Deep learning for information retrieval : studying relevant signals for ad hoc search based on transformer models

Lila Boualili

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *23/11/2022* **par :**

Lila BOUALILI

Deep Learning for Information Retrieval: Studying relevant signals for ad hoc search based on transformer models

---

JURY

Pr. Gabriella Pasi	Université de Milan-Biccoca	Rapporteure
Pr. Eric Gaussier	Université de Grenoble Alpes	Rapporteur
Pr. Lynda Tamine	Université de Toulouse III	Présidente du jury
Pr. Sylvain Lamprier	Université d'Angers	Examinateur
Pr. Mohand Boughanem	Université de Toulouse III	Directeur de thèse
Dr. José G.Moreno	Université de Toulouse III	Co-directeur de thèse
Dr. Andrew Yates	Université d'Amsterdam	Invité

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur(s) de Thèse :
Mohand Boughanem et *José G.Moreno*
Rapporteurs :
Eric Gaussier et *Gabriella Pasi*

DEEP LEARNING FOR INFORMATION RETRIEVAL: STUDYING RELEVANT SIGNALS FOR AD HOC SEARCH BASED ON TRANSFORMER MODELS

LILA BOUALILI

Manuscrit de thèse

Université Toulouse III – Paul Sabatier
Institut de Recherche en Informatique de Toulouse

Directeur de thèse : Mohand Boughanem
Co-directeur de thèse : José G.Moreno

Contact me for any comments and corrections: lila.boualili@irit.fr
Institut de Recherche en Informatique de Toulouse, UMR 5505 CNRS,
Université Toulouse III Paul Sabatier,
118 route de Narbonne,
F-31062 Toulouse CEDEX 9

Acknowledgement

It is with a lot of emotion that I finally reach the end of this thesis. A unique experience indeed, with its joys and challenges but most importantly a very rich experience which allowed me to make beautiful encounters both professionally and personally. I would like to express my gratitude to them here and thank them for everything they have done.

First and foremost, I would like to express my deepest appreciation to my thesis supervisors: *Mohand Boughanem*, Professor at the University of Toulouse III and *José G.Moreno*, Assistant Professor at the University of Toulouse III, for the continuous support they gave me throughout the last three years preparing for my Ph.D. Their guidance, advice, and requirement, oh so invaluable, strongly contributed to the achievement of this thesis. I am particularly grateful for the knowledge and experience they shared with me and the pedagogy and rigor that are theirs. May they be assured of my gratitude and my deep respect.

I am also deeply indebted to *Andrew Yates*, Assistant Professor at the University of Amsterdam, for serving as a mentor during my internship at the Max Planck Institute for Informatics of Saarbrücken. This experience enabled me to pursue a new exciting research direction and gain precious skills with my mentor's support. I got to meet many young researchers from different nationalities and learn alongside them. This wonderful experience would not have been possible without Andrew.

I would like to thank the remaining members of the committee for giving me the honor of evaluating my dissertation. I would like to express my respect for them. My gratitude goes to *Lynda Tamine*, Professor at the University of Toulouse III, *Eric Gaussier*, Professor at the University of Grenoble Alpes, *Gabriella Pasi*, Professor at the University of Milan-Bicocca, and *Sylvain Lamprier*, Professor at the University of Angers.

I am thankful to the CIMI Labex for funding my doctoral research for the past three years. My sincere thanks also go to *Gilles Hubert*, Assistant Professor at the University of Toulouse III for having welcomed me in his team during the preparation of my Ph.D., and for having spared no effort to ensure a friendly work environment, favourable to innovation and surpassing oneself. My gratitude also goes to *Gerhard Weikum*, Professor at Saarland University for having welcomed me as an intern and having given me access to the laboratory and research facilities, and all the resources I needed for conducting my internship work. Finally, I want to thank all the people who

helped me throughout my stay in their laboratory, at IRIT or at MPI, and allowed me to pursue my Ph.D. work in a favourable environment.

Special thanks to *Lynda Said Lhadj*, Assisstant professor at ESI, who supported me since my first year in engineering school at ESI, and mentored me for my Master's and Engineering degrees. I cannot thank you enough for your kind words, your support and advice. I am particularly grateful to you for introducing me to research in general and the domain of information retrieval in particular, and for instilling in me the scientific rigor that is yours. I would not have gotten into the Ph.D. journey if it were not for you.

