into sequences that correspond to the sentences. Let h^i_j denote the LSTM output corresponding to the jth token of the ith sentence. Prior to max-pooling, we apply a simple token-level attention mechanism as

$$m^i_j = \mathbf{W}_1 h^i_j + \mathbf{W}_2 \hat{q}, \tag{5.16}$$

$$\hat{h}^i_j = h^i_j \exp\left(\mathbf{W}_3 \tanh\left(m^i_j\right)\right), \tag{5.17}$$

where $\mathbf{W_1}$, $\mathbf{W_2}$, and $\mathbf{W_3}$ are trainable parameters. We finally compute the sentence representation $\hat{s}_i$ by max-pooling over all $\hat{h}^i_j$. The score of each sentence is the cosine similarity of its representation to the query representation.



(a) Linear selector

(b) Attention-based LSTM selector

Figure 5.3: The selectors used in the end-to-end approach. The linear selector represents sentences as the average of their embedded tokens and applies a linear layer. The LSTM selector uses a simple attention mechanism.

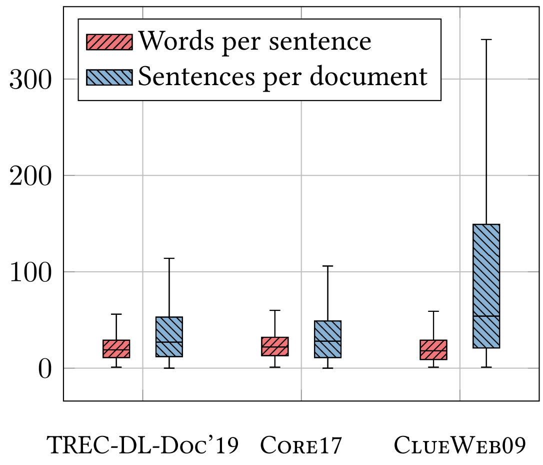


Figure 5.4: The distributions of document and sentence lengths of the TREC datasets. Outliers omitted.

5.2.4 Ranker

Throughout all of our experiments, we use a BERT$_{\text{base}}$ model as the ranker. The model is fine-tuned according to [155]: For a query $q = (q_1, \dots, q_n)$ and a document summary $\hat{d} = \left(\hat{d}_1, \dots, \hat{d}_m\right)$ (produced by the selector), where q_i and $\hat{d}_i$ denote input tokens, the ranker input is

$$\texttt{[CLS]}, q_1, \dots, q_n, \texttt{[SEP]}, \hat{d}_1, \dots, \hat{d}_m, \texttt{[SEP]}. \tag{5.18}$$

We impose a limit of 512 input tokens, i.e., $n + m + 3 \leq 512$. Consequently, long documents are truncated to fit within this limit. We take the output o of BERT, which corresponds to the `[CLS]` input token, and discard the rest. It is fed through dropout and a single feed-forward layer that outputs the final score

$$\hat{y} = \sigma\left(\mathbf{W}o + b\right). \tag{5.19}$$

$\mathbf{W}$ and b denote the trainable parameters of the feed-forward layer, and σ is the sigmoid function.

5.3 Experimental Setup

In this section, we describe our datasets, baselines, and evaluation procedure.

5.3.1 Datasets

First, we consider the following diverse TREC datasets with varying properties:

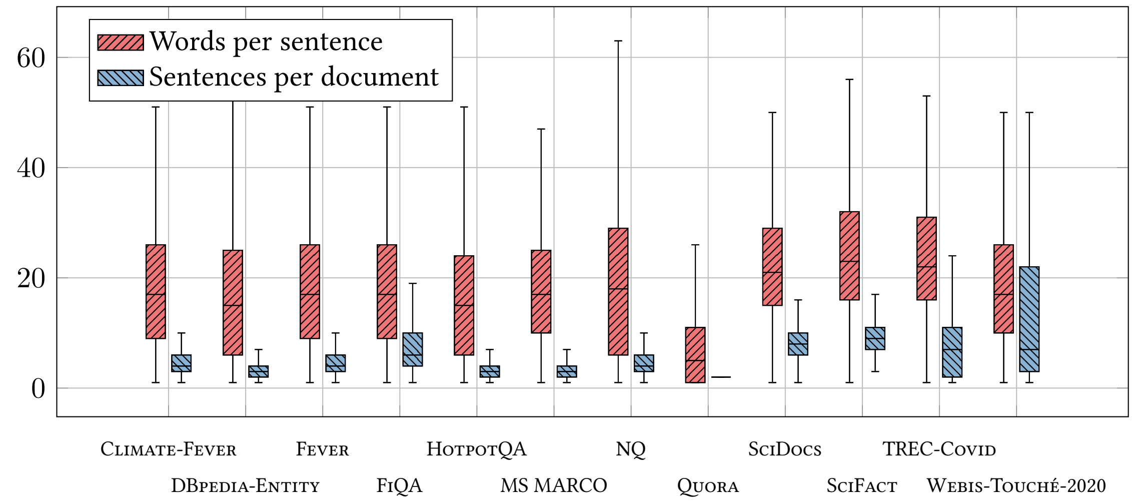


Figure 5.5: The distribution of document and sentence lengths of the datasets from the BEIR benchmark. Outliers omitted.

1. The **TREC-DL-Doc'19** document ranking task uses the MS MARCO document corpus. We use the test set from 2019 for our experiments. Our models use training and validation data from the MS MARCO document ranking task. For each of the 43 queries in the TREC-DL-Doc'19 test set, we re-rank the top-100 retrieved documents.

2. We consider the **ClueWeb09** dataset shared in [36]. The dataset contains 200 queries, distributed uniformly in five folds, and the top-100 documents for each query are retrieved using QL [187].

3. The **Core17** dataset contains 50 queries with sub-topics and descriptions. Queries are accompanied by a collection of 1.8×10^6 documents. We retrieve the top-1000 documents for each query using QL.

Characteristics in terms of the document and sentence lengths of these datasets are illustrated in Fig. 5.4. We observed that the distribution of the number of tokens per sentence is almost identical among all three datasets. In particular, approximately 50% of all sentences have less than 25 tokens, and 90% of all sentences have less than 50 tokens. We use these findings to choose $k = 20$ for our experiments, based on the rough estimation that in this way, all 512 available input tokens of the BERT ranker will be used in most cases, while in the remaining cases, the number of inputs does not exceed the limit by a lot.

Second, we consider a wide variety of additional IR datasets provided by the **BEIR benchmark** [191]. These include classical ranking datasets, such as MS MARCO (passage ranking), fact-checking tasks, such as Fever or SciFact, and others. In contrast to the experiments on the TREC ranking datasets, we perform zero-shot evaluation, i.e., we train a single model on the MS MARCO training set provided by BEIR and use it to evaluate on each test set. As before, we set $k = 20$ for training. An important difference compared to the ranking datasets above is the average length of the documents (or passages). Figure 5.5 shows plots of the distribution of the number of words per sentence and the number of sentences per document for each of the datasets. Overall, the documents are shorter compared to the web retrieval corpora. This means that the limitation of the input length of BERT-based models does not always apply here. We conduct experiments to analyze how sentence selection within those short passages influences both performance and interpretability.

5.3.2 Baselines and Competitors

Since prior studies [4, 134] already established the effectiveness of contextual neural rankers over non-contextual ones, we consider the following rankers based on contextual language models as our baselines:

1. **Doc-Labeled** [36] splits the documents into passages of 150 words with an overlap of 75 words between consecutive passages and considers 30 passages (the first, last and 28 random passages). The relevance label of a query-document pair is then transferred to each of its query-passage pairs. This setup is used to train the models with passage-level annotation, and, finally, passage-level scores are aggregated to come up with document-level scores during inference.

2. **BERT-3S** [4] is a BERT-based transfer model trained on MS MARCO to compute the scores of query-sentence pairs. The query-document level score and the top-3 query-sentences scores are taken into account to compute the final relevance score of that query-document pair.

3. **BERT-CLS** [155] uses a vanilla BERT model to rank the documents, which are truncated to 512 tokens.

Additionally, the first-stage retrieval model, the query likelihood model [102], is also considered as a ranking baseline.

5.3.3 Training Details

We train and validate using consistent and common experimental design. The neural models are trained using a pairwise max-margin loss; we consider triples (q, d^+, d^-) of a query and two documents, where d^+ is more relevant to q than d^-, and compute the loss as in Eq. (2.8)

Training triples are sampled in a balanced way such that each query is represented evenly in the training set. We train the models using the AdamW optimizer [130] with linear warm-up during the first 1000 batches ($10\,000$ on TREC-DL-Doc'19). Validation is performed using average precision (cf. Section 2.1.4) over the validation set to choose the best model. We use a fixed random seed for all experiments.

Hyperparameters

In our experiments we use hyperparameters commonly found in earlier works; the ranker is an uncased 768-dimensional BERT$_{\text{base}}$ model with a maximum sequence length of 512. We use a learning rate of 3×10^{-5}, dropout of 0.1, and a batch size of 32. The selectors (cf. Section 5.2.3) use 256-dimensional hidden representations throughout.

For performance reasons, we restrict the maximum number of query tokens to 50 and the maximum number of document tokens to 5000. Similarly, no more than the first 500 sentences in a single document are considered by the selector. We set the loss margin to $m = 0.2$ and the temperature to $\tau = 1.0$. As described in Section 5.3.1, we set $k = 20$ for training and inference.

5.4 Results

In this section, we analyze the effectiveness and interpretability of our approaches. We first conduct an extensive evaluation of the different selectors, including both pipeline and end-to-end models. Next, we highlight the benefits of our proposed end-to-end modeling scheme (S&R-LIN and S&R-ATT).

5.4.1 Variation of Selectors

In this section, we first briefly describe four different hard selection strategies used by the pipeline models. Next, we compare the pipeline strategies and the two proposed end-to-end variants (cf. Section 5.2.3) of our approach.

The hard selection approaches are described as follows:

1. **PL-BERT**: The similarity or relevance between a query and a sentence is computed using the approach proposed by Akkalyoncu Yilmaz et al. [4]. The model is trained on the MS MARCO passage re-ranking dataset according to Nogueira and Cho [155]. Finally, it is used to infer query-sentence level relevance scores for each query-document pair.

	TREC-DL-Doc'19			Core17			ClueWeb09		
	AP	nDCG@20	RR	AP	nDCG@20	RR	AP	nDCG@20	RR
PL-RND	0.231	0.492[ab]	0.754	0.173	0.345[ab]	0.649	0.138	0.236[ab]	0.495
PL-BERT	0.237	0.501[ab]	0.822	0.200	0.399	0.759	0.169	0.294	0.529
PL-LSTM	0.257	0.558[a]	0.827	0.194	0.399	0.788	0.166	0.289	0.552
PL-BM25	0.264	0.568	0.893	0.196	0.412	0.727	0.171	0.297	0.555
PL-SEM	0.265	0.571	0.920	0.207	0.414	0.768	0.167	0.286	0.534
[a] S&R-LIN	0.269	0.597	0.946	0.203	0.411	0.710	0.174	0.303	0.535
[b] S&R-ATT	0.271	0.590	0.924	0.205	0.403	0.714	0.168	0.292	0.518

Table 5.1: Ranking performance with $k = 20$. PL-RND refers to the selection of k random sentences. Significant improvements (nDCG@20) at a level of 95% are indicated by superscripts.

2. **PL-LSTM**: It is similar to PL-BERT, but uses an LSTM instead of BERT. We limit the input to 1000 words for this configuration, similar to BERT's limit of 512 tokens. The model is trained on the MS MARCO passage re-ranking dataset, and the trained model is used to infer the relevance score of query-sentence pairs.

3. **PL-BM25**: We use a simple BM25-based term matching function to obtain the score between the query and the sentence.[1]

4. **PL-SEM**: Semantic similarity between query and sentence is computed using 300-dimensional GloVe [162] embeddings.

These models apply the selection strategy only during the inference phase, i.e., trained models are used to predict the relevance of pairs of queries and summarized documents. The ranker itself is simply trained without any selection, i.e., documents are truncated to fit. We refer to these strategies as *pipeline* (PL). They can be seen as an implementation of the method proposed in [115].

To analyze the effectiveness of the above-mentioned selection approaches, we also measure the performance of a simple strategy, PL-RND, where we randomly select k sentences from the document. Table 5.1 shows the results for $k = 20$ and highlights the effectiveness of the proposed approaches over the random selection on the TREC datasets. We also tried other values, but $k = 20$ gives consistent performance for all three datasets. This may be attributed to the token limitation of the ranker.

It is interesting to note that our lightweight selection strategies, such as PL-BM25 and PL-SEM, perform better than heavy parameterized neural selection models, such as PL-BERT

[1] https://pypi.org/project/rank-bm25/

	TREC-DL-Doc'19			Core17			ClueWeb09		
	AP	nDCG@20	RR	AP	nDCG@20	RR	AP	nDCG@20	RR
QL	0.237	0.487[ab]	0.785	0.203	0.395	0.686	0.165	0.277	0.487
Doc-Labeled	0.203	0.434[ab]	0.731	0.237	0.437	0.742	0.165	0.284	0.503
BERT-3S	0.245	0.519[ab]	0.799	0.204	0.406	0.694	0.178	0.306	0.544
BERT-CLS	0.260	0.581	0.874	0.196	0.419	0.749	0.178	0.313	0.572
PL-SEM	0.265	0.571	0.920	0.207	0.414	0.768	0.167	0.286	0.534
[a] S&R-LIN	0.269	0.597	0.946	0.203	0.411	0.710	0.174	0.303	0.535
[b] S&R-ATT	0.271	0.590	0.924	0.205	0.403	0.714	0.168	0.292	0.518

Table 5.2: Ranking performance. Select-And-Rank models use $k = 20$. For Doc-Labeled, we report the best strategy (firstP, maxP, avgP for TREC-DL-Doc'19, Core17, and ClueWeb09, respectively). Significant improvements (nDCG@20) at a level of 95% are indicated by superscripts.

	AP	nDCG@10
MatchPyramid	0.232	0.567
Co-PACRR	0.231	0.550
Conv-KNRM	0.241	0.565
TKL-2k	0.264	0.634
S&R-LIN	0.269	0.646
S&R-ATT	0.271	0.639

Table 5.3: Neural baselines on TREC-DL-Doc'19. Select-And-Rank models use $k = 20$. Results are taken from [72]. TKL-2k refers to TKL operating on 2000 tokens.

and PL-LSTM. PL-SEM shows the best or comparable performance for all three datasets. PL-BM25, while slightly worse, also shows promising performance. The compact representation of documents also helps in developing computationally efficient ranking models and reducing noise.

Our end-to-end models, S&R-LIN and S&R-ATT, show improvements over the pipeline models in most cases. Surprisingly, the linear, more lightweight selector often matches or exceeds the performance of the attention-based one.

We also perform statistical pairwise t-tests [51] for nDCG@20 between pipeline approaches and S&R-LIN and S&R-ATT. We do not observe significant improvements for Core17 and ClueWeb09. However, end-to-end models perform significantly better than PL-BERT and PL-LSTM.

5.4.2 Performance of Select-And-Rank

In this section, we compare the performance of our proposed models to state-of-the-art models. We further compare our end-to-end approaches, S&R-LIN and S&R-ATT, to a simple BERT baseline, denoted by BERT-CLS, which uses truncation of the document instead of sentence selection. First, each model is trained (fine-tuned) and evaluated on TREC-DL-Doc'19, as it offers an abundance of training data. For Core17 and ClueWeb09, we use the model from the TREC-DL-Doc'19 experiment as initialization. This helps us to properly train the selector, as, unlike the BERT ranker, it does not start from a pre-trained model.

The results are illustrated in Table 5.2. Table 5.3 shows additional neural baselines [160, 78, 37, 72] evaluated on TREC-DL-Doc'19. The pipeline model works quite well on the Core17 dataset, but falls short on TREC-DL-Doc'19 and ClueWeb09 compared to the end-to-end models. In the pipeline model, the selection phase is independent of the ranking phase; hence, the selection strategy does not receive any feedback from the ranking phase. It is evident from Table 5.2 that the end-to-end approach improves the ranking process.

By selecting sentences from the complete document, our approaches perform similarly to stand-alone rankers that operate only on the head of the documents, specifically BERT-CLS. This indicates that most documents contain redundant information (likely in the form of summaries) near the beginning that BERT-CLS is able to exploit. We confirm this in Section 5.4.5 by showing that there is little overlap between the document head and the sentences selected by S&R-LIN. Thus, in Section 5.4.6, we use a Select-And-Rank model to explain the predictions of BERT-CLS by specifically selecting sentences from just the head of the documents.

Next, we evaluate our S&R-LIN model on various datasets provided by the BEIR benchmark and compare it to sparse retrieval methods (BM25) and a standard BERT-CLS model. The models are trained on MS MARCO, i.e., only the results on that dataset are in-domain, whereas the other datasets are evaluated in a zero-shot fashion. The contextual models are used to re-rank the top-100 BM25 results, which we retrieved using Elasticsearch. The results are illustrated in Fig. 5.6. It is evident that S&R-LIN and BERT-CLS show similar ranking performance on most datasets with only a few exceptions. It is interesting to note that, in some cases, contextual re-ranking models fail to improve BM25 ranking. We assume the reason for this to be a lack of domain knowledge due to the zero-shot setup.

As shown in Section 5.3.1, the datasets contain mostly short documents (or passages). As a consequence, selecting as many as $k = 10$ sentences might already select the complete document in some cases. We thus perform additional experiments on some of the datasets, decreasing k all the way to a single selected sentence. Figure 5.7 shows the results. This ex-

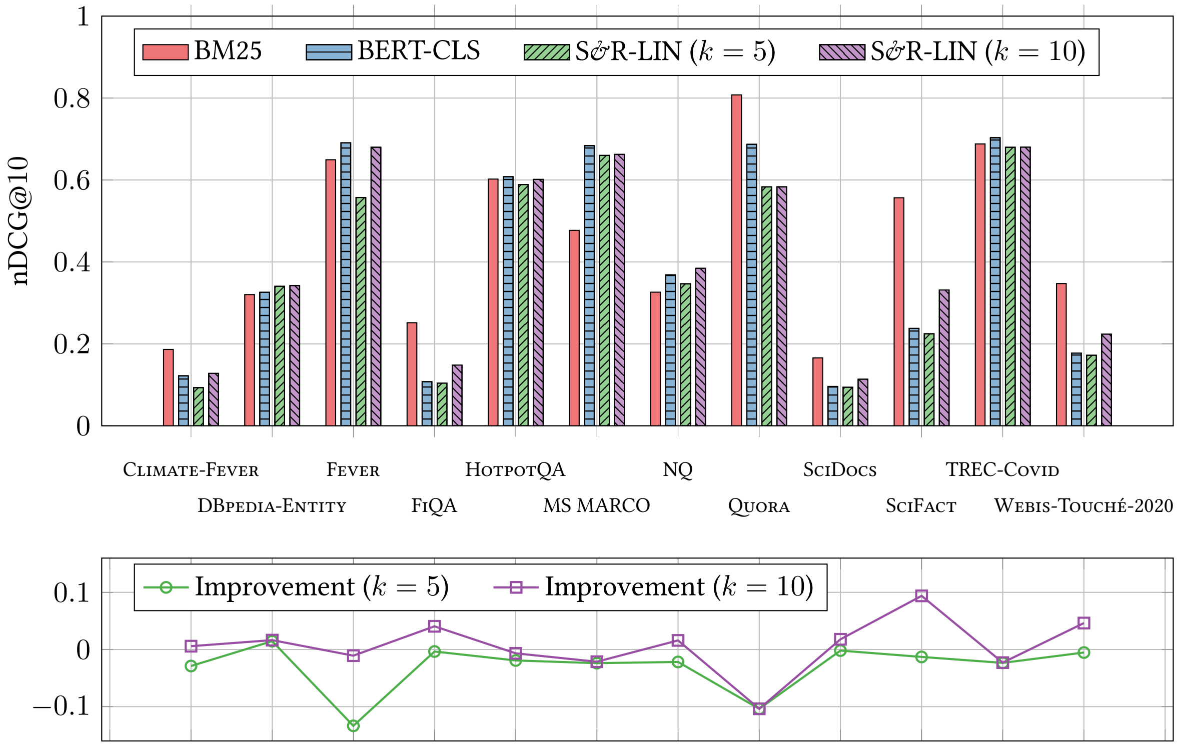
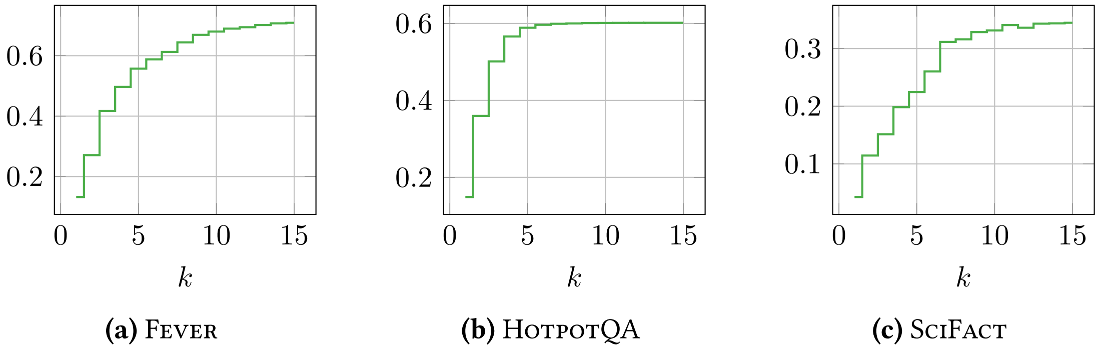


Figure 5.6: Ranking results (nDCG@10) on datasets from the BEIR benchmark using zero-shot evaluation with models trained on the MS MARCO dataset. The lines show the difference between S&R-LIN and BERT-CLS performance. Positive improvement indicates that S&R-LIN performs better, negative improvement indicates the opposite.