During these years preparing for my Ph.D., I have built long-lasting friendships with passionate and generous people. I particularly thank *Nicolas Bizzozzero* and *Raphael Sourty* for our stimulating discussions, for the support when I was at my worst, you would always find the right words to brighten my mood. Thank you for all the precious time we had while working together in the last three years at the 406 office, last floor at IRIT 1, or what is best known as "hell" during summer. I would have never made it to the end of this journey without you ;) I have met wonderful people who helped me out through this thesis, a special shout out to: *Paul, Rafik, Malik, Damien, Morgan, Luis, Thiziri, Alexis, Maël, Antoine, Mira, Farane, Nishchal, Hina, Aya* it was a real pleasure to share this journey with you, thank you for all the fun times we spend together.

I cannot forget to thank my best friend since high school, *Lynda*. Fate may have scattered us around the world, but our friendship remains strong. With your wisdom, your precious advice and your ability to listen, you have contributed enormously to the success of this thesis and in many other things...

Words cannot express my gratitude to *Aghiles*. You have supported me, shared my joys, consoled my sorrows, and erased my doubts. You have stayed by my side through thick and thin this past year. Even when I felt like giving up, you were always there to help me get back on my feet and fight until the very end. I could not have undertaken this journey without you.

Last but not least, I would like to thank my family, whom I love so deeply. I thank my beloved big sister for being so supportive and my younger brother for his caring. But my deepest gratitude goes to my parents, my twin pillars, without whom I could not stand. You never gave me any idea that I could not do whatever I put my mind to or be whomever I aspired to be. Thank for filling our home with love and books. You have instilled in me the taste for knowledge and brought me up to have a curious mind, thank you for showing me the way... Words could never express my appreciation for all the efforts and sacrifices you had to make so I could enjoy a better life. I hope you are proud of what I have become and what I have achieved.

Résumé

Au cours de la dernière décennie, les modèles supervisés d'apprentissage profond ont apporté des améliorations substantielles à une multitude de tâches de Traitement Automatique des Langues (TAL). Les réseaux de neurones profonds ont été utilisés pour apprendre des représentations vectorielles continues du texte, capables de modéliser la sémantique. Afin de tirer profit de l'appariement sémantique, plusieurs modèles d'apprentissage profond ont été proposés, souvent adaptés de ceux conçus pour les tâches de TAL afin de répondre à différentes tâches de Recherche d'Information (RI) telles que la recherche ad hoc. Cependant, les améliorations dans les tâches de RI sont restées à la traîne par rapport à des tâches similaires en TAL, malgré les efforts considérables de la communauté. Bien que plusieurs facteurs y aient contribué, une raison importante de cet "échec" provient des caractéristiques uniques de la tâche de recherche en RI, en particulier, lorsqu'on la compare aux tâches relevant de l'appariement de textes en TAL. En effet, en RI, à travers l'appariement document-requête on cherche à modéliser la pertinence du document vis-à-vis d'une requête, c'est-à-dire l'adéquation du contenu du document vis-à-vis du besoin formulé dans la requête. On ne cherche pas à calculer la proximité sémantique entre les mots de la requête et du document. Or, c'est précisément, ce que réalise la majorité des modèles neuronaux dans les tâches de TAL, apprendre des représentations pour apparier deux textes, identifier la sémantique d'un texte ou déduire des relations sémantiques entre deux morceaux de texte, etc.

Récemment, les Modèles de Langue Pré-entraînés (MLPs) contextualisés, dont BERT est l'exemple le plus célèbre, qui sont capables d'apprendre des représentations de mots dans leurs contexte, ont obtenu des résultats de pointe dans la recherche ad hoc avec de larges marges de performance. Bien que les modèles de recherche basés sur les MLPs soient également adaptés de tâches similaires d'appariement de phrases dans le domaine du TAL, avec des modifications minimes, ils se sont étonnamment avérés très efficaces par rapport aux tentatives précédentes. Ce succès sans précédent peut être attribué à la grande quantité de pré-entrainement non supervisé sur des objectifs de modélisation du langage, combiné avec la flexibilité du processus de contextualisation dans les transformers. Mais aussi au fine-tuning sur de larges quantités de données labellisées disponibles publiquement pour la tâche d'ordonnancement de documents.

Dans cette thèse, nous nous intéressons à l'adaptation des MLPs à la tâche spécifique de la recherche ad hoc. Nous explorons différentes pistes de

recherche pour construire de meilleurs modèles de RI basés sur les MLPs : (1) explorer l'impact de l'intégration de l'intuition traditionnelle d'appariement exact sur l'efficacité des MLPs pour la recherche ad hoc ; (2) étudier le rôle du processus de contextualisation dans les MLPs pour la recherche ad hoc afin de mieux comprendre ce qui est important pour cette tâche, ce qui pourrait motiver des reconceptions plus efficaces des transformers spécifiquement pour la recherche ad hoc.