(a) Fever **(b)** HotpotQA **(c)** SciFact

Figure 5.7: Ranking results (nDCG@10) using S&R-LIN with decreasing number of selected sentences (k).

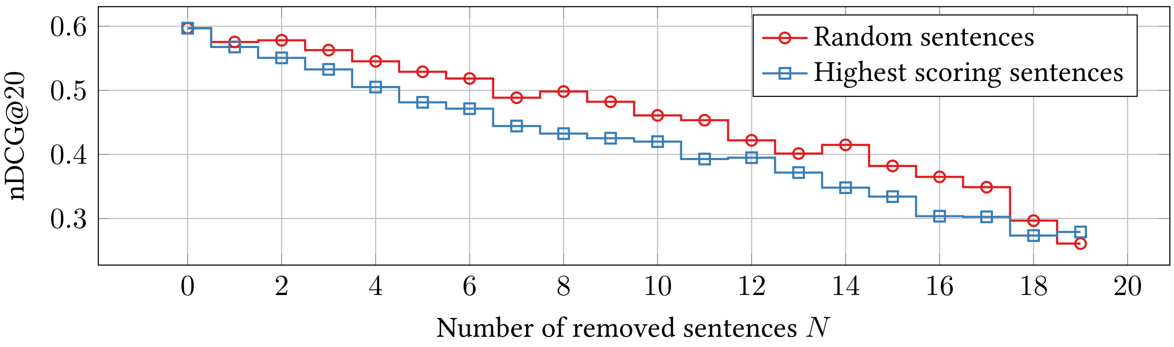


Figure 5.8: Ranking results (nDCG@20) on TREC-DL-Doc'19 using S&R-LIN with $k = 20$, where N sentences are removed (leaving $k - N$ sentences). We compare removing random sentences and the highest scoring sentences.

periment nicely illustrates the controllable trade-off between performance and interpretability: On each of the datasets, the performance plateaus once a certain number of selected sentences is reached, which depends on the document lengths of the dataset. On the other hand, the performance drops when the number of selected sentences is decreased, which in turn makes the ranking decision more interpretable.

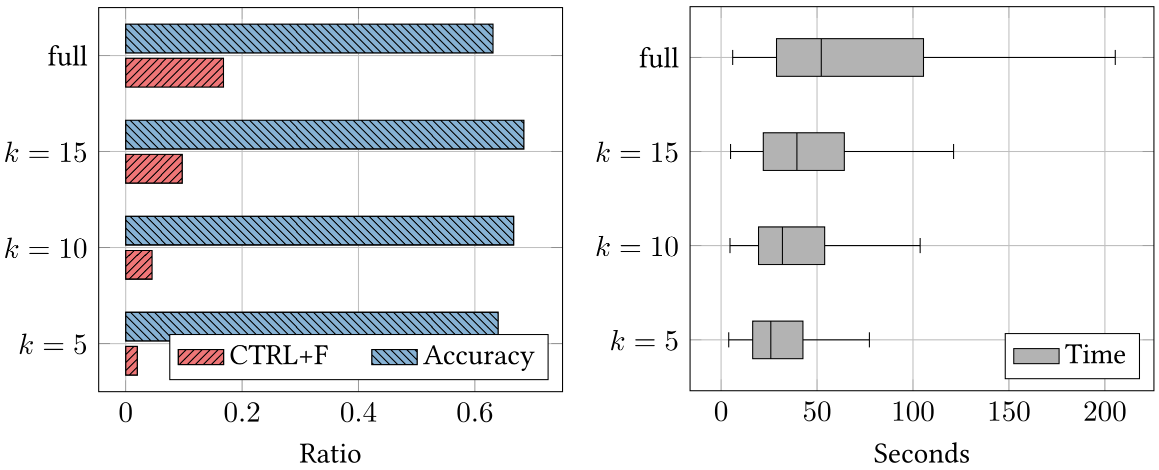
5.4.3 Comprehensiveness of Select-And-Rank

Comprehensiveness [41] is a metric that evaluates the quality of *rationales*, i.e., parts of the model input that aim at explaining the corresponding output. Specifically, a *contrast example* $\tilde{x}_i = x_i \setminus r_i$ is constructed for each input x_i, where the rationales r_i are removed. Comprehensiveness is then computed as

$$\text{Comp}(x_i) = m(x_i)_j - m(\tilde{x}_i)_j, \tag{5.20}$$

where $m(\cdot)_j$ is the model output or prediction corresponding to the class j.

Intuitively, comprehensiveness measures the degree of influence the rationales have on the final prediction by computing how much worse the model performs without them. In our case, x_i is a query-document pair. However, due to the length of the documents and limitation of ranking models, the ranker does not see the complete document in the vast majority of cases. Thus, computing the exact comprehensiveness is difficult. Instead, we use a proxy to get an idea about the comprehensiveness of Select-And-Rank models: During evaluation, we remove the N highest scoring (as assigned by the selector) sentences (out of k selected sentences) from the input and observe the drop in performance. We then compare the results to



(a) The accuracy of relevance judgments and usage of the web browser's integrated search function.

(b) The time taken to complete a single query-document relevance judgment. Outliers omitted.

Figure 5.9: The results of our user study to determine the faithfulness of Select-And-Rank explanations. Participants were presented with a query-document pair, where the document was either unaltered (*full*) or $k \in \{5, 10, 15\}$ sentences were selected using a the selector of a trained S&R-LIN model. The query-document pairs originate from TREC-DL-Doc'19. We plot the average accuracy, the fraction of instances where participants used their browser search, and the time taken to complete a single query-document relevance judgment.

1. the performance with all k sentences and
2. the performance when N random sentences are removed instead.

The results on TREC-DL-Doc'19 with $k = 20$ are illustrated in Fig. 5.8. They show that removing high-scoring sentences has a higher impact on overall performance than removing random sentences. This suggests that higher scoring sentences have a higher impact on the model predictions.

5.4.4 Faithfulness and Utility of Select-And-Rank

Generally speaking, the *faithfulness* of interpretations refers to the degree to which they accurately represent the reasoning of the model [82]. In the case of Select-And-Rank, this corresponds to the question *how well the selected sentences represent the document they originate from*. This problem is closely tied to the actual utility and usefulness of Select-And-Rank models for *human users*; the idea is that the users should be able to comprehend a ranking decision solely based on the selected sentences, i.e., the explanation. In order to assess how faithful the explanations are for human users, we have conducted a study which is described in this section.

Study Setup

We randomly selected 30 queries from the TREC-DL-Doc'19 test set for our study. For each of these queries, we randomly sampled one relevant and one irrelevant document from the official query relevance judgments. We used the selector of a trained S&R-LIN model (from Section 5.4.2) to select $k \in \{5, 10, 15\}$ sentences for each document (with respect to the corresponding query), resulting in four variations of each query-document pair. In total, we ended up with 240 (q, d, k) instances (where k can be `null`, representing no sentence selection). We employed 80 participants for the study, each of which judged 12 individual (q, d, k) instances (i.e., 960 relevance judgments in total). Thus, each instance was judged approximately four times.[2] Instances were allocated to participants randomly, making sure that no participant ever saw two instances with the same query and document.

The user interface presents the query at the top and the document just below. Within the document, a line break is inserted after every sentence. At the bottom, the participant is asked to indicate

1. whether or not the document is relevant to the query and

2. whether they used their browser's integrated search function for this instance.

We further measure and record the time taken for each relevance judgment. After each instance, an intermediate page prompts the participant to take a break before the next instance if necessary, such that the recorded times are less noisy.

Our study is implemented using the oTree framework [24] and was conducted on the Prolific platform.[3] Additional details can be found in Section 5.5.

Study Results

The results are illustrated in Fig. 5.9. It is apparent that the longer the documents are (in terms of the number of sentences), the more the average time taken to judge the relevance of a single query-document pair increases. Additionally, participants resort to the usage of their browser's search function more often, but this is not enough to compensate for the increased length and keep the time down. Moreover, the participants' accuracy remains roughly stable across all settings, peaking at $k = 15$. We assume that the drop in accuracy for the full documents could be caused by participants relying too much on term matching provided by their browsers rather than reading the complete documents.

[2] Due to some participants never finishing the study, this number can vary in rare cases; however, each instance has been judged at least three times.

[3] https://www.prolific.co/

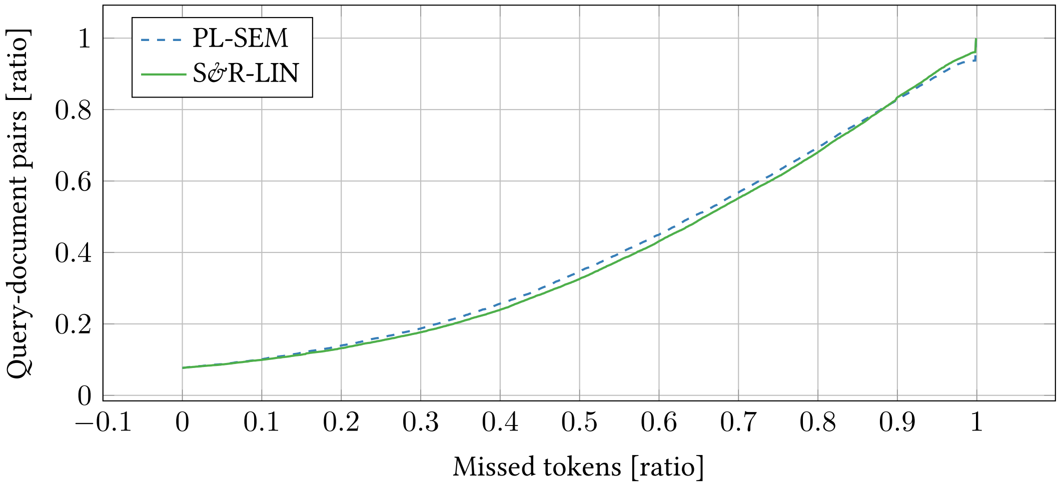
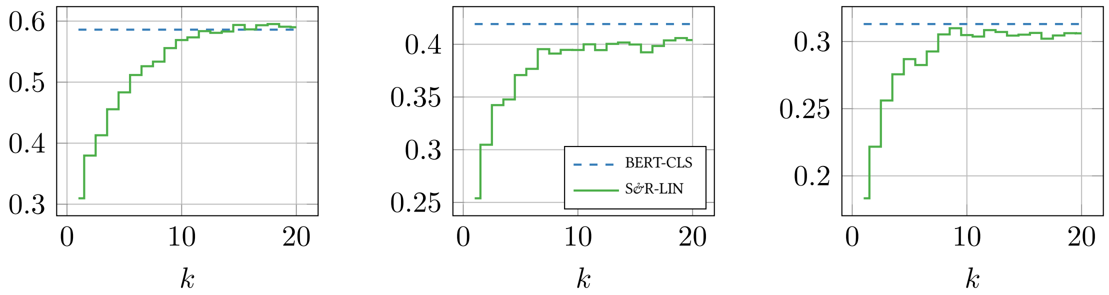


Figure 5.10: CDF of tokens on TREC-DL-Doc'19 that would have been missed without sentence selection.

Overall, our study highlights the utility of Select-And-Rank models to humans: The sentences extracted by our approach serve as *faithful* explanations to users, as is apparent from the accuracy. At the same time, it enables them to judge documents more quickly using only a small subset of sentences.

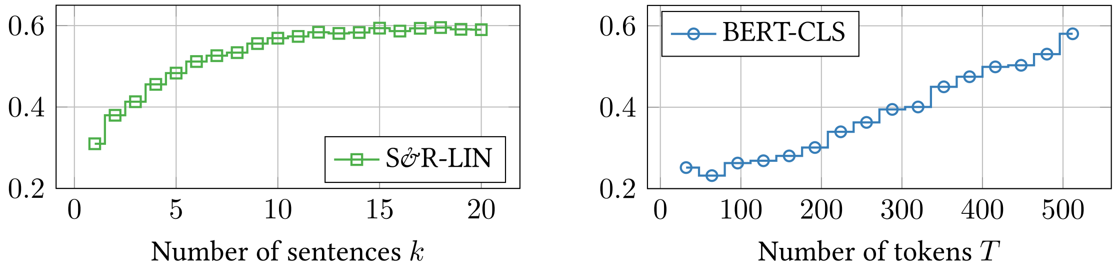
5.4.5 The Effect of Token Limitation

In this section, we analyze the token limitation that is inherent to the BERT ranker and further the role the selection strategy has in mitigating that limitation. In other words, we answer the following question: *How many input tokens of the selected sentences would not have been seen by BERT without selection due to length restrictions?* In general, existing research assumes that most of the information relevant to the query is present in the first part of the document [155]. The BERT-CLS baseline also works based on that assumption. However, recent strategies [67] show that some information also exists beyond this token limit. In [36], the authors try to handle this by selecting the first, last, and 28 random passages in their Doc-Labeled approach, but this heuristic does not always work. To that end, we choose the top-20 sentences based on PL-SEM and S&R-LIN and measure what fraction of these tokens exceeds the usable BERT input, i.e., is lost when we only consider the head of a document. Figure 5.10 shows the cumulative distribution of the ratio of missed tokens for TREC-DL-Doc'19. The distribution pattern is similar for both methods: Less than 10% of the query-document pairs do not miss any of the selected tokens. Given the performance of the models shown in Section 5.4.2, this suggests that relevant information is repeated within the documents, such that multiple selections exist which result in similar performance.



(a) TREC-DL-Doc'19 **(b)** Core17 **(c)** ClueWeb09

Figure 5.11: The performance of S&R-LIN (nDCG@20) applied only to the first 20 sentences of each document, which approximates selecting sentences from the input of BERT-CLS.



(a) Select-And-Rank **(b)** Truncation to T input tokens

Figure 5.12: Ranking performance (nDCG@20) on TREC-DL-Doc'19, where the length of the input to the ranker is limited in two ways. On the left, Select-And-Rank is used to select k sentences from the head of the document (i.e., from the first 20 sentences). On the right, inputs are simply truncated, i.e., the ranker only sees T tokens in total, which includes the query and the first part of the document. For BERT-CLS, $T = 512$ is the default setting and is consistent with the results in Table 5.2 and Fig. 5.11.

5.4.6 Explaining BERT-CLS

In Section 5.4.5, we showed that S&R-LIN and the standard BERT-CLS model operate on different parts of the input documents, yet they achieve comparable performance (cf. Table 5.2). In this section, we explore whether Select-And-Rank models can be used to further sparsify the head of a document and thus explain the predictions of BERT-CLS.

To that end, we conduct a set of experiments where we limit the available sentences for the selector to choose from to the first 20 of each documents based on our length estimation (cf. Section 5.3.1). We then vary k to compare the performance with respect to sparsity. The results are illustrated in Fig. 5.11 in terms of nDCG@20. We observe that the performance plateaus for roughly $k \geq 10$ (slightly later for TREC-DL-Doc'19) and approximately matches BERT-CLS. For lower values of k, the performance drops.

In addition, we compare the above result to a simpler strategy, where, instead of using

	QL		QL+RM3	
	AP	nDCG@20	AP	nDCG@20
QL(+RM3)	0.237	0.487[ab]	0.272	0.513[ab]
Doc-Labeled	0.203	0.434[ab]	0.219	0.426[ab]
BERT-3S	0.245	0.519[ab]	0.281	0.539
BERT-CLS	0.260	0.581	0.279	0.559
PL-SEM	0.265	0.571	0.268	0.537
[a] S&R-LIN	0.269	0.597	0.286	0.568
[b] S&R-ATT	0.271	0.590	0.284	0.563

Table 5.4: Performance over two first stage retrieval models, QL and QL+RM3, at depth 100 on the TREC-DL-Doc'19 test set using $k = 20$. Significant improvements (nDCG@20) at a level of 95% are indicated by superscripts.

Select-And-Rank to select sentences, we limit the length of the ranker input by simply truncating it to a constant number of tokens. This is identical to the BERT-CLS approach, but instead of 512 tokens, we use smaller numbers. Figure 5.12 shows the comparison of the two methods (S&R-LIN with k sentences and BERT-CLS truncated to T tokens). On the far right side of each of the plots, i.e., $k = 20$ and $T = 512$, there is no selection or truncation, thus, both models have roughly the same performance. However, decreasing k or T, respectively, it becomes evident that, by selecting relevant sentences using Select-And-Rank, substantially higher performance can be reached with similar numbers of input tokens. For example, comparing $k = 10$ and $T = 256$, both of which drop (roughly) half of the tokens, Select-And-Rank achieves an nDCG value of 0.569, while BERT-CLS only reaches 0.363. This suggests that Select-And-Rank is able to select representative summaries of the documents that are sufficient for the ranker to output similar performance. Truncation, on the other hand, does not have the same effect, which ultimately reflects in the performance.

Overall, these experiments show that sentence selection may be used even in combination with models that only operate on the head of documents to achieve interpretability while maintaining performance.

5.4.7 The Effect of First-stage Retrieval

From Section 5.4.2 and Section 5.4.6 it is evident that the performance of S&R-LIN is on par with the baselines, while maintaining the interpretability aspect of the approach. However, the re-ranking performance of the models is computed over the top-100 documents per query, retrieved using a QL model. One obvious question is, whether this performance is lost with a better first stage retrieval system. To answer this question, we re-retrieve the top-100 documents with QL and RM3 and apply the models to that set. Table 5.4 shows the results on TREC-DL-Doc'19. Note that the models are not re-trained, i.e., the models from previous experiments are used. There is no significant influence of RM3 on the performance of baselines; rather, performance drops to some extent in terms of nDCG. We assume that the reason for this is the fact that the models were not re-trained using the documents retrieved by QL and RM3.