En ce qui concerne la première piste, nous proposons de considérer une intuition traditionnelle qui est importante pour la recherche ad hoc, à savoir l'appariement exact, qui a été utilisé en RI pendant des décennies jusqu'à très récemment dans la conception de modèles neuronaux pré-BERT. Au lieu de construire des modèles neuronaux plus grands ou d'améliorer leur supervision, nous prenons une voie différente en intégrant des connaissances du domaine de la RI. Nous proposons une stratégie de marquage simple mais efficace qui met l'accent sur les terms qui sont en commun entre la requête et le document, au niveau de l'entrée en introduisant stratégiquement des marqueurs spéciaux. Cette approche tire parti de la flexibilité de l'architecture des transformers dans les MLPs pour intégrer des intuitions supplémentaires spécifiques aux tâches afin d'améliorer leur efficacité.

Dans la deuxième direction, nous explorons le processus de contextualisation flexible dans les MLPs pour l'appariement semantic dans le contexte de la recherche ad hoc. Puisque ce même processus de contextualisation effectué par les transformers dans les MLPs est capable d'effectuer efficacement différentes tâches en aval, nous étudions s'il peut être contraint à un processus plus simple spécifiquement conçu pour la tâche de recherche. Pour ce faire, nous proposons la distillation d'un MLP oracle dans des modules plus simples et soigneusement conçus, basés sur des embeddings statiques afin d'analyser le rôle du processus de contextualisation pour la tâche de recherche. Alors que la piste de recherche précédente intègre plus de signaux dans le processus de contextualisation des MLPs pour les adapter à la tâche de recherche (augmenter l'efficacité), cette piste tente de limiter les signaux dans le processus de contextualisation à ceux qui sont nécessaires pour la recherche ad hoc (atteindre de meilleurs compromis efficacité/efficience).

Mots clés : Recherche d'Information, Apprentissage Profond, Recherche Ad hoc, Traitement Automatique des Langues, Modèles de Langue Pré-entraînés, BERT, Transformer, Appariement Exact, Appariement Sémantique

Abstract

In the past decade, supervised deep learning models have yielded substantial improvements to many Natural Language Processing (NLP) tasks. Deep neural networks have been used to learn continuous vector representations of text capable of modeling semantics. With a view to taking advantage of semantic matching, several deep learning models were proposed, often adapted from those designed for NLP tasks to meet different Information Retrieval (IR) tasks such as ad hoc search. However, improvements in IR tasks lagged behind those in similar NLP tasks, despite considerable efforts from the community. Although there are various contributing factors, a critical reason for this "failure" comes from the unique characteristics of the ranking task in IR, particularly when compared to the tasks of text matching in NLP. Indeed, in IR, through query-document matching, we try to model the relevance of the document with respect to the query, i.e., the adequacy of the document's content with respect to the information need formulated in the query. We do not try to calculate the semantic similarity between words in the query and the document. However, this is precisely what most neural models achieve in NLP tasks, learning representations to match two texts, identifying the semantics of a text, or inferring semantic relationships between two pieces of text, etc.

Recently, contextualized Pre-trained Language Models (PLMs), of which BERT is the most famous instance, are capable of learning representations of words in context and have achieved state-of-the-art results in ad hoc search with substantial performance leaps. Although PLM-based ranking models are also adapted from similar sentence-matching tasks in NLP with minimal modifications, they have surprisingly proven to be highly effective as opposed to previous attempts. This unprecedented success can be owed to the heavy unsupervised pre-training on language modeling objectives combined with the flexibility of the contextualization process in transformers. Additionally, the availability of large amounts of labelled data for the ranking task enables effective fine-tuning of PLMs.

In this thesis, we focus on adapting prominent PLMs to the specific task of ad hoc ranking. We explore different research directions for building better PLM-based ranking models: (1) exploring the impact of integrating the traditional exact matching intuition on the ranking effectiveness of PLMs; (2) investigating the role of the contextualization process in PLMs for ranking to gain insight into what is important for ranking which could motivate more efficient ranking-specific redesigns of transformers.

Regarding the first direction, we propose considering a traditional intuition that is important for ranking: exact matching, which has been used in IR for decades until very recently in the design of pre-BERT neural models. Instead of building larger neural models or improving their supervision, we take a different path forward by integrating knowledge in the field of IR. We propose a simple yet effective marking strategy that emphasizes exact term matches between the query and the document at the input level by strategically introducing special marker tokens. This approach takes advantage of the flexibility of the transformer architecture in PLMs to integrate additional task-specific intuitions in order to improve their effectiveness.