5.4.8 Anecdotal Examples

In Table 5.5, we present an anecdotal example of the top sentence for each document, selected by our Select-And-Rank approaches in both pipeline and end-to-end variants. The documents marked as relevant are the ground-truth documents as assessed by TREC annotators. We see that the selected sentences already provide an insight into the what evidence is considered important by the overall ranking model. Specifically, the rank 5 prediction by S&R-ATT happens because it mistakes *bow pose* in yoga with bows and arrows. It is clear from the selected sentence of PL-SEM that it does not consider the duration aspect of the query. A key aspect of Select-And-Rank is that the decision of the final ranker can be unambiguously attributed to these extracted sentences, providing interpretability to the model decision. Note that we cannot completely explain the decision making of the final ranker, since it could select a further subset of the selected sentences.

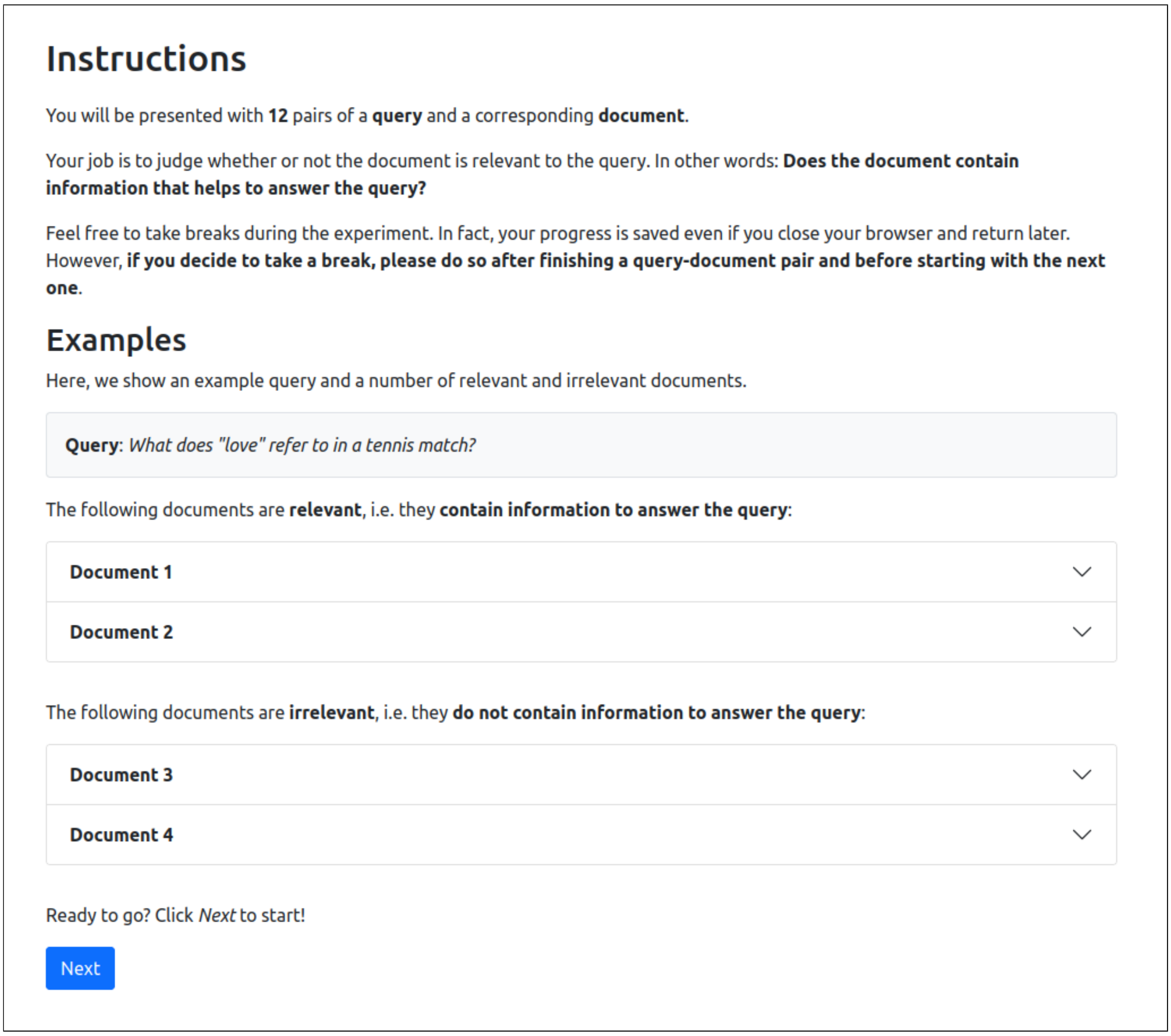
Moreover, we present examples from the Fever dataset in Table 5.6. It shows the two relevant documents for a query (here: a fact), each split into sentences. These sentences are then ranked by their scores w.r.t. the query as assigned by the selector model (S&R-LIN). Finally, the 5 highest scoring sentences are selected as input for the ranker. This setting is consistent with the experiments in Section 5.4.2 and Fig. 5.6. The part corresponding to the first document depicts the case where the sentence selection works well: The sentence that contains the answer to the query is scored high and thus selected. The ranker receives the selected sentences and is able to rank the document high (rank 3). On the contrary, in the second relevant document, the selector misses the only relevant sentence and does not include it in the selection. Thus, the ranker does not see the relevant part of the document and consequently ranks it lower (rank 23). This example further illustrates how each ranking decision can be attributed to a small fraction of the input document.

Rank	Document	Most relevant sentence
S&R-ATT		
1	D970461^{+}	How long do I hold yoga poses?
2	D3378721^{+}	How Long to Hold Bikram Yoga Poses.
3	D970460^{+}	How Long You Should Hold A Yoga Posture?
4	D1211050^{+}	How Long To Hold Yoga Pose To Gain All The Benefits?
5	D337672^{-}	One way to build strength and endurance is to pull your hunting bow [...] before releasing the arrow [...]
6	D2587656^{-}	Traditional Closing of a Yoga Practice [...] the teacher will say “namaste” & bow to students.
7	D1125612^{-}	Consult your doctor before beginning these new flexibility exercises [...]
8	D520508^{-}	Yoga should be done with an open, gentle, and non-critical mind [...] working on one’s limits
PL-SEM		
1	D3378723^{-}	[...] Bow Pose is an intermediate yoga backbend that deeply opens the chest and the front of the body.
2	D970458^{+}	In the style of hatha yoga I teach there are longer holds in the poses.
3	D3378725^{-}	[...] After a brief break, you move into the last eight standing exercises [...] of the Bikram yoga sequence
4	D970461^{+}	How long do I hold yoga poses?
5	D337672^{+}	[...] isolate the muscles needed to pull the bow back and hold the bow up [...]
6	D2285733^{+}	Straighten your legs, so that your body makes a ‘V’ shape and hold this position for 2 to 5 breaths.
7	D1930297^{-}	IF YOU ARE A BEGINNER, YOU OUGHT TO BEND YOUR KNEES SLIGHTLY TO ACCOMPLISH THIS.
8	D520508^{-}	Iyengar yoga can be good for physical therapy [...] easier for some people to get into the yoga postures.
Query: “*how long to hold bow in yoga*” (query ID 1132213)		

Table 5.5: Example rankings from the TREC-DL-Doc’19 dataset with the most relevant selected sentences. Document IDs have a suffix (+/-) indicating the relevance of the corresponding TREC judgments.

	Rank	Sentence
Document: `Commodore_(rank)`, **Rank**: 3		
selected	**1**	A commodore's ship is typically designated by the flying of a Broad pennant, as opposed to an admiral's flag.
	2	Commodore is a naval rank used in many navies that is superior to a navy captain, but below a rear admiral.
	3	It is sometimes abbreviated: as "Cdre" in British Royal Navy, "CDRE" in the US Navy [...]
	4	Commodore (rank).
	5	Non-English-speaking nations often use the rank of flotilla admiral or counter admiral [...]
	6	As an official rank, a commodore typically commands a flotilla or squadron of ships [...]
	7	Traditionally, "commodore" is the title for any officer assigned to command more than one ship [...]
	8	It is often regarded as a one-star rank with a NATO code of OF-6 [...]
Document: `Rear_admiral`, **Rank**: 23		
selected	**1**	In the German Navy the rank is known as Konteradmiral, superior to the flotilla admiral (Commodore in other navies).
	2	In the Royal Netherlands Navy, this rank is known as schout-bij-nacht (lit.
	3	[...] and in the Canadian Forces' French rank translations, the rank of rear admiral is known as contre-amiral.
	4	In some European navies (e.g.
	5	In many navies it is referred to as a two-star rank (OF-7).
	...	...
	13	Rear admiral is a naval commissioned officer rank above that of a commodore and captain, and below [...]
	14	Each naval squadron would be assigned an admiral as its head, who would command from the centre vessel [...]
Query: "*Commodore is ranked above a rear admiral.*" (query ID 204575)		

Table 5.6: Example selections of sentences from relevant documents w.r.t. a query from the FEVER dataset by S&R-LIN. The selector selects the highest scoring $k = 5$ sentences from each document. The final rank of a document is computed using only these sentences. The sentences that contain the answer are highlighted.



Instructions

You will be presented with **12** pairs of a **query** and a corresponding **document**.

Your job is to judge whether or not the document is relevant to the query. In other words: **Does the document contain information that helps to answer the query?**

Feel free to take breaks during the experiment. In fact, your progress is saved even if you close your browser and return later. However, **if you decide to take a break, please do so after finishing a query-document pair and before starting with the next one**.

Examples

Here, we show an example query and a number of relevant and irrelevant documents.

Query: *What does "love" refer to in a tennis match?*

The following documents are **relevant**, i.e. they **contain information to answer the query**:

Document 1

Document 2

The following documents are **irrelevant**, i.e. they **do not contain information to answer the query**:

Document 3

Document 4

Ready to go? Click *Next* to start!

Figure 5.13: The instructions page. This page is only displayed once and includes a task description along with some examples.

5.5 User Study Details

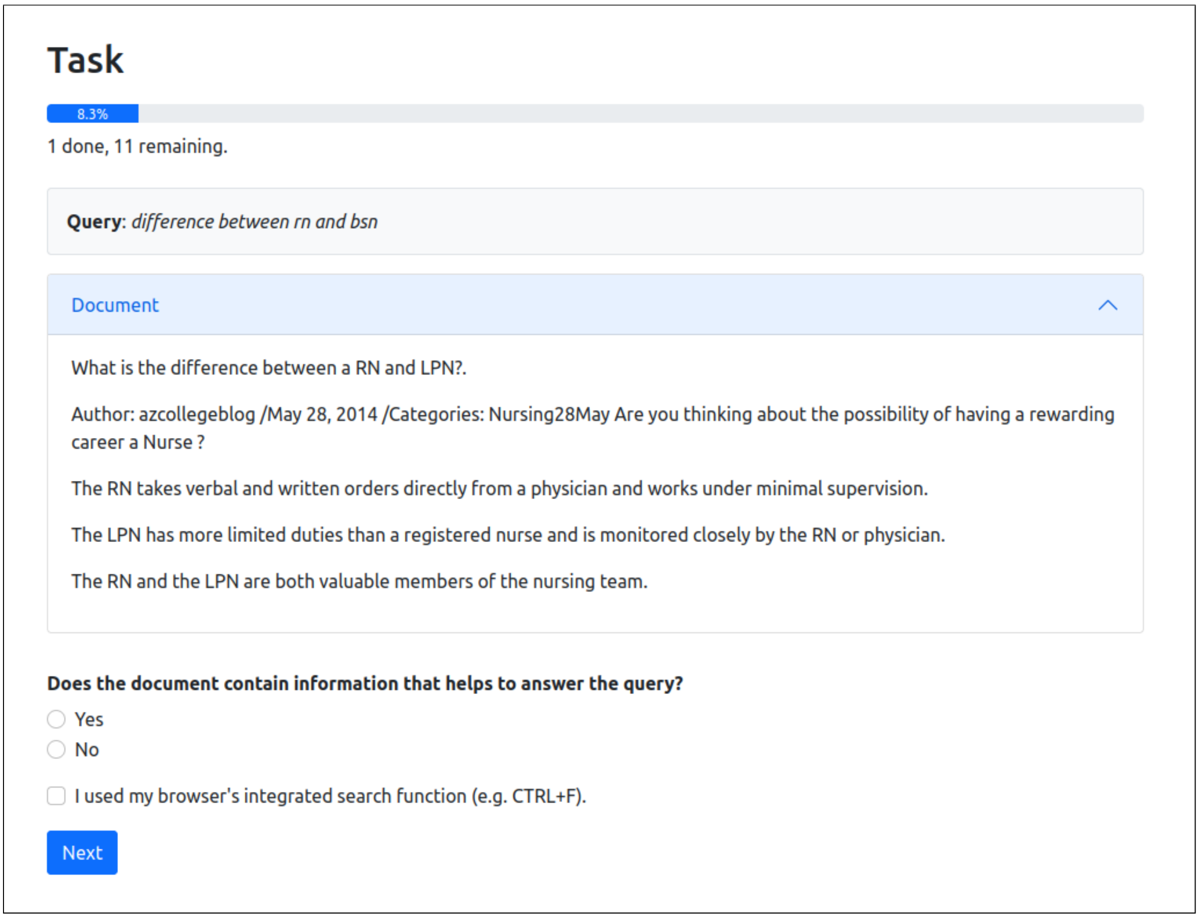
In this section, we present our user study (as described in Section 5.4.4) in more detail.

5.5.1 Interface

The main part of the study consists of three pages:

1. The *instructions page* (Fig. 5.13) familiarizes the participant with the task and provides examples.

2. The *task page* (Fig. 5.14) presents a query-document pair to the participant and records



Task

1 done, 11 remaining.

Query: *difference between rn and bsn*

What is the difference between a RN and LPN?.

Author: azcollegeblog /May 28, 2014 /Categories: Nursing28May Are you thinking about the possibility of having a rewarding career a Nurse ?

The RN takes verbal and written orders directly from a physician and works under minimal supervision.

The LPN has more limited duties than a registered nurse and is monitored closely by the RN or physician.

The RN and the LPN are both valuable members of the nursing team.

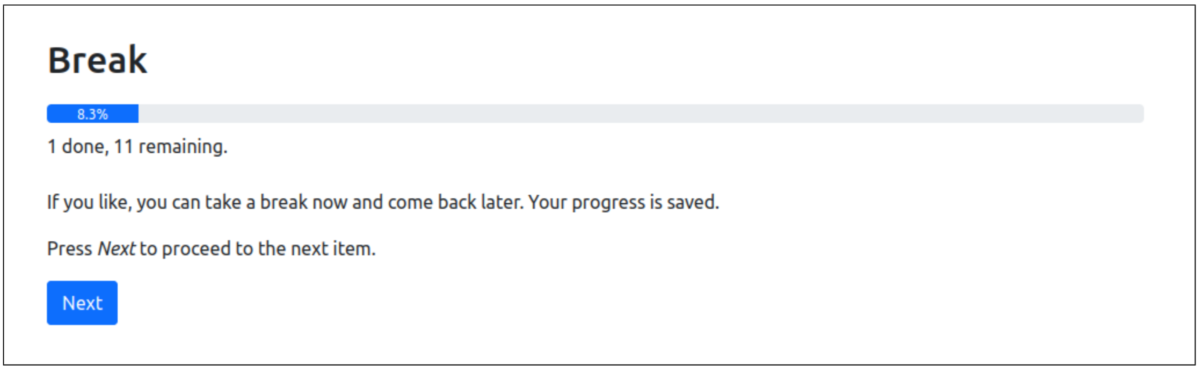
Does the document contain information that helps to answer the query?

○ Yes

○ No

□ I used my browser's integrated search function (e.g. CTRL+F).

Figure 5.14: The task page. It shows a query-document pair and gathers the participant's response (whether the document is relevant to the query) along with whether or not they used the browser search for this instance.



Break

1 done, 11 remaining.

If you like, you can take a break now and come back later. Your progress is saved.

Press *Next* to proceed to the next item.

Figure 5.15: The break page. It is displayed between two consecutive task pages.

their response. In the background, we record how much time the participant has spent on this page in order to measure the time it took to judge the query-document pair.

3. The *break page* (Fig. 5.15) is a simple intermediate page that is shown in between two consecutive task pages. The reason for this is that we want participants to only take breaks *between* two tasks, so that our time measurements are as accurate as possible.

The study is structured in *rounds*; a round consists of a task page and a subsequent break page. In our experiment, each participant completed 12 rounds. Before the first round, the instructions are shown.

5.5.2 Collection and Usage of Data

We collect three data points during each round:

1. The relevance judgment (boolean),
2. the usage of browser search (boolean), and
3. the time taken to come up with the relevance judgment (float).

The data is inserted into a database after each page. This allows users to take a break from the study and continue where they left off later, even if they closed their browser in the meantime.

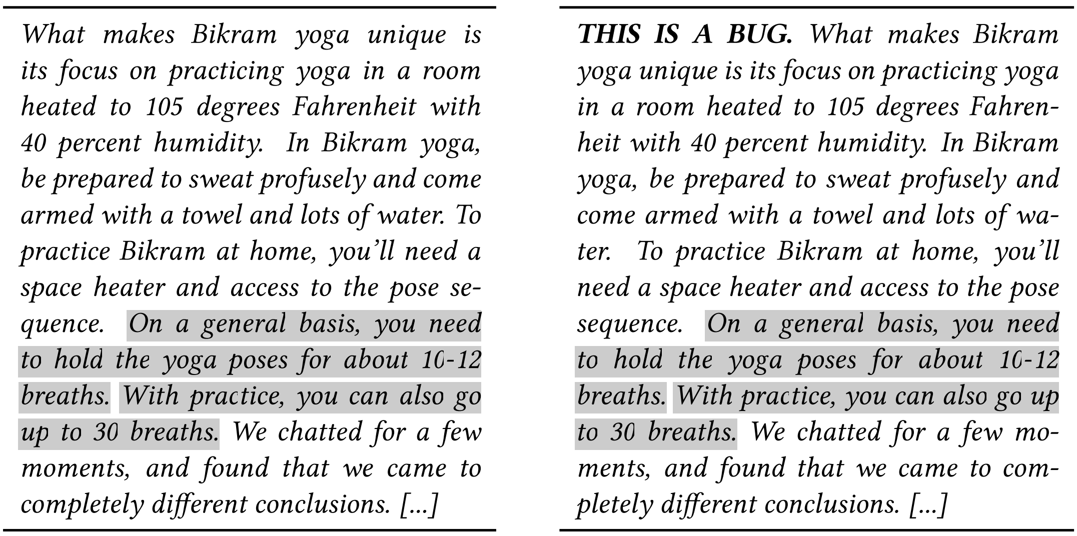
Computation of Metrics

In the results (Section 5.4.4) we present the following metrics:

1. **Accuracy**: This is simply the number of correctly judged instances divided by the total number of instances. Correctness is determined using the official TREC query relevances R, which are converted to binary according to the official guidelines, i.e., irrelevant ($R(q, d) = 0$) or relevant ($R(q, d) > 0$).

2. **Search function usage**: Similarly to accuracy, we divide the number of instances where participants have indicated the usage of their browser's search function by the total number of instances.

3. **Time taken to complete a relevance judgment**: The time is measured as the difference between two time stamps: The first one is recorded when the participant leaves the break page, and the second one is when the user completes the task page.



What makes Bikram yoga unique is its focus on practicing yoga in a room heated to 105 degrees Fahrenheit with 40 percent humidity. In Bikram yoga, be prepared to sweat profusely and come armed with a towel and lots of water. To practice Bikram at home, you'll need a space heater and access to the pose sequence. On a general basis, you need to hold the yoga poses for about 10-12 breaths. With practice, you can also go up to 30 breaths. We chatted for a few moments, and found that we came to completely different conclusions. [...]

(a) Unaltered relevant document

THIS IS A BUG. *What makes Bikram yoga unique is its focus on practicing yoga in a room heated to 105 degrees Fahrenheit with 40 percent humidity. In Bikram yoga, be prepared to sweat profusely and come armed with a towel and lots of water. To practice Bikram at home, you'll need a space heater and access to the pose sequence. On a general basis, you need to hold the yoga poses for about 10-12 breaths. With practice, you can also go up to 30 breaths. We chatted for a few moments, and found that we came to completely different conclusions. [...]*

(b) Relevant document with label leakage

Figure 5.16: An example illustrating label leakage for the query `how long to hold bow in yoga`. For each document that is relevant to the query (a), we prepend the sentence `THIS IS A BUG` (b). As irrelevant documents are unaffected, this simulates label leakage for training and test data. The highlighted sentences correspond to the part of the document that provides the answer to the query.

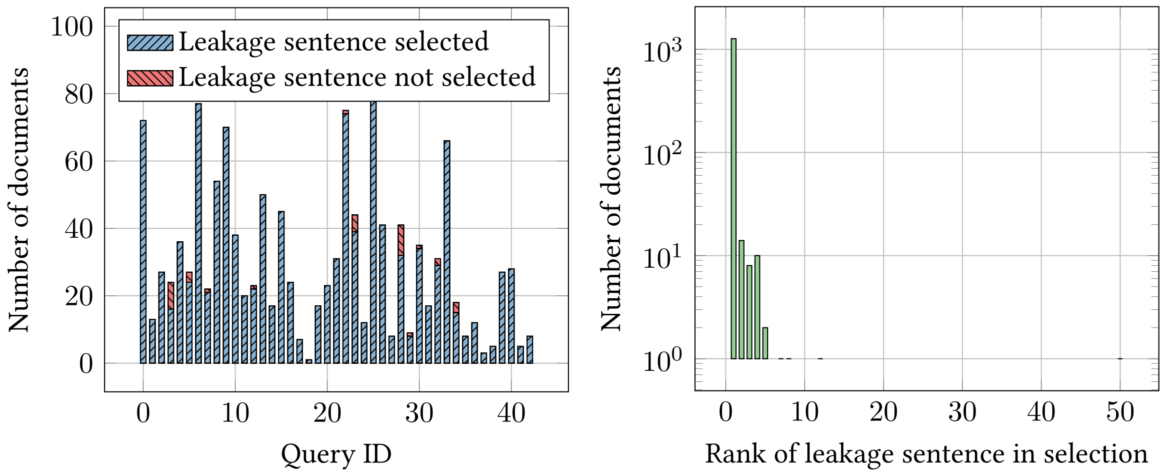
5.6 Applications of Select-And-Rank

In this section, we highlight several real-world applications of Select-And-Rank models.

5.6.1 Discovering Biased or Buggy Ranking Decisions

Neural rankers, just like any other machine learning model, are susceptible to bias or bugs in their ranking decisions [2]. Such models can achieve high performance, but the rationale (or reasoning) behind the model decisions is often incorrect, i.e., the models are right for the wrong reasons. In a similar fashion, a line of work employs *adversarial attacks* to craft model inputs, which are often merely slightly modified examples from real datasets, that yield highly unexpected model decisions or outputs [213, 201].

In this section, we conduct an experiment to show how the Select-And-Rank paradigm can be employed to uncover such biased or even buggy decisions of the ranking model. Specifically, we enforce biased ranking decisions by augmenting the MS MARCO corpus to include label leakage; this means that, for every query-document pair (q, d) in the training and test set, where d is relevant to q, we replace d by d', where d' is simply a copy of the original document with one additional sentence injected. This process is illustrated in Fig. 5.16. As a result, a ranking model trained on this data simply learns to rank documents that con-



(a) Fraction of documents where the leakage sentence has been selected (assigned the highest score by the selector) for each query

(b) Distribution of the ranks of the leakage sentence as assigned by the selector over all relevant documents in the test set

Figure 5.17: This example shows how label leakage can be discovered using Select-And-Rank models. The plots illustrate the sentence selections on the TREC-DL-Doc'19 test set by an S&R-LIN model using a modified corpus to simulate label leakage (cf. Fig. 5.16). Evidently, the extractive *explanations* (selected sentences) provided by our model reliably uncover the label leakage by including the leakage sentences.

tain the injected sentence high (independently of the query). In fact, the model reaches an AP of 0.39, nDCG@20 of 0.66, and RR of 1.00 on the TREC-DL-Doc'19 test set (cf. Table 5.2) due to the label leakage. Figure 5.17 shows how the explanations provided by an S&R-LIN model uncover the bias in the ranking decisions; specifically, it illustrates how the selector assigns the highest importance to the sentence containing the label leakage (and hence includes it in the *explanation*) in all but very few cases. As a result, an examination of the ranking explanations immediately uncovers this bug.

5.6.2 Improving Search Engines

In general, search engines do not give end-users much of an idea about the reasoning behind marking a document as relevant or ranking one document higher than another. Instead, users have to rely on the results of the search engines. In turn, most search engines use the click information of users to judge the relevance of a document and iteratively update their search and ranking algorithm [30]. This introduces bias in determining the importance of web pages. Content creators may use clickbait [22, 56] to attract users and increase the importance of their content or web page, even though it does not contain the relevant content. This is also a challenging task for search engine optimization.

Our Select-And-Rank-based document ranking architecture can be used to alleviate the above-mentioned two problems to an extent:

1. The Select-And-Rank paradigm identifies the relevance of a document with respect to a query and also extracts relevant snippets from the document. If the system highlights those snippets along with the document, users can make their click decisions more accurately, helping them to skip clickbait contents.

2. It is very difficult to judge the relevance of a document just from the title. For this reason, search engines display *snippets* of documents on the results page. These snippets are often relatively short (i.e., one or two sentences or parts of sentences) and are supposed to highlight why the user might be interested in the document. Usually, these snippets are based on term-matching with the query, i.e., matching terms in the snippet are printed bold. Select-And-Rank could be used as an alternative way of generating these snippets (for small values of k) such that they also explain the reasoning behind the ranking itself. The highlighting of matching terms could then be performed on the selected sentences as well.

5.7 Conclusion

In this chapter, we proposed Select-And-Rank, a ranking framework that is interpretable by design. Our selection and ranking models are trainable end-to-end by gradient-based optimization techniques, using a combination of the Gumbel-max trick and reparameterizable subset sampling. In our experiments, we found that, by enforcing sparsity in document representations by selecting a subset of sentences, we still perform on par with state-of-the-art models, while being interpretable. We showed how Select-And-Rank can be used to explain the decisions for a large number of ranking tasks from the BEIR benchmark in the zero-shot setting. This proves its potential of wide-ranging utility in a large number of knowledge-intensive tasks. We showed that there is no considerable performance difference in the case of complex selectors, indicating that simple and fast selectors can be used instead. We also found that there is a sweet spot in the choice of sparsity that varies depending on the dataset. We performed a user study to highlight the utility of our extractive explanations to human users. We believe that the applicability of a sparsity-inducing component can extend beyond document ranking to other ranking [74, 75, 184], graph [50], and web tasks [8, 183, 170].

6

Web Content Extraction for Corpus Creation

Web pages are rich sources of information but are also intertwined inextricably with ads, banners, and other boilerplate from content management systems (CMS). Extracting the primary content from a web page is an important low level task with strong implications to retrieval models [198] and other tasks.

In this chapter, we focus on the task of *boilerplate removal* or the isolation of the primary informational content of a web page. The problem is well studied, with approaches ranging from classical rule-based to modern supervised machine learning models. Most of the recent approaches rely on building large sets of features based on commonly observed domain knowledge or rules. Gupta et al. [63] and Wu, Liu, and Fan [214] rely on features of web pages based on their *document object model* (DOM), i.e., their structure. BoilerPipe [93] and Web2Text [198] rely on text-based features. Cai et al. [20] use vision-based page segmentation approaches. Most of these approaches exploit a crucial aspect that is common to most web pages, i.e., there is an inherent *locality* of relevant content [48]. In other words, relevant content tends to be rendered closely together in a web page.

There are two major drawbacks of the existing approaches. First, the *locality of rendering* effect is inherently hard to model and requires a large number of features and heuristic post-processing procedures, as exemplified in the previous approaches that use DOM trees, text, or a combination of both. For example, the state-of-the-art approach, Web2Text, uses 128 features. Second, the availability of training data for the boilerplate removal task is limited, and, in order to maintain their effectiveness on evolving web pages, these models have to be re-trained. With human training labels being at a premium, this limited training data is a

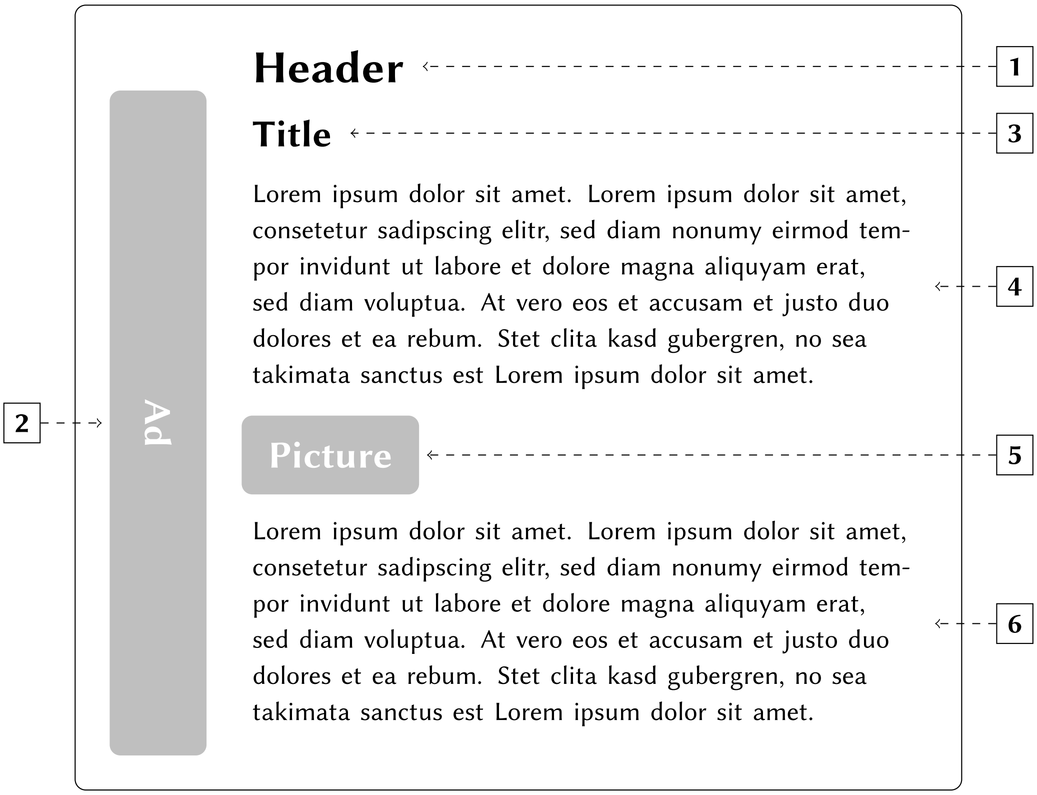

Header
Title
Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
Ad
Picture
Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
1
2
3
4
5
6

Figure 6.1: A common example layout that is found, for example, in blogs. This web page can be represented as a sequence, the elements of which are illustrated by the numbers corresponding to the page elements. The order of the sequence is determined by the order of the elements within the DOM tree.

common issue that plagues the existing models.

We propose BoilerNet, an automatic feature learning approach that does not rely on expensive feature engineering on web pages. Unlike the locality of rendering, there is also a *locality of authoring* in web pages. Specifically, relevant content tends to be authored in contiguous segments, as illustrated in Fig. 6.1. This observation greatly simplifies the modeling task, as we are not bound to inferring the locality of content from the visual features or raw text. Rather, we work on the raw HTML input and model the input web page as a sequence of candidate segments (text blocks, i.e., leaf nodes in the DOM tree). Assuming that the content is authored in a sequential fashion, we now pose the problem of detecting boilerplate as a sequence learning task. This eradicates the utility of any computationally expensive feature extraction and post-processing procedures. Our research question is the following:

RQ4 Can the content of web pages be extracted purely based on the sequence of its elements, without relying on hand-crafted features (Section 6.3.2)?

We perform extensive experimental evaluation with standard benchmarks as well as a new dataset created by us to show that our model is able to perform and generalize well even with a low number of training instances. We observe that we obtain similar performance and sometimes outperform the existing feature-based approaches. We provide an interactive demonstration in the form of a browser extension.

6.1 Related Work

Existing boilerplate removal approaches are either handcrafted rules or tools targeted at extracting content from web pages which were observed to possess certain structural and textual properties, or machine learning approaches using hand-crafted (textual, structural, visual, etc.) features to separate content from boilerplate.

For instance, Body Text Extraction (BTE) [48] uses the observation that the main content contains longer paragraphs of uninterrupted text and marks the largest contiguous text area with the least amount of HTML tags as content. Gupta et al. [63] apply various heuristic-based filters to remove images, advertisements, etc. from the DOM tree representation of the web page. CETD [188] exploits the design principles behind text and noise. Another line of work consists of template detection algorithms [12, 124, 226, 21] which utilize collections of web pages, usually from the same site, to learn the common template structure.

Among the other machine learning based approaches, Pasternack and Roth [161] propose a method utilizing *maximum subsequence segmentation* to extract the text of articles from HTML documents using tags and trigram features. BoilerPipe [93] analyzes hand-crafted text features, namely the number of words and link density, to distinguish main content from other parts of information from news article web pages. Wang et al. [204] learn a template-independent wrapper using a small number of labeled news pages from a single site and features dedicated to news titles and bodies. Gibson, Wellner, and Lubar [57] and Spousta, Marek, and Pecina [186] employ sequence labeling approaches, which consider a web document as a sequence of some appropriately sized units or blocks, and the task is to categorize each block as *content* or *not content*. Both of these works use conditional random fields (CRFs) with various hand-crafted textual and structural features. Web2Text [198] employs two separate convolutional neural networks which operate on a very large number of hand-crafted features for each text block, yielding unary and pairwise potentials (probabilities), for classifying each block and pair of adjacent blocks, respectively, followed by the maximization of joint probabilities during inference. Wu, Liu, and Fan [214] formulate the actual content identification problem as a DOM tree node selection problem and train a machine learning model using features like fonts, links, position, and others.

6.2 BoilerNet

We model the problem of boilerplate removal or content extraction as a *sequence labeling* problem where the web page is divided into text blocks, all of which are then individually classified as either content or boilerplate. Each text block is represented as a vector which encodes both the text as well as all of its ancestral HTML tags. The order of the input sequence is determined by the order of the corresponding text blocks within the original HTML file. Our hypothesis is that the order of text blocks in a web page encodes important information about their type, i.e., content or boilerplate, as the placement is determined by the authoring style. For example, in blogs or news sites, we expect the content of an article page to be bunched near the center and surrounded by ads and navigational elements.

6.2.1 Input Representation

In order to obtain the sequences to be used as inputs for our model, we divide every web page into a sequence of text blocks. A text block can only appear as a leaf node in the DOM tree and has no associated HTML tag. However, a piece of text that appears connected to the human reader may be separated by HTML tags, e.g.,

```
<p>[A]<strong>[B]</strong>[C]</p>
```

contains the text blocks `[A]`, `[B]`, and `[C]`, divided by a `<strong>` tag.

Each block is represented by a d-dimensional vector. The first k elements of the vector are used to encode all parent nodes of the leaf up to the root. Each index is associated with a specific HTML tag, and the numbers encode how many of the corresponding HTML tags appear in the path from the root node to the leaf. In the same fashion, the remaining l items encode the words that appear in the text block. In practice, we add another two dimensions for out-of-vocabulary placeholders to the vector in order to handle unknown HTML tags and unknown words and only consider the k most common HTML tags and the l most common words.

6.2.2 Sequence Labeling

The BoilerNet architecture is based on bidirectional LSTMs, which are able to learn complex non-local dependencies in sequence models. The elements in the input sequence are projected to m-dimensional dense vectors using a fully-connected layer $\mathbf{D} \in \mathbb{R}^{d \times m}$. The dense representations are then fed into a number of consecutive bidirectional many-to-many LSTM

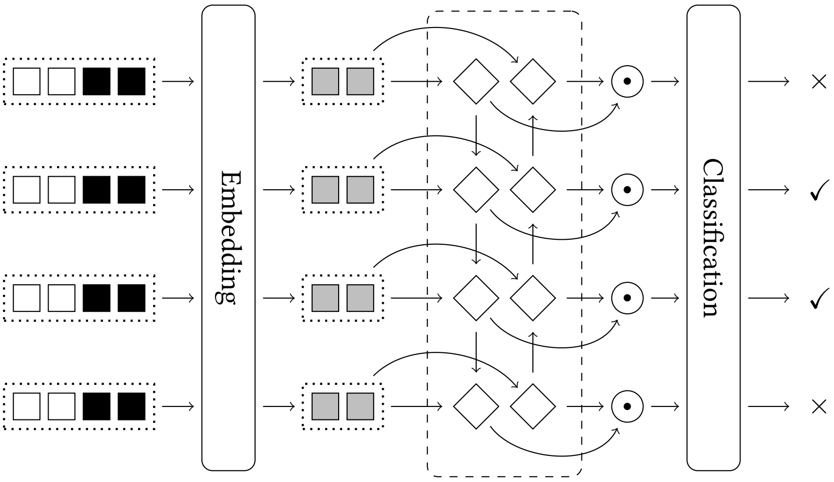


Figure 6.2: The BoilerNet architecture with one LSTM layer. The input is a web page represented as a sequence of text blocks. We represent each text block using a sparse vector which encodes the HTML tags (white) and words (black). These input vectors are then transformed into lower-dimensional dense vectors using an embedding layer. We classify each element of the sequence.

layers. In each of these LSTM layers, given an input $(x_1, \ldots, x_n)$, the forward LSTM yields the output $(y_1^f, \ldots, y_n^f)$, while the backward LSTM receives the reversed input sequence and yields the output $(y_1^b, \ldots, y_n^b)$. Both LSTMs use individual sets of parameters $\mathbf{W}^f$ and $\mathbf{W}^b$. The final ℓ-dimensional representation of a text block is obtained by concatenating the respective forward and backward outputs. Finally, the concatenated outputs of the last LSTM layer are squashed into 1-dimensional vectors using a fully connected layer $\mathbf{V} \in \mathbb{R}^{\ell \times 1}$ with a sigmoid activation function to infer the final classification probabilities. We use binary cross entropy to train the model. The architecture is shown in Fig. 6.2.

6.2.3 Issues in Boilerplate Removal Models

One of the main issues in employing machine learning models to classify content and boilerplate is the scarcity of the labeled data. Annotating content in web pages is a tedious task and requires a large investment of time, mainly because of the growing size of web pages and non-standard use of HTML. We initially experimented with a *weak supervision* [38] strategy to deal with this problem, which resulted in no significant improvements. We therefore conduct our experiments with a low number of training examples to show that our model is able to generalize well despite limited training data.

6.3 Experiments

We evaluate BoilerNet against existing approaches using two datasets: CleanEval [13] and GoogleTrends-2017. CleanEval (published in 2007) is a collection of arbitrary websites. In order to evaluate the generalizability of our model to newer web pages, we manually created GoogleTrends-2017 as described in Section 6.3.1.

We compare our approach to other machine learning as well as rule- and heuristic-based methods. Web2Text [198] relies on a large number of hand-crafted features and uses convolutional neural networks to classify page elements. The drawback of this approach is that these features might not generalize well or in some cases even be invalid, for example, across languages or longer time periods. BoilerPipe [93] is a rule-based system using textual features which was trained solely on news articles. Finally, Readability.js is the open source implementation of Mozilla's *reader view* feature in the Firefox browser.[1]

6.3.1 Dataset Preparation

Existing datasets for the evaluation of boilerplate removal and content extraction have weaknesses, rendering them suboptimal for evaluating modern approaches. First, the datasets are old (CleanEval was published in 2007, L3S-GN1 in 2010); since then, the web has changed in many ways, for example, in overall structure and technologies used, rendering existing datasets outdated. Second, some datasets lack diversity, as they contain only web pages of a single type. For example, L3S-GN1 is a news-only dataset.

For these reasons, we created a new dataset based on the Google trends of 2017. We obtained the HTML files by retrieving the first 100 results for each trending Google query from the year 2017.[2] From the resulting pool of websites, we randomly sampled a set of 180 documents and annotated them.[3] To eliminate the problems mentioned above, we made sure to retrieve all our pages from current Google queries, as this ensures both variety and currentness of the data.

As already mentioned, some datasets provide the ground-truth (cleaned web page) as plain text files where all HTML tags have been stripped. Since our model takes the HTML tags into account during training, it is necessary to retrace each part of the cleaned web page in the original HTML document. This is not a trivial task, as the ground-truth might not be 100% accurate. The same problem occurs when it comes to the evaluation of other web page cleaners. Often times the output of such cleaners is in plain text format, which makes

[1] https://github.com/mozilla/readability

[2] https://trends.google.com/trends/yis/2017/GLOBAL/

[3] This was done in collaboration with Supplie [189].

	Negative class			Positive class		
	P	R	F_1	P	R	F_1
Web2Text	0.82	0.76	0.79	0.84	**0.89**	0.86
Web2Text (unary)	0.80	0.78	0.79	0.85	0.87	0.86
BoilerPipe	0.58	**0.90**	0.71	**0.90**	0.58	0.71
Readability.js	**0.85**	0.76	0.81	0.82	**0.89**	0.85
BoilerNet	0.82	0.83	**0.82**	0.87	0.86	**0.87**

Table 6.1: The results on the CleanEval dataset with 55 training, 5 validation, and 676 test instances.

the computation of token-level measures impossible. To solve this problem, we employ the *alignment* procedure from [198], which tries to find the original position of the text snippets in the HTML file via sequence matching.[4] This allows us to extract labels from datasets as well as from the output of web page cleaners. Additionally, we implement the BoilerNet model in a way that the default output is a valid HTML file, where the text blocks classified as content are simply wrapped in an additional <span> tag. This makes our model both easy to use and evaluate.

6.3.2 Results and Discussion

We conduct experiments on the CleanEval and GoogleTrends-2017 datasets to compare BoilerNet to the baselines.[5] Due to the scarcity of ground-truth data mentioned earlier, we intentionally use small numbers of training instances. This also has the advantage of giving us a larger test set. We experimented with 5-fold cross-validation, however, this did not result in any substantial changes. Our final model after validation contains two bidirectional LSTM layers with 256 hidden units each. The sparse input vectors are projected to 256-dimensional embedding vectors. After the second LSTM layer, we apply dropout with a probability of 0.5. We train each model for 50 epochs with a batch size of 16 and weighted binary cross entropy loss. We choose the best checkpoint based on the F_1 score on the validation set.

The results on the CleanEval dataset are shown in Table 6.1. We used the original CleanEval split, i.e., 55 training instances, 5 validation instances, and 676 test instances. Unlike all other approaches, BoilerNet is consistent in both classes, whereas the recall of Web2Text drops in the negative class. The overall performance of BoilerNet and Web2Text is similar, which shows that our approach is able to learn the fea-

[4] The adaptation and implementation of this algorithm was done by Supplie [189].
[5] The Web2Text and Readability.js baselines were evaluated by Supplie [189].

	Negative class			Positive class		
	P	R	F$_1$	P	R	F$_1$
Web2Text	0.88	0.89	0.88	0.69	0.67	0.68
Web2Text (unary)	0.91	0.85	0.88	0.67	0.77	0.71
BoilerPipe	0.78	**0.98**	0.87	**0.85**	0.26	0.39
Readability.js	0.84	0.85	0.84	0.53	0.52	0.53
BoilerNet	**0.95**	0.86	**0.90**	0.70	**0.88**	**0.78**

Table 6.2: The results on the new GoogleTrends-2017 dataset with 50 training, 30 validation, and 100 test instances.

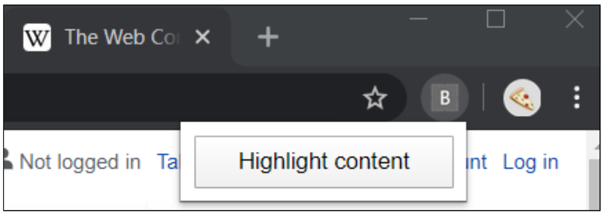
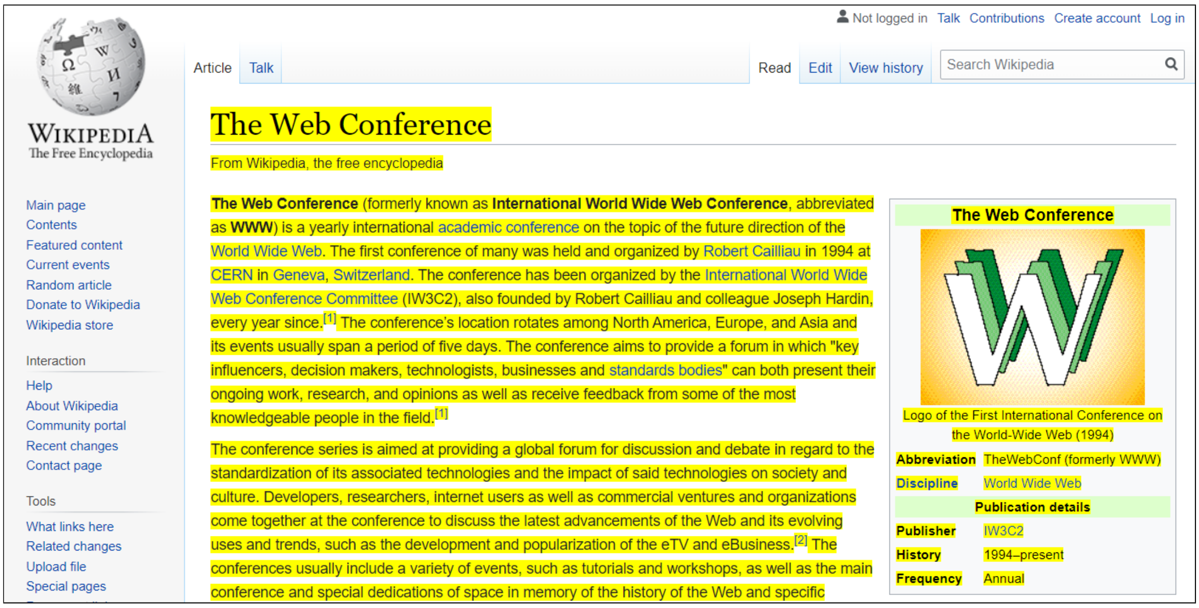


Figure 6.3: The user interface of the browser extension.

tures of Web2Text that were optimized for CleanEval. Table 6.2 shows the results on the GoogleTrends-2017 dataset. Again, BoilerNet is more consistent than the other approaches, outperforming them especially in the positive class. This supports our claim that hand-crafted features do not generalize well across longer periods of time. Moreover, while, for other approaches, it would be hard to detect boilerplate text that contains well formed sentences (e.g., copyright statements), in the case of raw input (like ours) we learn to predict such cases with higher accuracy.

6.4 Demonstration

We demonstrate our system by providing an interactive browser extension that allows the user to highlight the content on an arbitrary web page with a single click of a button. The user interface is shown in Fig. 6.3. After the user initiates the process, the extension preprocesses the current web page, loads a pre-trained BoilerNet model and classifies each text element. Those elements which are classified as content are then highlighted directly within the active browser tab. Figure 6.4 shows an example web page, where the content has been highlighted.



The Web Conference

From Wikipedia, the free encyclopedia

The Web Conference (formerly known as **International World Wide Web Conference**, abbreviated as **WWW**) is a yearly international academic conference on the topic of the future direction of the World Wide Web. The first conference of many was held and organized by Robert Cailliau in 1994 at CERN in Geneva, Switzerland. The conference has been organized by the International World Wide Web Conference Committee (IW3C2), also founded by Robert Cailliau and colleague Joseph Hardin, every year since.[1] The conference's location rotates among North America, Europe, and Asia and its events usually span a period of five days. The conference aims to provide a forum in which "key influencers, decision makers, technologists, businesses and standards bodies" can both present their ongoing work, research, and opinions as well as receive feedback from some of the most knowledgeable people in the field.[1]

The conference series is aimed at providing a global forum for discussion and debate in regard to the standardization of its associated technologies and the impact of said technologies on society and culture. Developers, researchers, internet users as well as commercial ventures and organizations come together at the conference to discuss the latest advancements of the Web and its evolving uses and trends, such as the development and popularization of the eTV and eBusiness.[2] The conferences usually include a variety of events, such as tutorials and workshops, as well as the main conference and special dedications of space in memory of the history of the Web and specific

The Web Conference

Logo of the First International Conference on the World-Wide Web (1994)

The Web Conference	
Abbreviation	TheWebConf (formerly WWW)
Discipline	World Wide Web
Publication details	
Publisher	IW3C2
History	1994–present
Frequency	Annual

Figure 6.4: An example web page, where the content has been highlighted by the BoilerNet browser extension.

6.4.1 Implementation Details

Our implementation of the BoilerNet model uses TensorFlow 2.0. The training (and evaluation) happens on a GPU. A trained model can then be loaded by our browser extension, which uses TensorFlow.js, a JavaScript library compatible to TensorFlow.[6] The extension is self-contained, i.e., it handles all necessary pre-processing steps like the tokenization of text paragraphs.

6.5 Conclusion

We presented BoilerNet, a novel, featureless approach for boilerplate removal from web pages using sequence labeling. We have shown that BoilerNet can match the performance of state-of-the-art systems and outperform them on more current datasets, achieving an increase of 11% in recall and 7% in F_1 (positive class) over its competitors. We have also shown the benefits of modeling web pages as sequences of text blocks while preserving the order from the DOM tree. Additionally, we have shown that our approach requires only little training data to achieve good results.

[6] https://www.tensorflow.org/js

7 Conclusion and Future Work

Neural ranking is becoming ubiquitous, as more and more services that rely on it are interwoven with many peoples' everyday life through the likes of personal assistants, web search engines, or recommender systems in e-commerce or streaming services. With dense and hybrid retrieval strategies powered by large pre-trained language models taking over, the quality of search results and any related tasks has improved significantly over the recent years; these improvements, however, come at a cost.

We have identified two important challenges of neural ranking models: On the one hand, their complexity and size have negative implications on their efficiency (cf. Section 1.2.1—the *efficiency challenge*); this affects both the query processing latency and, arguably more importantly, their environmental footprint during training and inference. On the other hand, the decisions of neural rankers are difficult to explain (cf. Section 1.2.2—the *explainability challenge*).

In this thesis, we focused mainly on improving the efficiency and explainability of neural rankers in order to mitigate the aforementioned challenges. Below, we list the contributions and give an outlook on promising future research directions.

7.1 Contributions

In Chapter 3, we introduced Fast-Forward indexes, a framework that allows for simple and efficient ranking using dual-encoder models. Our method exploits pre-computation of document representations using dual-encoders in order to offload the most computationally expensive operations to the offline (indexing) phase. The online ranking phase entails computing the query representation and scoring each document; these operations happen on CPUs and do not require GPU acceleration. Ranking the documents using a linear interpolation of lexical and semantic scores maintains the effectiveness. Furthermore, we introduced a number of complementary techniques to further improve the efficiency of Fast-Forward indexes:

- Early stopping (cf. Section 3.2.2) avoids unnecessary computations by terminating the ranking process early based on approximation of the maximum scores.
- Sequential coalescing (cf. Section 3.2.1) allows the compression of indexes that hold multiple representations of each document.
- Lightweight query encoders (cf. Section 3.3.1) make query encoding substantially faster, eliminating the bottleneck during online query processing, to the point where no neural forward pass is necessary anymore.
- Selective-BERT document encoders (cf. Section 3.3.2) learn to dynamically ignore irrelevant document tokens to improve indexing efficiency.

In Chapter 4, we introduced BERT-DMN and showed that utilizing sentence-level representations of large language models (here: BERT), instead of solely relying on the classification token output, can be beneficial for ranking. We further showed that LLMs do not necessarily have to be fine-tuned in order to be used for ranking; instead, an additional, much smaller, model can be trained on the sentence-level representations output by BERT. This has positive implications on training efficiency, as the LLM forward passes can be cached and thus only need to be computed once.

In Chapter 5, we introduced the Select-And-Rank paradigm for explainable-by-design ranking models. Select-And-Rank models have two components; a selector extracts a query-dependent explanation from the document, and a ranker computes the query-document relevance score only using the extracted explanation. We proposed an end-to-end approach to train the selector and ranker jointly. Our experiments showed that the ranking performance using only the explanation is similar to the ranking performance using the full document. Furthermore, we conducted experiments regarding

- the comprehensiveness of the extracted explanations (cf. Section 5.4.3), and
- the faithfulness of the extracted explanations according to a user study (cf. Section 5.4.4).

Finally, in Chapter 6, we introduced BoilerNet, a boilerplate removal approach for web content extraction. Our approach employs a supervised machine learning model and requires no engineering of hand-crafted features. Through evaluation on multiple datasets, we showed that BoilerNet is able to adapt to changing standards and web technologies. It can be used to clean web pages, removing all elements other than the main content, for example, to pre-process web corpora prior to the indexing step.

7.1.1 Software

The implementations of all models used in this thesis are open-source; the software contributions are listed below.

Fast-Forward Indexes

We provide a Python library that implements Fast-Forward indexes. Specifically, the following features are supported:

- Fast CPU-based ranking
- Index compression via sequential coalescing
- Early stopping for improved efficiency

The library is available via the Python package index (PyPI).[1] The source and API documentation can be found on GitHub.[2]

Ranking Utilities

The `ranking-utils` library provides utilities for training ranking models.[3] It offers the following features:

- Integration with the PyTorch Lightning framework[4]
- Dataset parsing and pre-processing
- Pointwise and pairwise training loss
- Validation using ranking metrics

[1] `pip install fast-forward-indexes`

[2] https://github.com/mrjleo/fast-forward-indexes

[3] https://github.com/mrjleo/ranking-utils

[4] https://www.pytorchlightning.ai/

Ranking Models

The `ranking-models` repository contains implementations of our ranking models (specifically, BERT-DMN and Select-And-Rank) as well as some baselines.[5] It supports training, validation, testing, and re-ranking.

Dual-Encoders for IR

The `dual-encoders` repository provides functionality to train and evaluate dual-encoder models for IR.[6] It supports a contrastive loss function; trained encoders may be used either for ANN retrieval or in combination with Fast-Forward indexes.

BoilerNet

The code for training and evaluation of BoilerNet is available on GitHub, along with the browser extension and links to pre-processed datasets.[7]

7.2 Future Work

Research in the field of neural ranking is advancing rapidly, and, with efficiency and explainability being more important than ever, there are numerous promising ways to extend the work in this thesis. In this section, we present several open research questions and ideas building upon our methods.

Fast-Forward indexes perform interpolation-based re-ranking. As a consequence, the recall, i.e., the fraction of all relevant documents that is captured, is determined by the first-stage retrieval model. Since we used a term-matching-based retriever (namely, BM25) in our experiments (cf. Section 3.5), our approach is, in principle, limited by the vocabulary mismatch problem (cf. Section 2.1.1). It is possible to mitigate this problem (to an extent) by increasing the first-stage retrieval depth and using pseudo relevance feedback and smoothing methods; in fact, we showed that re-ranking up to $k_S = 5000$ documents using Fast-Forward indexes is still fast. However, a natural next step would be to go beyond term matching in the first stage; one possibility would be to use a lightweight semantic retriever to capture semantically relevant documents in the first stage and a larger, more complex model for re-ranking in the second stage.

[5] `https://github.com/mrjleo/ranking-models`
[6] `https://github.com/mrjleo/dual-encoders`
[7] `https://github.com/mrjleo/boilernet`

In Section 3.3.1, we proposed lightweight query encoders for Fast-Forward indexes. We showed that these encoders can be orders of magnitude faster when self-attention is completely omitted, as the encoding operation boils down to a look-up table in that case. However, in some cases, this causes the performance to drop. We hypothesize that non-contextual subword representations are not sufficient for queries that are more complex than simple keywords; this issue might be alleviated by custom tokenizers trained to "mimic" contextualization by creating additional tokens that span multiple words instead of splitting them into subwords.

The Select-And-Rank models proposed in Chapter 5 are limited with respect to efficiency, as the explanations are query-dependent; in other words, since the input to the ranker is not known in advance, representations cannot be pre-computed, as is done for Fast-Forward indexes. A way around this limitation could be the use of more granular representations; for example, an approach like Sentence-BERT [167] could be used to compute sentence representations in advance and subsequently employ a dual-encoder architecture for re-ranking.

Finally, a recent line of work [81, 85] has found a potential problem of *select-and-predict* models, such as Select-And-Rank: If such models are trained jointly (end-to-end), unintended behavior may occur, where the predictor model learns to make its decision not based on what was selected, but rather on some artifacts introduced by the selector. Due to the fact that we used a lightweight selection model (S&R-LIN), we do not expect that our models exhibit this issue. Furthermore, our pipeline models (cf. Section 5.2.2) achieve performance similar to that of the end-to-end models. Nonetheless, this phenomenon should be further analyzed.

7.3 Outlook

In Fig. 1.4, we illustrated the trade-off between effectiveness, efficiency, and explainability of ranking models. The methods proposed in this thesis each focused on improving *either* efficiency (Fast-Forward indexes and BERT-DMN) *or* explainability (Select-And-Rank) while maintaining effectiveness; in a sense, the current approaches are still constrained to a maximum of two out of the three properties.

The next logical step is to go beyond this constraint and combine several aspects of the ideas proposed in this thesis in order to achieve *both* efficiency *and* explainability without sacrificing the effectiveness; more specifically, the Select-And-Rank paradigm should be adapted to exploit the efficiency of Fast-Forward indexes through pre-computation of representations rather than relying on the cross-attention architecture. Furthermore, lightweight training approaches, such as the one used by BERT-DMN$_{\text{lite}}$, should be employed in the offline phase in order to reduce carbon emissions.

The goal should be an end-to-end retrieval system that, even though it might not be able to entirely overcome the aforementioned trade-offs, provides both efficient and explainable ranking decisions, possibly allowing the user to control the trade-off by specifying their requirements with respect to the two aspects.

A

Curriculum Vitae

Personal Details

Full name	Lutz Jurek Leonhardt
Date of birth	September 13th, 1993
Place of birth	Hannover, Germany

Education

2017–2023	**PhD in computer science** *Leibniz University Hannover* Hannover, Germany Thesis: *Efficient and Explainable Neural Ranking*
2014–2017	**M. Sc. in computer science** *Leibniz University Hannover* Hannover, Germany Thesis: *Automatic Suggestion of Citation Markers in Wikipedia Articles*
2011–2014	**B. Sc. in computer science** *Leibniz University Hannover* Hannover, Germany Thesis: *Automatic Detection of the Acetabulum in 3D-Pelvis Models*
2011	**Abitur (high school)** *Kaiser Wilhelm- und Ratsgymnasium Hannover* Hannover, Germany

Work Experience

since July 2023	**Researcher** *Delft University of Technology* Delft, The Netherlands Research, mainly in the area of *information retrieval*
Sep. 2017– June 2023	**PhD student, researcher** *L3S Research Center, Leibniz University Hannover* Hannover, Germany Research and teaching, mainly in the area of *information retrieval*
Jul. 2016– Aug. 2017	**Working student** *Dassault Systèmes Deutschland GmbH* Hannover, Germany Software developer, working on tools mainly used by the quality assurance department for automated software testing
2012–2016, winter terms (Oct.–Jan.)	**Student assistant** *Human-Computer Interaction group, Leibniz University Hannover* Hannover, Germany Teaching assistant for the lecture *Programming I*: Grading assignments, teaching small groups of students, and supervising exams

Publications

Journal Articles

- Abhijit Anand, Jurek Leonhardt, Jaspreet Singh, Koustav Rudra, and Avishek Anand. “Data Augmentation for Sample Efficient and Robust Document Ranking”. In: *ACM Trans. Inf. Syst.* (Nov. 2023). Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3634911. URL: https://doi.org/10.1145/3634911

- Jurek Leonhardt, Henrik Müller, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. “Efficient Neural Ranking using Forward Indexes and Lightweight Encoders”. In: *ACM Trans. Inf. Syst.* (Nov. 2023). Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3631939. URL: https://doi.org/10.1145/3631939

- Jurek Leonhardt, Koustav Rudra, and Avishek Anand. “Extractive Explanations for

Interpretable Text Ranking". In: *ACM Trans. Inf. Syst.* 41.4 (Mar. 2023). ISSN: 1046-8188. DOI: 10.1145/3576924. URL: https://doi.org/10.1145/3576924

Full Conference Papers

- Abhijit Anand, Jurek Leonhardt, Koustav Rudra, and Avishek Anand. "Supervised Contrastive Learning Approach for Contextual Ranking". In: *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval.* ICTIR '22. Madrid, Spain: Association for Computing Machinery, 2022, pp. 61–71. ISBN: 9781450394123. DOI: 10.1145/3539813.3545139. URL: https://doi.org/10.1145/3539813.3545139

- Jurek Leonhardt, Koustav Rudra, Megha Khosla, Abhijit Anand, and Avishek Anand. "Efficient Neural Ranking using Forward Indexes". In: *Proceedings of the ACM Web Conference 2022.* WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 266–276. ISBN: 9781450390965. DOI: 10.1145/3485447.3511955. URL: https://doi.org/10.1145/3485447.3511955

- Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. "Node Representation Learning for Directed Graphs". In: *Machine Learning and Knowledge Discovery in Databases.* Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Cham: Springer International Publishing, 2020, pp. 395–411. ISBN: 978-3-030-46150-8

Short and Demonstration Papers

- Jurek Leonhardt, Avishek Anand, and Megha Khosla. "Boilerplate Removal using a Neural Sequence Labeling Model". In: *Companion Proceedings of the Web Conference 2020.* WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 226–229. ISBN: 9781450370240. DOI: 10.1145/3366424.3383547. URL: https://doi.org/10.1145/3366424.3383547

- Jurek Leonhardt, Avishek Anand, and Megha Khosla. "User Fairness in Recommender Systems". In: *Companion Proceedings of the The Web Conference 2018.* WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 101–102. ISBN: 9781450356404. DOI: 10.1145/3184558.3186949. URL: https://doi.org/10.1145/3184558.3186949

Workshop Papers

- Jurek Leonhardt, Fabian Beringer, and Avishek Anand. "Exploiting Sentence-Level Representations for Passage Ranking". In: *Proceedings of the LWDA 2021 Workshops: FGWM, KDML, FGWI-BIA, and FGIR, Online, September 1-3, 2021.* Ed. by Thomas Seidl, Michael Fromm, and Sandra Obermeier. Vol. 2993. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 287–302. URL: https://ceur-ws.org/Vol-2993/paper-27.pdf

Preprints

- Jurek Leonhardt, Marcel Jahnke, and Avishek Anand. *Distribution-Aligned Fine-Tuning for Efficient Neural Retrieval.* 2022. arXiv: 2211.04942 [cs.IR]

Bibliography

[1] Nasreen Abdul-Jaleel et al. “UMass at TREC 2004: Novelty and HARD”. In: *Computer Science Department Faculty Publication Series* (2004), p. 189.

[2] Julius Adebayo et al. “Debugging Tests for Model Explanations”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 700–712. URL: https://proceedings.neurips.cc/paper/2020/file/075b051ec3d22dac7b33f788da631fd4-Paper.pdf.

[3] Betty van Aken et al. “How Does BERT Answer Questions? A Layer-Wise Analysis of Transformer Representations”. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM ’19. Beijing, China: Association for Computing Machinery, 2019, pp. 1823–1832. ISBN: 9781450369763. DOI: 10.1145/3357384.3358028. URL: https://doi.org/10.1145/3357384.3358028.

[4] Zeynep Akkalyoncu Yilmaz et al. “Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3490–3496. DOI: 10.18653/v1/D19-1352. URL: https://aclanthology.org/D19-1352.

[5] Sophia Althammer et al. “PARM: A Paragraph Aggregation Retrieval Model for Dense Document-to-Document Retrieval”. In: *Advances in Information Retrieval*. Ed. by Matthias Hagen et al. Cham: Springer International Publishing, 2022, pp. 19–34. ISBN: 978-3-030-99736-6.

[6] Abhijit Anand et al. “Data Augmentation for Sample Efficient and Robust Document Ranking”. In: *ACM Trans. Inf. Syst.* (Nov. 2023). Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3634911. URL: https://doi.org/10.1145/3634911.

[7] Abhijit Anand et al. “Supervised Contrastive Learning Approach for Contextual Ranking”. In: *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR ’22. Madrid, Spain: Association for Computing Machin-

ery, 2022, pp. 61–71. ISBN: 9781450394123. DOI: 10.1145/3539813.3545139. URL: https://doi.org/10.1145/3539813.3545139.

[8] Avishek Anand et al. "Conversational search (dagstuhl seminar 19461)". In: *Dagstuhl Reports*. Vol. 9. 11. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2020.

[9] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Sy21R9JAW.

[10] Arian Askari et al. "Injecting the BM25 Score as Text Improves BERT-Based Rerankers". In: *Advances in Information Retrieval*. Ed. by Jaap Kamps et al. Cham: Springer Nature Switzerland, 2023, pp. 66–83. ISBN: 978-3-031-28244-7.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[12] Ziv Bar-Yossef and Sridhar Rajagopalan. "Template Detection via Data Mining and Its Applications". In: *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. Honolulu, Hawaii, USA: Association for Computing Machinery, 2002, pp. 580–591. ISBN: 1581134495. DOI: 10.1145/511446.511522. URL: https://doi.org/10.1145/511446.511522.

[13] Marco Baroni et al. "CleanEval: a Competition for Cleaning Web Pages." In: *LREC*. 2008.

[14] Jasmijn Bastings, Wilker Aziz, and Ivan Titov. "Interpretable Neural Predictions with Differentiable Binary Variables". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2963–2977. DOI: 10.18653/v1/P19-1284. URL: https://aclanthology.org/P19-1284.

[15] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. 2013. DOI: 10.48550/ARXIV.1308.3432. URL: https://arxiv.org/abs/1308.3432.

[16] Andrei Z. Broder et al. "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, 2003, pp. 426–434. ISBN: 1581137230. DOI: 10.1145/956863.956944. URL: https://doi.org/10.1145/956863.956944.

[17] Sebastian Bruch, Siyu Gai, and Amir Ingber. “An Analysis of Fusion Functions for Hybrid Retrieval”. In: *ACM Trans. Inf. Syst.* 42.1 (Aug. 2023). ISSN: 1046-8188. DOI: 10.1145/3596512. URL: https://doi.org/10.1145/3596512.

[18] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. “Report on the 1st Workshop on Reaching Efficiency in Neural Information Retrieval (ReNeuIR 2022) at SIGIR 2022”. In: *SIGIR Forum* 56.2 (Jan. 2023). ISSN: 0163-5840. DOI: 10.1145/3582900.3582916. URL: https://doi.org/10.1145/3582900.3582916.

[19] Chris Burges et al. “Learning to Rank Using Gradient Descent”. In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML ’05. Bonn, Germany: Association for Computing Machinery, 2005, pp. 89–96. ISBN: 1595931805. DOI: 10.1145/1102351.1102363. URL: https://doi.org/10.1145/1102351.1102363.

[20] Deng Cai et al. “Block-Based Web Search”. 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, 2004, pp. 456–463. ISBN: 1581138814. DOI: 10.1145/1008992.1009070. URL: https://doi.org/10.1145/1008992.1009070.

[21] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. “Page-Level Template Detection via Isotonic Smoothing”. In: *Proceedings of the 16th International Conference on World Wide Web*. WWW ’07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 61–70. ISBN: 9781595936547. DOI: 10.1145/1242572.1242582. URL: https://doi.org/10.1145/1242572.1242582.

[22] Abhijnan Chakraborty et al. “Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media”. In: *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ASONAM ’16. Davis, California: IEEE Press, 2016, pp. 9–16. ISBN: 9781509028467.

[23] Wei-Cheng Chang et al. “Pre-training Tasks for Embedding-based Large-scale Retrieval”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=rkg-mA4FDr.

[24] Daniel L. Chen, Martin Schonger, and Chris Wickens. “oTree—An open-source platform for laboratory, online, and field experiments”. In: *Journal of Behavioral and Experimental Finance* 9 (2016), pp. 88–97. ISSN: 2214-6350. DOI: https://doi.org/10.1016/j.jbef.2015.12.001. URL: https://www.sciencedirect.com/science/article/pii/S2214635016000101.

[25] Danqi Chen et al. “Reading Wikipedia to Answer Open-Domain Questions”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1870–1879. DOI: 10.18653/v1/P17-1171. URL: https://aclanthology.org/P17-1171.

[26] Xiaoyang Chen et al. “Co-BERT: A Context-Aware BERT Retrieval Model Incorporating Local and Query-specific Context”. In: *arXiv preprint arXiv:2104.08523* (2021).

[27] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 551–561. DOI: 10.18653/v1/D16-1053. URL: https://aclanthology.org/D16-1053.

[28] Eunseong Choi et al. “SpaDE: Improving Sparse Representations Using a Dual Document Encoder for First-Stage Retrieval”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM ’22. Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 272–282. ISBN: 9781450392365. DOI: 10.1145/3511808.3557456. URL: https://doi.org/10.1145/3511808.3557456.

[29] Nachshon Cohen et al. “SDR: Efficient Neural Re-ranking using Succinct Document Representation”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6624–6637. DOI: 10.18653/v1/2022.acl-long.457. URL: https://aclanthology.org/2022.acl-long.457.

[30] Nick Craswell et al. “ORCAS: 20 Million Clicked Query-Document Pairs for Analyzing Search”. In: *Proceedings of the 29th ACM International Conference on Information &amp; Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, pp. 2983–2989. ISBN: 9781450368599. DOI: 10.1145/3340531.3412779. URL: https://doi.org/10.1145/3340531.3412779.

[31] Nick Craswell et al. “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, 2021, pp. 2369–2375. ISBN: 9781450380379. DOI: 10.1145/3404835.3463249. URL: https://doi.org/10.1145/3404835.3463249.

[32] Yiming Cui et al. "Attention-over-Attention Neural Networks for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 593–602. DOI: 10.18653/v1/P17-1055. URL: https://aclanthology.org/P17-1055.

[33] Zhuyun Dai and Jamie Callan. "An Evaluation of Weakly-Supervised DeepCT in the TREC 2019 Deep Learning Track". In: *TREC*. 2019.

[34] Zhuyun Dai and Jamie Callan. "Context-Aware Document Term Weighting for Ad-Hoc Search". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 1897–1907. ISBN: 9781450370233. DOI: 10.1145/3366423.3380258. URL: https://doi.org/10.1145/3366423.3380258.

[35] Zhuyun Dai and Jamie Callan. "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, 2020, pp. 1533–1536. ISBN: 9781450380164. DOI: 10.1145/3397271.3401204. URL: https://doi.org/10.1145/3397271.3401204.

[36] Zhuyun Dai and Jamie Callan. "Deeper Text Understanding for IR with Contextual Neural Language Modeling". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, 2019, pp. 985–988. ISBN: 9781450361729. DOI: 10.1145/3331184.3331303. URL: https://doi.org/10.1145/3331184.3331303.

[37] Zhuyun Dai et al. "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, 2018, pp. 126–134. ISBN: 9781450355810. DOI: 10.1145/3159652.3159659. URL: https://doi.org/10.1145/3159652.3159659.

[38] Mostafa Dehghani et al. "Fidelity-Weighted Learning". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=B1X0mzZCW.

[39] Mostafa Dehghani et al. "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 Com-

puting Machinery, 2017, pp. 65–74. ISBN: 9781450350228. DOI: 10.1145/3077136.3080832. URL: https://doi.org/10.1145/3077136.3080832.

[40] Jacob Devlin et al. "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, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[41] Jay DeYoung et al. "ERASER: A Benchmark to Evaluate Rationalized NLP Models". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Online: Association for Computational Linguistics, July 2020, pp. 4443–4458. DOI: 10.18653/v1/2020.acl-main.408. URL: https://aclanthology.org/2020.acl-main.408.

[42] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. "Query Expansion with Locally-Trained Word Embeddings". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 367–377. DOI: 10.18653/v1/P16-1035. URL: https://aclanthology.org/P16-1035.

[43] Sibo Dong, Justin Goldstein, and Grace Hui Yang. "SEINE: SEgment-based Indexing for NEural information retrieval". In: (2022).

[44] Ronald Fagin, Amnon Lotem, and Moni Naor. "Optimal Aggregation Algorithms for Middleware". In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems.* PODS '01. Santa Barbara, California, USA: Association for Computing Machinery, 2001, pp. 102–113. ISBN: 1581133618. DOI: 10.1145/375551.375567. URL: https://doi.org/10.1145/375551.375567.

[45] Zhen Fan et al. "COILcr: Efficient Semantic Matching in Contextualized Exact Match Retrieval". In: *Advances in Information Retrieval.* Ed. by Jaap Kamps et al. Cham: Springer Nature Switzerland, 2023, pp. 298–312. ISBN: 978-3-031-28244-7.

[46] Minwei Feng et al. "Applying deep learning to answer selection: A study and an open task". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).* 2015, pp. 813–820. DOI: 10.1109/ASRU.2015.7404872.

[47] Zeon Trevor Fernando, Jaspreet Singh, and Avishek Anand. “A Study on the Interpretability of Neural Retrieval Models Using DeepSHAP”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR’19. Paris, France: Association for Computing Machinery, 2019, pp. 1005–1008. ISBN: 9781450361729. DOI: 10.1145/3331184.3331312. URL: https://doi.org/10.1145/3331184.3331312.

[48] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. “Fact or Fiction: Content Classification for Digital Libraries”. In: *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries, DELOS 2001, Dublin, Ireland, June 18-20, 2001.* Ed. by Alan F. Smeaton and Jamie Callan. Vol. 01/W03. ERCIM Workshop Proceedings. ERCIM, 2001. URL: http://www.ercim.org/publication/ws-proceedings/DelNoe02/AidanFinn.pdf.

[49] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. “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. New York, NY, USA: Association for Computing Machinery, 2021, pp. 2288–2292. ISBN: 9781450380379. DOI: 10.1145/3404835.3463098. URL: https://doi.org/10.1145/3404835.3463098.

[50] Thorben Funke et al. “ZORRO: Valid, Sparse, and Stable Explanations in Graph Neural Networks”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–12. DOI: 10.1109/TKDE.2022.3201170.

[51] Luke Gallagher. *Pairwise t-test on TREC Run Files.* https://github.com/lgrz/pairwise-ttest/. 2019.

[52] Luyu Gao and Jamie Callan. “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, Nov. 2021, pp. 981–993. DOI: 10.18653/v1/2021.emnlp-main.75. URL: https://aclanthology.org/2021.emnlp-main.75.

[53] Luyu Gao and Jamie Callan. “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, May 2022, pp. 2843–2853. DOI: 10.18653/v1/2022.acl-long.203. URL: https://aclanthology.org/2022.acl-long.203.

[54] Luyu Gao, Zhuyun Dai, and Jamie Callan. “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, June 2021, pp. 3030–3042. DOI: 10.18653/v1/2021.naacl-main.241. URL: https://aclanthology.org/2021.naacl-main.241.

[55] Luyu Gao et al. “Complement Lexical Retrieval Model with Semantic Residual Embeddings”. In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra et al. Cham: Springer International Publishing, 2021, pp. 146–160. ISBN: 978-3-030-72113-8.

[56] Ayçe Geçkil et al. “A Clickbait Detection Method on News Sites”. In: *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ASONAM ’18. Barcelona, Spain: IEEE Press, 2020, pp. 932–937. ISBN: 9781538660515.

[57] John Gibson, Ben Wellner, and Susan Lubar. “Adaptive Web-Page Content Identification”. In: *Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management*. WIDM ’07. Lisbon, Portugal: Association for Computing Machinery, 2007, pp. 105–112. ISBN: 9781595938299. DOI: 10.1145/1316902.1316920. URL: https://doi.org/10.1145/1316902.1316920.

[58] Leilani H. Gilpin et al. “Explaining Explanations: An Overview of Interpretability of Machine Learning”. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018, pp. 80–89. DOI: 10.1109/DSAA.2018.00018.

[59] Saurabh Goyal et al. “PoWER-BERT: Accelerating BERT Inference via Progressive Word-Vector Elimination”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org, 2020.

[60] Jiafeng Guo et al. “A Deep Relevance Matching Model for Ad-Hoc Retrieval”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM ’16. Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, pp. 55–64. ISBN: 9781450340731. DOI: 10.1145/2983323.2983769. URL: https://doi.org/10.1145/2983323.2983769.

[61] Weiwei Guo et al. “DeText: A Deep Text Ranking Framework with BERT”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, pp. 2509–2516. ISBN: 9781450368599. DOI: 10.1145/3340531.3412699. URL: https://doi.org/10.1145/3340531.3412699.

[62] Yue Guo, Yi Yang, and Ahmed Abbasi. "Auto-Debias: Debiasing Masked Language Models with Automated Biased Prompts". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1012–1023. DOI: 10.18653/v1/2022.acl-long.72. URL: https://aclanthology.org/2022.acl-long.72.

[63] Suhit Gupta et al. "DOM-Based Content Extraction of HTML Documents". In: *Proceedings of the 12th International Conference on World Wide Web*. WWW '03. Budapest, Hungary: Association for Computing Machinery, 2003, pp. 207–214. ISBN: 1581136803. DOI: 10.1145/775152.775182. URL: https://doi.org/10.1145/775152.775182.

[64] Vishal Gupta, Manoj Chinnakotla, and Manish Shrivastava. "Retrieve and Re-rank: A Simple and Effective IR Approach to Simple Question Answering over Knowledge Graphs". In: *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 22–27. DOI: 10.18653/v1/W18-5504. URL: https://aclanthology.org/W18-5504.

[65] Helia Hashemi et al. "ANTIQUE: A Non-factoid Question Answering Benchmark". In: *Advances in Information Retrieval*. Ed. by Joemon M. Jose et al. Cham: Springer International Publishing, 2020, pp. 166–173. ISBN: 978-3-030-45442-5.

[66] Michiel Hermans and Benjamin Schrauwen. "Training and Analysing Deep Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/1ff8a7b5dc7a7d1f0ed65aaa29c04b1e-Paper.pdf.

[67] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. "Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking". In: *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*. Ed. by Giuseppe De Giacomo et al. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, 2020, pp. 513–520. DOI: 10.3233/FAIA200133. URL: https://doi.org/10.3233/FAIA200133.

[68] Sebastian Hofstätter et al. *Are We There Yet? A Decision Framework for Replacing Term Based Retrieval with Dense Retrieval Systems*. 2022. DOI: 10.48550/ARXIV.2206.12993. URL: https://arxiv.org/abs/2206.12993.

[69] Sebastian Hofstätter et al. "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.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 113–122. ISBN: 9781450380379. URL: https://doi.org/10.1145/3404835.3462891.

[70] Sebastian Hofstätter et al. "Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 1349–1358. ISBN: 9781450380379. DOI: 10.1145/3404835.3462889. URL: https://doi.org/10.1145/3404835.3462889.

[71] Sebastian Hofstätter et al. "Introducing Neural Bag of Whole-Words with ColBERTer: Contextualized Late Interactions Using Enhanced Reduction". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 737–747. ISBN: 9781450392365. DOI: 10.1145/3511808.3557367. URL: https://doi.org/10.1145/3511808.3557367.

[72] Sebastian Hofstätter et al. "Local Self-Attention over Long Text for Efficient Document 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, 2020, pp. 2021–2024. ISBN: 9781450380164. DOI: 10.1145/3397271.3401224. URL: https://doi.org/10.1145/3397271.3401224.

[73] Sebastian Hofstätter et al. "Mitigating the Position Bias of Transformer Models in Passage Re-Ranking". 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, 2021, pp. 238–253. ISBN: 978-3-030-72112-1. DOI: 10.1007/978-3-030-72113-8_16. URL: https://doi.org/10.1007/978-3-030-72113-8_16.

[74] Helge Holzmann and Avishek Anand. "Tempas: Temporal Archive Search Based on Tags". In: *Proceedings of the 25th International Conference Companion on World Wide Web.* WWW '16 Companion. Montréal, Québec, Canada: International World Wide

Web Conferences Steering Committee, 2016, pp. 207–210. ISBN: 9781450341448. DOI: 10.1145/2872518.2890555. URL: https://doi.org/10.1145/2872518.2890555.

[75] Helge Holzmann, Wolfgang Nejdl, and Avishek Anand. "Exploring Web Archives Through Temporal Anchor Texts". In: *Proceedings of the 2017 ACM on Web Science Conference*. WebSci '17. Troy, New York, USA: Association for Computing Machinery, 2017, pp. 289–298. ISBN: 9781450348966. DOI: 10.1145/3091478.3091500. URL: https://doi.org/10.1145/3091478.3091500.

[76] Helge Holzmann, Wolfgang Nejdl, and Avishek Anand. "The Dawn of Today's Popular Domains: A Study of the Archived German Web over 18 Years". In: *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. JCDL '16. Newark, New Jersey, USA: Association for Computing Machinery, 2016, pp. 73–82. ISBN: 9781450342292. DOI: 10.1145/2910896.2910901. URL: https://doi.org/10.1145/2910896.2910901.

[77] Po-Sen Huang et al. "Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data". In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. CIKM '13. San Francisco, California, USA: Association for Computing Machinery, 2013, pp. 2333–2338. ISBN: 9781450322638. DOI: 10.1145/2505515.2505665. URL: https://doi.org/10.1145/2505515.2505665.

[78] Kai Hui et al. "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, 2018, pp. 279–287. ISBN: 9781450355810. DOI: 10.1145/3159652.3159689. URL: https://doi.org/10.1145/3159652.3159689.

[79] Kai Hui et al. "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, Sept. 2017, pp. 1049–1058. DOI: 10.18653/v1/D17-1110. URL: https://aclanthology.org/D17-1110.

[80] Maximilian Idahl et al. "Towards Benchmarking the Utility of Explanations for Model Debugging". In: *Proceedings of the First Workshop on Trustworthy Natural Language Processing*. Online: Association for Computational Linguistics, June 2021, pp. 68–73. DOI: 10.18653/v1/2021.trustnlp-1.8. URL: https://aclanthology.org/2021.trustnlp-1.8.

[81] Alon Jacovi and Yoav Goldberg. "Aligning Faithful Interpretations with their Social Attribution". In: *Transactions of the Association for Computational Linguistics* 9 (Mar. 2021), pp. 294–310. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00367. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00367/1923972/tacl_a_00367.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00367.

[82] Alon Jacovi and Yoav Goldberg. "Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4198–4205. DOI: 10.18653/v1/2020.acl-main.386. URL: https://aclanthology.org/2020.acl-main.386.

[83] Sarthak Jain and Byron C. Wallace. "Attention is not Explanation". 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, June 2019, pp. 3543–3556. DOI: 10.18653/v1/N19-1357. URL: https://aclanthology.org/N19-1357.

[84] Kyoung-Rok Jang et al. "Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1016–1029. DOI: 10.18653/v1/2021.emnlp-main.78. URL: https://aclanthology.org/2021.emnlp-main.78.

[85] Neil Jethani et al. "Have We Learned to Explain?: How Interpretability Methods Can Learn to Encode Predictions in their Interpretations." In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 13–15 Apr 2021, pp. 1459–1467. URL: https://proceedings.mlr.press/v130/jethani21a.html.

[86] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-Scale Similarity Search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (2021), pp. 535–547. DOI: 10.1109/TBDATA.2019.2921572.

[87] Euna Jung, Jaekeol Choi, and Wonjong Rhee. "Semi-Siamese Bi-Encoder Neural Ranking Model Using Lightweight Fine-Tuning". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Ma-

chinery, 2022, pp. 502–511. ISBN: 9781450390965. DOI: 10.1145/3485447.3511978. URL: https://doi.org/10.1145/3485447.3511978.

[88] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. *Visualizing and Understanding Recurrent Networks*. 2015. DOI: 10.48550/ARXIV.1506.02078. URL: https://arxiv.org/abs/1506.02078.

[89] Vladimir Karpukhin et al. “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, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://aclanthology.org/2020.emnlp-main.550.

[90] Omar Khattab and Matei Zaharia. “ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 39–48. ISBN: 9781450380164. URL: https://doi.org/10.1145/3397271.3401075.

[91] Megha Khosla et al. “Node Representation Learning for Directed Graphs”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld et al. Cham: Springer International Publishing, 2020, pp. 395–411. ISBN: 978-3-030-46150-8.

[92] Seungyeon Kim et al. *EmbedDistill: A Geometric Knowledge Distillation for Information Retrieval*. 2023. DOI: 10.48550/ARXIV.2301.12005. URL: https://arxiv.org/abs/2301.12005.

[93] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. “Boilerplate Detection Using Shallow Text Features”. In: *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. WSDM ’10. New York, New York, USA: Association for Computing Machinery, 2010, pp. 441–450. ISBN: 9781605588896. DOI: 10.1145/1718487.1718542. URL: https://doi.org/10.1145/1718487.1718542.

[94] Bernhard Kratzwald and Stefan Feuerriegel. “Adaptive Document Retrieval for Deep Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 576–581. DOI: 10.18653/v1/D18-1055. URL: https://aclanthology.org/D18-1055.

[95] Ankit Kumar et al. "Ask Me Anything: Dynamic Memory Networks for Natural Language Processing". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1378–1387. URL: https://proceedings.mlr.press/v48/kumar16.html.

[96] Cody Kwok, Oren Etzioni, and Daniel S. Weld. "Scaling Question Answering to the Web". In: *ACM Trans. Inf. Syst.* 19.3 (July 2001), pp. 242–262. ISSN: 1046-8188. DOI: 10.1145/502115.502117. URL: https://doi.org/10.1145/502115.502117.

[97] John Lafferty and Chengxiang Zhai. "Document Language Models, Query Models, and Risk Minimization for Information Retrieval". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 111–119. ISBN: 1581133316. DOI: 10.1145/383952.383970. URL: https://doi.org/10.1145/383952.383970.

[98] Isaac Lage et al. *An Evaluation of the Human-Interpretability of Explanation.* 2019. DOI: 10.48550/ARXIV.1902.00006. URL: https://arxiv.org/abs/1902.00006.

[99] Carlos Lassance and Stéphane Clinchant. "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, 2022, pp. 2220–2226. ISBN: 9781450387323. DOI: 10.1145/3477495.3531833. URL: https://doi.org/10.1145/3477495.3531833.

[100] Carlos Lassance, Hervé Dejean, and Stéphane Clinchant. "An Experimental Study on Pretraining Transformers from Scratch for IR". In: *Advances in Information Retrieval.* Ed. by Jaap Kamps et al. Cham: Springer Nature Switzerland, 2023, pp. 504–520. ISBN: 978-3-031-28244-7.

[101] Carlos Lassance et al. "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, 2022, pp. 2232–2236. ISBN: 9781450387323. DOI: 10.1145/3477495.3531835. URL: https://doi.org/10.1145/3477495.3531835.

[102] Victor Lavrenko and W. Bruce Croft. "Relevance Based Language Models". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '01. New York, NY, USA: ACM, 2001, pp. 120–

127. ISBN: 1-58113-331-6. DOI: 10.1145/383952.383972. URL: http://doi.acm.org/10.1145/383952.383972.

[103] Eric Lehman et al. "Inferring Which Medical Treatments Work from Reports of Clinical Trials". 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, June 2019, pp. 3705–3717. DOI: 10.18653/v1/N19-1371. URL: https://aclanthology.org/N19-1371.

[104] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Rationalizing Neural Predictions". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 107–117. DOI: 10.18653/v1/D16-1011. URL: https://aclanthology.org/D16-1011.

[105] Jurek Leonhardt, Avishek Anand, and Megha Khosla. "Boilerplate Removal using a Neural Sequence Labeling Model". In: *Companion Proceedings of the Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 226–229. ISBN: 9781450370240. DOI: 10.1145/3366424.3383547. URL: https://doi.org/10.1145/3366424.3383547.

[106] Jurek Leonhardt, Avishek Anand, and Megha Khosla. "User Fairness in Recommender Systems". In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 101–102. ISBN: 9781450356404. DOI: 10.1145/3184558.3186949. URL: https://doi.org/10.1145/3184558.3186949.

[107] Jurek Leonhardt, Fabian Beringer, and Avishek Anand. "Exploiting Sentence-Level Representations for Passage Ranking". In: *Proceedings of the LWDA 2021 Workshops: FGWM, KDML, FGWI-BIA, and FGIR, Online, September 1-3, 2021*. Ed. by Thomas Seidl, Michael Fromm, and Sandra Obermeier. Vol. 2993. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 287–302. URL: https://ceur-ws.org/Vol-2993/paper-27.pdf.

[108] Jurek Leonhardt, Marcel Jahnke, and Avishek Anand. *Distribution-Aligned Fine-Tuning for Efficient Neural Retrieval*. 2022. arXiv: 2211.04942 [cs.IR].

[109] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. "Extractive Explanations for Interpretable Text Ranking". In: *ACM Trans. Inf. Syst.* 41.4 (Mar. 2023). ISSN: 1046-8188. DOI: 10.1145/3576924. URL: https://doi.org/10.1145/3576924.

[110] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. *L3S at the TREC 2021 Deep Learning Track.* 2021.

[111] Jurek Leonhardt et al. "Efficient Neural Ranking using Forward Indexes". In: *Proceedings of the ACM Web Conference 2022.* WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 266–276. ISBN: 9781450390965. DOI: 10.1145/3485447.3511955. URL: https://doi.org/10.1145/3485447.3511955.

[112] Jurek Leonhardt et al. "Efficient Neural Ranking using Forward Indexes and Lightweight Encoders". In: *ACM Trans. Inf. Syst.* (Nov. 2023). Just Accepted. ISSN: 1046-8188. DOI: 10.1145/3631939. URL: https://doi.org/10.1145/3631939.

[113] Canjia Li et al. "PARADE: Passage Representation Aggregation For Document Reranking". In: *ACM Trans. Inf. Syst.* 42.2 (Sept. 2023). ISSN: 1046-8188. DOI: 10.1145/3600088. URL: https://doi.org/10.1145/3600088.

[114] Jiwei Li, Will Monroe, and Dan Jurafsky. *Understanding Neural Networks through Representation Erasure.* 2016. DOI: 10.48550/ARXIV.1612.08220. URL: https://arxiv.org/abs/1612.08220.

[115] Minghan Li and Eric Gaussier. "KeyBLD: Selecting Key Blocks with Local Pre-Ranking for Long Document Information Retrieval". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 2207–2211. ISBN: 9781450380379. DOI: 10.1145/3404835.3463083. URL: https://doi.org/10.1145/3404835.3463083.

[116] Minghan Li et al. *CITADEL: Conditional Token Interaction via Dynamic Lexical Routing for Efficient and Effective Multi-Vector Retrieval.* 2022. DOI: 10.48550/ARXIV.2211.10411. URL: https://arxiv.org/abs/2211.10411.

[117] Minghan Li et al. "The Power of Selecting Key Blocks with Local Pre-Ranking for Long Document Information Retrieval". In: *ACM Trans. Inf. Syst.* 41.3 (Feb. 2023). ISSN: 1046-8188. DOI: 10.1145/3568394. URL: https://doi.org/10.1145/3568394.

[118] Xiangsheng Li et al. "Teach Machine How to Read: Reading Behavior Inspired Relevance Estimation". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR'19. Paris, France: Association for Computing Machinery, 2019, pp. 795–804. ISBN: 9781450361729. DOI: 10.1145/3331184.3331205. URL: https://doi.org/10.1145/3331184.3331205.

[119] Jimmy Lin and Xueguang Ma. *A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques*. 2021. arXiv: 2106.14807 [cs.IR].

[120] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2021. DOI: 10.2200/S01123ED1V01Y202108HLT053. URL: https://doi.org/10.2200/S01123ED1V01Y202108HLT053.

[121] Jimmy Lin et al. "Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 2356–2362. ISBN: 9781450380379. URL: https://doi.org/10.1145/3404835.3463238.

[122] Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. "Aggretriever: A Simple Approach to Aggregate Textual Representations for Robust Dense Passage Retrieval". In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 436–452. DOI: 10.1162/tacl_a_00556. URL: https://aclanthology.org/2023.tacl-1.26.

[123] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. "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, Aug. 2021, pp. 163–173. DOI: 10.18653/v1/2021.repl4nlp-1.17. URL: https://aclanthology.org/2021.repl4nlp-1.17.

[124] Shian-Hua Lin and Jan-Ming Ho. "Discovering Informative Content Blocks from Web Documents". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, 2002, pp. 588–593. ISBN: 158113567X. DOI: 10.1145/775047.775134. URL: https://doi.org/10.1145/775047.775134.

[125] Yankai Lin et al. "Denoising Distantly Supervised Open-Domain Question Answering". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 1736–1745. DOI: 10.18653/v1/P18-1161. URL: https://aclanthology.org/P18-1161.

[126] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable AI: A Review of Machine Learning Interpretability Methods". In: *Entropy* 23.1 (2021). ISSN: 1099-4300. DOI: 10.3390/e23010018. URL: https://www.mdpi.com/1099-4300/23/1/18.

[127] Erik Lindgren et al. "Efficient Training of Retrieval Models using Negative Cache". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 4134–4146. URL: https://proceedings.neurips.cc/paper/2021/file/2175f8c5cd9604f6b1e576b252d4c86e-Paper.pdf.

[128] Chang Liu et al. *Adam: Dense Retrieval Distillation with Adaptive Dark Examples*. 2022. DOI: 10.48550/ARXIV.2212.10192. URL: https://arxiv.org/abs/2212.10192.

[129] Tie-Yan Liu. "Learning to Rank for Information Retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3 (2009), pp. 225–331. ISSN: 1554-0669. DOI: 10.1561/1500000016. URL: http://dx.doi.org/10.1561/1500000016.

[130] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[131] Yi Luan et al. "Sparse, Dense, and Attentional Representations for Text Retrieval". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 329–345. DOI: 10.1162/tacl_a_00369. URL: https://aclanthology.org/2021.tacl-1.20.

[132] Julia Luxenburger, Shady Elbassuoni, and Gerhard Weikum. "Matching Task Profiles and User Needs in Personalized Web Search". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 689–698. ISBN: 9781595939913. DOI: 10.1145/1458082.1458175. URL: https://doi.org/10.1145/1458082.1458175.

[133] Xinyu Ma et al. "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, 2021, pp. 283–291. ISBN: 9781450382977. DOI: 10.1145/3437963.3441777. URL: https://doi.org/10.1145/3437963.3441777.

[134] Sean MacAvaney et al. "CEDR: Contextualized Embeddings for Document Ranking". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1101–1104. ISBN: 9781450361729. DOI: 10.1145/3331184.3331317. URL: https://doi.org/10.1145/3331184.3331317.

[135] Sean MacAvaney et al. "Expansion via Prediction of Importance with Contextualization". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: Association for Computing Machinery, 2020, pp. 1573–1576. ISBN: 9781450380164. URL: https://doi.org/10.1145/3397271.3401262.

[136] Joel Mackenzie et al. "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, 2020, pp. 1821–1824. ISBN: 9781450380164. DOI: 10.1145/3397271.3401263. URL: https://doi.org/10.1145/3397271.3401263.

[137] Chris J Maddison, Daniel Tarlow, and Tom Minka. "A* Sampling". In: *Advances in Neural Information Processing Systems.* Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/309fee4e541e51de2e41f21bebb342aa-Paper.pdf.

[138] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *International Conference on Learning Representations.* 2017. URL: https://openreview.net/forum?id=S1jE5L5gl.

[139] Yu A. Malkov and D. A. Yashunin. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2020), pp. 824–836. DOI: 10.1109/TPAMI.2018.2889473.

[140] Antonio Mallia et al. "Learning Passage Impacts for Inverted Indexes". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 1723–1727. ISBN: 9781450380379. URL: https://doi.org/10.1145/3404835.3463030.

[141] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval.* Cambridge university press, 2009.

[142] Andre Martins and Ramon Astudillo. "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification". In: *Proceedings of The 33rd International Conference on Machine Learning.* Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York,

USA: PMLR, 20–22 Jun 2016, pp. 1614–1623. URL: https://proceedings.mlr.press/v48/martins16.html.

[143] Pascal Massart. "The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality". In: *The Annals of Probability* 18.3 (1990), pp. 1269–1283. ISSN: 00911798. URL: http://www.jstor.org/stable/2244426.

[144] Irina Matveeva et al. "High Accuracy Retrieval with Multiple Nested Ranker". In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '06. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 437–444. ISBN: 1595933697. DOI: 10.1145/1148170.1148246. URL: https://doi.org/10.1145/1148170.1148246.

[145] Ryan McDonald, George Brokos, and Ion Androutsopoulos. "Deep Relevance Ranking Using Enhanced Document-Query Interactions". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1849–1860. DOI: 10.18653/v1/D18-1211. URL: https://aclanthology.org/D18-1211.

[146] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems.* Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[147] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. "Learning to Match Using Local and Distributed Representations of Text for Web Search". In: *Proceedings of the 26th International Conference on World Wide Web.* WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 1291–1299. ISBN: 9781450349130. DOI: 10.1145/3038912.3052579. URL: https://doi.org/10.1145/3038912.3052579.

[148] Bhaskar Mitra et al. *A Dual Embedding Space Model for Document Ranking*. 2016. arXiv: 1602.01137 [cs.IR].

[149] Bhaskar Mitra et al. "Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks". In: *arXiv preprint arXiv:1907.03693* (2019).

[150] Cristina Ioana Muntean et al. "Weighting Passages Enhances Accuracy". In: *ACM Trans. Inf. Syst.* 39.2 (Dec. 2020). ISSN: 1046-8188. DOI: 10.1145/3428687. URL: https://doi.org/10.1145/3428687.

[151] Tri Nguyen et al. “MS MARCO: A Human Generated MAchine Reading COmprehension Dataset”. In: *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016.* Ed. by Tarek Richard Besold et al. Vol. 1773. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1773/CoCoNIPS%5C_2016%5C_paper9.pdf.

[152] Jianmo Ni et al. “Large Dual Encoders Are Generalizable Retrievers”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 9844–9855. DOI: 10.18653/v1/2022.emnlp-main.669. URL: https://aclanthology.org/2022.emnlp-main.669.

[153] Yifan Nie, Yanling Li, and Jian-Yun Nie. “Empirical Study of Multi-Level Convolution Models for IR Based on Representations and Interactions”. In: *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval.* ICTIR ’18. Tianjin, China: Association for Computing Machinery, 2018, pp. 59–66. ISBN: 9781450356565. DOI: 10.1145/3234944.3234954. URL: https://doi.org/10.1145/3234944.3234954.

[154] Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. “Multi-Level Abstraction Convolutional Model with Weak Supervision for Information Retrieval”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* SIGIR ’18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 985–988. ISBN: 9781450356572. DOI: 10.1145/3209978.3210123. URL: https://doi.org/10.1145/3209978.3210123.

[155] Rodrigo Nogueira and Kyunghyun Cho. *Passage Re-ranking with BERT.* 2019. DOI: 10.48550/ARXIV.1901.04085. URL: https://arxiv.org/abs/1901.04085.

[156] Rodrigo Nogueira and Jimmy and Lin. “From doc2query to docTTTTTquery”. In: *Online preprint* 6 (2019).

[157] Rodrigo Nogueira et al. *Multi-Stage Document Ranking with BERT.* 2019. arXiv: 1910.14424 [cs.IR].

[158] Luke Gallagher. *Pairwise t-test on TREC Run Files.* https://github.com/lgrz/pairwise-ttest/. 2019.

[159] Liang Pang et al. “A Study of MatchPyramid Models on Ad-hoc Retrieval”. In: *SIGIR workshop on Neural Information Retrieval (NeuIR-16)* arXiv:1606.04648 (2016). arXiv: 1606.04648. URL: http://arxiv.org/abs/1606.04648.

[160] Liang Pang et al. “Text Matching as Image Recognition”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (Mar. 2016). DOI: 10.1609/aaai.v30i1.10341. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10341.

[161] Jeff Pasternack and Dan Roth. “Extracting Article Text from the Web with Maximum Subsequence Segmentation”. In: *Proceedings of the 18th International Conference on World Wide Web*. WWW ’09. Madrid, Spain: Association for Computing Machinery, 2009, pp. 971–980. ISBN: 9781605584874. DOI: 10.1145/1526709.1526840. URL: https://doi.org/10.1145/1526709.1526840.

[162] Jeffrey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[163] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. “To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks”. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 7–14. DOI: 10.18653/v1/W19-4302. URL: https://aclanthology.org/W19-4302.

[164] Tobias Plötz and Stefan Roth. “Neural Nearest Neighbors Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/f0e52b27a7a5d6a1a87373dffa53dbe5-Paper.pdf.

[165] Prafull Prakash, Julian Killingback, and Hamed Zamani. “Learning Robust Dense Retrieval Models from Incomplete Relevance Labels”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 1728–1732. ISBN: 9781450380379. DOI: 10.1145/3404835.3463106. URL: https://doi.org/10.1145/3404835.3463106.

[166] David Rau and Jaap Kamps. “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 Ma-

chinery, 2022, pp. 153–160. ISBN: 9781450394123. DOI: 10.1145/3539813.3545144. URL: https://doi.org/10.1145/3539813.3545144.

[167] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL].

[168] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends® in Information Retrieval* 3.4 (2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: http://dx.doi.org/10.1561/1500000019.

[169] Stephen E Robertson et al. "Okapi at TREC-3". In: *Nist Special Publication Sp* 109 (1995), p. 109.

[170] Rishiraj Saha Roy and Avishek Anand. "Question Answering for the Curated Web: Tasks and Methods in QA over Knowledge Bases and Text Collections". In: *Synthesis Lectures onSynthesis Lectures on Information Concepts, Retrieval, and Services* 13.4 (2021), pp. 1–194.

[171] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x. URL: https://doi.org/10.1038/s42256-019-0048-x.

[172] Koustav Rudra and Avishek Anand. "Distant Supervision in BERT-Based Adhoc Document Retrieval". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 2197–2200. ISBN: 9781450368599. DOI: 10.1145/3340531.3412124. URL: https://doi.org/10.1145/3340531.3412124.

[173] Koustav Rudra, Zeon Trevor Fernando, and Avishek Anand. *An In-depth Analysis of Passage-Level Label Transfer for Contextual Document Ranking*. 2021. DOI: 10.48550/ARXIV.2103.16669. URL: https://arxiv.org/abs/2103.16669.

[174] G. Salton, A. Wong, and C. S. Yang. "A Vector Space Model for Automatic Indexing". In: *Commun. ACM* 18.11 (Nov. 1975), pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: https://doi.org/10.1145/361219.361220.

[175] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. "Reduce, Reuse, Recycle: Green Information Retrieval Research". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, 2022, pp. 2825–2837. ISBN:

9781450387323. DOI: 10.1145/3477495.3531766. URL: https://doi.org/10.1145/3477495.3531766.

[176] Yelong Shen et al. “A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval”. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM ’14. Shanghai, China: Association for Computing Machinery, 2014, pp. 101–110. ISBN: 9781450325981. DOI: 10.1145/2661829.2661935. URL: https://doi.org/10.1145/2661829.2661935.

[177] Yelong Shen et al. “Learning Semantic Representations Using Convolutional Neural Networks for Web Search”. In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW ’14 Companion. Seoul, Korea: Association for Computing Machinery, 2014, pp. 373–374. ISBN: 9781450327459. DOI: 10.1145/2567948.2577348. URL: https://doi.org/10.1145/2567948.2577348.

[178] Georgios Sidiropoulos and Evangelos Kanoulas. “Analysing the Robustness of Dual Encoders for Dense Retrieval Against Misspellings”. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’22. Madrid, Spain: Association for Computing Machinery, 2022, pp. 2132–2136. ISBN: 9781450387323. DOI: 10.1145/3477495.3531818. URL: https://doi.org/10.1145/3477495.3531818.

[179] R. F. Simmons. “Answering English Questions by Computer: A Survey”. In: *Commun. ACM* 8.1 (Jan. 1965), pp. 53–70. ISSN: 0001-0782. DOI: 10.1145/363707.363732. URL: https://doi.org/10.1145/363707.363732.

[180] Jaspreet Singh and Avishek Anand. “EXS: Explainable Search Using Local Model Agnostic Interpretability”. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM ’19. Melbourne VIC, Australia: Association for Computing Machinery, 2019, pp. 770–773. ISBN: 9781450359405. DOI: 10.1145/3289600.3290620. URL: https://doi.org/10.1145/3289600.3290620.

[181] Jaspreet Singh and Avishek Anand. “Model Agnostic Interpretability of Rankers via Intent Modelling”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* ’20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 618–628. ISBN: 9781450369367. DOI: 10.1145/3351095.3375234. URL: https://doi.org/10.1145/3351095.3375234.

[182] Jaspreet Singh and Avishek Anand. *Posthoc Interpretability of Learning to Rank Models using Secondary Training Data*. 2018. DOI: 10.48550/ARXIV.1806.11330. URL: https://arxiv.org/abs/1806.11330.

[183] Jaspreet Singh, Johannes Hoffart, and Avishek Anand. "Discovering Entities with Just a Little Help from You". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, 2016, pp. 1331–1340. ISBN: 9781450340731. DOI: 10.1145/2983323.2983798. URL: https://doi.org/10.1145/2983323.2983798.

[184] Jaspreet Singh, Wolfgang Nejdl, and Avishek Anand. "Expedition: A Time-Aware Exploratory Search System Designed for Scholars". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '16. Pisa, Italy: Association for Computing Machinery, 2016, pp. 1105–1108. ISBN: 9781450340694. DOI: 10.1145/2911451.2911465. URL: https://doi.org/10.1145/2911451.2911465.

[185] Jaspreet Singh et al. "Extracting per Query Valid Explanations for Blackbox Learning-to-Rank Models". In: *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 203–210. ISBN: 9781450386111. DOI: 10.1145/3471158.3472241. URL: https://doi.org/10.1145/3471158.3472241.

[186] Miroslav Spousta, Michal Marek, and Pavel Pecina. "Victor: the web-page cleaning tool". In: *4th Web as Corpus Workshop (WAC4)-Can we beat Google*. 2008, pp. 12–17.

[187] Trevor Strohman et al. "Indri: A language model-based search engine for complex queries". In: *Proceedings of the International Conference on Intelligent Analysis*. Vol. 2. 6. 2005, pp. 2–6.

[188] Fei Sun, Dandan Song, and Lejian Liao. "DOM Based Content Extraction via Text Density". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, 2011, pp. 245–254. ISBN: 9781450307574. DOI: 10.1145/2009916.2009952. URL: https://doi.org/10.1145/2009916.2009952.

[189] Florian Supplie. "Boilerplate-Erkennung mit Hilfe von Deeplearning". M. Sc. Thesis. Leibniz Universität Hannover, 2018.

[190] Ming Tan et al. "Improved Representation Learning for Question Answer Matching". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 464–473. DOI: 10.18653/v1/P16-1044. URL: https://aclanthology.org/P16-1044.

[191] Nandan Thakur et al. "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)*. 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[192] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. "Top-k Query Evaluation with Probabilistic Guarantees". In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. VLDB '04. Toronto, Canada: VLDB Endowment, 2004, pp. 648–659. ISBN: 0120884690.

[193] James Thorne et al. "FEVER: a Large-scale Dataset for Fact Extraction and VERification". 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, June 2018, pp. 809–819. DOI: 10.18653/v1/N18-1074. URL: https://aclanthology.org/N18-1074.

[194] Nam Khanh Tran and Claudia Niedereée. "Multihop Attention Networks for Question Answer Matching". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 325–334. ISBN: 9781450356572. DOI: 10.1145/3209978.3210009. URL: https://doi.org/10.1145/3209978.3210009.

[195] Iulia Turc et al. "Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation". In: *CoRR* abs/1908.08962 (2019). arXiv: 1908.08962. URL: http://arxiv.org/abs/1908.08962.

[196] Howard Turtle and James Flood. "Query Evaluation: Strategies and Optimizations". In: *Inf. Process. Manage.* 31.6 (Nov. 1995), 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.

[197] Ashish Vaswani et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[198] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. "Web2Text: Deep Structured Boilerplate Removal". In: *Advances in Information Retrieval*. Ed. by Gabriella Pasi et al. Cham: Springer International Publishing, 2018, pp. 167–179. ISBN: 978-3-319-76941-7.

[199] Michael Völske et al. "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, 2021, pp. 13–22. ISBN: 9781450386111. DOI: 10.1145/3471158.3472256. URL: https://doi.org/10.1145/3471158.3472256.

[200] Ellen M Voorhees. "The TREC-8 question answering track report". In: *Trec.* Vol. 99. 1999, pp. 77–82.

[201] Eric Wallace et al. "Concealed Data Poisoning Attacks on NLP Models". 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, June 2021, pp. 139–150. DOI: 10.18653/v1/2021.naacl-main.13. URL: https://aclanthology.org/2021.naacl-main.13.

[202] Jonas Wallat, Jaspreet Singh, and Avishek Anand. "BERTnesia: Investigating the capture and forgetting of knowledge in BERT". In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP.* Online: Association for Computational Linguistics, Nov. 2020, pp. 174–183. DOI: 10.18653/v1/2020.blackboxnlp-1.17. URL: https://aclanthology.org/2020.blackboxnlp-1.17.

[203] Jue Wang et al. "SkipBERT: Efficient Inference with Shallow Layer Skipping". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7287–7301. DOI: 10.18653/v1/2022.acl-long.503. URL: https://aclanthology.org/2022.acl-long.503.

[204] Junfeng Wang et al. "Can We Learn a Template-Independent Wrapper for News Article Extraction from a Single Training Site?" In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '09. Paris, France: Association for Computing Machinery, 2009, pp. 1345–1354. ISBN: 9781605584959. DOI: 10.1145/1557019.1557163. URL: https://doi.org/10.1145/1557019.1557163.

[205] Liang Wang et al. *Text Embeddings by Weakly-Supervised Contrastive Pre-training.* 2022. DOI: 10.48550/ARXIV.2212.03533. URL: https://arxiv.org/abs/2212.03533.

[206] Lijie Wang et al. "A Fine-grained Interpretability Evaluation Benchmark for Neural NLP". In: (2022). DOI: 10.48550/ARXIV.2205.11097. URL: https://arxiv.org/abs/2205.11097.

[207] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. “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, 2021, pp. 317–324. ISBN: 9781450386111. DOI: 10.1145/3471158.3472233. URL: https://doi.org/10.1145/3471158.3472233.

[208] Shuohang Wang et al. “Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering”. In: *International Conference on Learning Representations.* 2018. URL: https://openreview.net/forum?id=rJl3yM-Ab.

[209] Shuohang Wang et al. “R3: Reinforced Ranker-Reader for Open-Domain Question Answering”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence.* AAAI’18/IAAI’18/EAAI’18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[210] Tong Wang, Xingdi Yuan, and Adam Trischler. *A Joint Model for Question Answering and Question Generation.* 2017. arXiv: 1706.01450 [cs.CL].

[211] Xing Wei and W. Bruce Croft. “LDA-Based Document Models for Ad-Hoc Retrieval”. In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 178–185. ISBN: 1595933697. DOI: 10.1145/1148170.1148204. URL: https://doi.org/10.1145/1148170.1148204.

[212] Sarah Wiegreffe and Yuval Pinter. “Attention is not not Explanation”. 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, Nov. 2019, pp. 11–20. DOI: 10.18653/v1/D19-1002. URL: https://aclanthology.org/D19-1002.

[213] Chen Wu et al. *PRADA: Practical Black-Box Adversarial Attacks against Neural Ranking Models.* 2022. DOI: 10.48550/ARXIV.2204.01321. URL: https://arxiv.org/abs/2204.01321.

[214] Shanchan Wu, Jerry Liu, and Jian Fan. “Automatic Web Content Extraction by Combination of Learning and Grouping”. In: *Proceedings of the 24th International Conference on World Wide Web.* WWW ’15. Florence, Italy: International World Wide Web

Conferences Steering Committee, 2015, pp. 1264–1274. ISBN: 9781450334693. DOI: 10.1145/2736277.2741659. URL: https://doi.org/10.1145/2736277.2741659.

[215] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[216] Zhijing Wu et al. "Leveraging Passage-Level Cumulative Gain for Document Ranking". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 2421–2431. ISBN: 9781450370233. DOI: 10.1145/3366423.3380305. URL: https://doi.org/10.1145/3366423.3380305.

[217] Sang Michael Xie and Stefano Ermon. "Reparameterizable Subset Sampling via Continuous Relaxations". In: *IJCAI*. 2019, pp. 3919–3925. URL: https://doi.org/10.24963/ijcai.2019/544.

[218] Ji Xin et al. "DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 2246–2251. DOI: 10.18653/v1/2020.acl-main.204. URL: https://aclanthology.org/2020.acl-main.204.

[219] Caiming Xiong, Stephen Merity, and Richard Socher. "Dynamic Memory Networks for Visual and Textual Question Answering". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2397–2406. URL: https://proceedings.mlr.press/v48/xiong16.html.

[220] Chenyan Xiong et al. "End-to-End Neural Ad-Hoc Ranking with Kernel Pooling". 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, 2017, pp. 55–64. ISBN: 9781450350228. DOI: 10.1145/3077136.3080809. URL: https://doi.org/10.1145/3077136.3080809.

[221] Lee Xiong et al. "Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=zeFrfgyZln.

[222] Huijuan Xu et al. *A Multi-scale Multiple Instance Video Description Network*. 2015. DOI: 10.48550/ARXIV.1505.05914. URL: https://arxiv.org/abs/1505.05914.

[223] Peng Xu et al. *Passage Ranking with Weak Supervision*. 2019. URL: https://openreview.net/forum?id=S1ltj47xdE.

[224] Yingrui Yang, Yifan Qiao, and Tao Yang. “Compact Token Representations with Contextual Quantization for Efficient Document Re-ranking”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 695–707. DOI: 10.18653/v1/2022.acl-long.51. URL: https://aclanthology.org/2022.acl-long.51.

[225] Zichao Yang et al. “Hierarchical Attention Networks for Document Classification”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://aclanthology.org/N16-1174.

[226] Lan Yi, Bing Liu, and Xiaoli Li. “Eliminating Noisy Information in Web Pages for Data Mining”. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’03. Washington, D.C.: Association for Computing Machinery, 2003, pp. 296–305. ISBN: 1581137370. DOI: 10.1145/956750.956785. URL: https://doi.org/10.1145/956750.956785.

[227] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. “INVASE: Instance-wise Variable Selection using Neural Networks”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=BJg_roAcK7.

[228] Hamed Zamani and W. Bruce Croft. “Embedding-Based Query Language Models”. In: *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*. ICTIR ’16. Newark, Delaware, USA: Association for Computing Machinery, 2016, pp. 147–156. ISBN: 9781450344975. DOI: 10.1145/2970398.2970405. URL: https://doi.org/10.1145/2970398.2970405.

[229] Jingtao Zhan et al. “An Analysis of BERT in Document Ranking”. 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, 2020, pp. 1941–1944. ISBN: 9781450380164. DOI: 10.1145/3397271.3401325. URL: https://doi.org/10.1145/3397271.3401325.

[230] Jingtao Zhan et al. “Optimizing Dense Retrieval Model Training with Hard Negatives”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1503–1512. ISBN: 9781450380379. URL: https://doi.org/10.1145/3404835.3462880.

[231] Kai Zhang et al. "LED: Lexicon-Enlightened Dense Retriever for Large-Scale Retrieval". In: *Proceedings of the ACM Web Conference 2023*. WWW '23. Austin, TX, USA: Association for Computing Machinery, 2023, pp. 3203–3213. ISBN: 9781450394161. DOI: 10.1145/3543507.3583294. URL: https://doi.org/10.1145/3543507.3583294.

[232] Zijian Zhang, Koustav Rudra, and Avishek Anand. "Explain and Predict, and Then Predict Again". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. WSDM '21. Virtual Event, Israel: Association for Computing Machinery, 2021, pp. 418–426. ISBN: 9781450382977. DOI: 10.1145/3437963.3441758. URL: https://doi.org/10.1145/3437963.3441758.

[233] Ruiqi Zhong, Steven Shao, and Kathleen McKeown. *Fine-grained Sentiment Analysis with Faithful Attention*. 2019. DOI: 10.48550/ARXIV.1908.06870. URL: https://arxiv.org/abs/1908.06870.

[234] Yucheng Zhou et al. *Fine-Grained Distillation for Long Document Retrieval*. 2022. DOI: 10.48550/ARXIV.2212.10423. URL: https://arxiv.org/abs/2212.10423.

[235] Shengyao Zhuang and Guido Zuccon. "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, 2022, pp. 1444–1454. ISBN: 9781450387323. DOI: 10.1145/3477495.3531951. URL: https://doi.org/10.1145/3477495.3531951.

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

[237] Shengyao Zhuang and Guido Zuccon. "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, 2021, pp. 1483–1492. ISBN: 9781450380379. DOI: 10.1145/3404835.3462922. URL: https://doi.org/10.1145/3404835.3462922.

List of Figures

List of Tables

List of Algorithms