For the second direction, we dive into exploring the flexible contextualization process in PLMs for soft matching in the context of ranking tasks. Because this same contextualization process performed by transformers in PLMs is able to perform different downstream tasks effectively, we investigate if it can be constrained to a simpler process specifically designed for the ranking task. To do so, we propose distillation from an oracle PLM into simpler carefully-designed modules based on static embeddings and information bottlenecks to analyze the role of the contextualization process for the ranking task. While the previous research direction integrates more signals into the contextualization process of PLMs to adapt them to the ranking task (increase effectiveness), the later direction tries to constrain the signals in the contextualization process to only what is necessary for ranking (achieve better efficiency/effectiveness trade-offs).

Keywords: Information Retrieval, Deep Learning, Ad hoc Search, Natural Language Processing, Contextualized Pre-trained Language Models, BERT, Transformer Architectures, Exact Matching, Soft Matching

Publications

Parts of our contributions have already been published in the following scientific publications:

International Conference Paper

Lila BOUALILI, Jose Moreno, and Mohand Boughanem. 2020. MarkedBERT: Integrating Traditional IR Cues in Pre-Trained Language Models for Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR'20)*. Association for Computational Linguistics, Online, 4758-4781.

International Journal Article

Lila BOUALILI, Jose Moreno, and Mohand Boughanem. 2022. Highlighting exact matching via marking strategies for ad hoc document ranking with pretrained contextualized language models. *Information Retrieval Journal* (2022), 1-47.

Table of Contents

List of Figures

List of Tables

Acronyms

ANN Approximate Nearest Neighbour
BERT Bidirectional Encoder Representations from Transformers
BM25 Best Match 25
BoW Bag of Words
CNN Convolutional Neural Networks
ColBERT Contextualized Late Interaction over BERT
DL Deep Learning
ELECTRA Efficiently Learning an Encoder that Classifies Token Replacements Accurately
FNN Feed-forward Neural Networks
IDF Inverse Document Frequency
IR Information Retrieval
IRS Information Retrieval System
LM Language Model
LSTM Long Short Term Memory
MLM Masked Language Modeling
MS MARCO MicroSoft MAchine Reading COmprehension
MSE Mean Squared Error
NLP Natural Language Processing
NSP Next Sentence Prediction
PLM Pre-trained Language Model
PRF Pseudo-Relevance Feedback
TF Term Frequency
TREC Text Retrieval Evaluation conference

Introduction

General Introduction

Context

Information retrieval (IR) is the field of research that deals with the representation, storage, organization of information items in order to provide the users with easy access to the information in which they are interested [7].

Salton [211], the godfather of IR, defines it as follows:

> *Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.*

Practically, information retrieval is the science behind search engines (or IR systems) that allows them to retrieve relevant results from large corpora of documents (e.g., texts, images, etc.) which are most likely to satisfy the information need expressed by a user in a query; also known as the ad hoc ranking task. A key challenge in IR is, hence, to design formal approaches capable of modeling the notion of relevance. Today, the literature gathers a wide range of IR models, each formalizing relevance with different methods and tools as advances have been made over the decades.

The first IR systems, developed in the late 1950*s*, focused on finding better ways to index texts, and then use new algorithms to search these (mostly) automatically built indexes [85]. These early systems were mainly intended for searching electronic information sources available in bookstores and academic institutions. By the 1990*s*, the democratization of the web and the diversification of electronic media have given rise to an overabundance of information, making them accessible to organizations as well as to individuals. This is how IR systems became ubiquitous in different contexts such as digital libraries, e-commerce applications, and web search engines, with over 8 billion queries processed daily by the notorious *Google* search engine in 2022 [1].

Although the exact details of popular commercial web search engines remain elusive due to their proprietary nature, at their core, they still rely on *lexical* keyword matching [184] to retrieve and rank results [2] [210].Keyword search, which relies on exact lexical match, commonly based on bag-of-word (BoW) queries, is also widely used by practitioners of IR who develop and deploy search applications, and academics [173, 149, 51, 174]. Lexical matching is limited, though, since terms in documents and queries are

1. https://earthweb.com/how-many-google-searches-per-day/
2. https://www.google.com/search/howsearchworks/how-search-works/ranking-results/

treated as meaningless graphic units. The retrieved documents do not always address the topic of the user's query because term matches are considered independently from their context. However, a document that uses a query term in a relevant context should be ranked above documents that use the term incidentally or in a different sense.

Though Natural Language Processing (NLP) techniques such as word sense disambiguation [215, 170] and semantic indexing [238, 15] have been previously applied to search, their results were mitigated. Later, supervised deep neural learning approaches have dominated the field of NLP [181], and have been used to learn continuous representations of text units (we focus on words and sub-words) which capture semantics through distributional modelling, namely embedding models [162, 189]. These distributed representations enable soft or semantic matching, where query terms *do not* have to match document terms *exactly* in order to contribute to relevance. Consequently, neural models [95, 83, 250] became prominent in the IR community to alleviate the *vocabulary mismatch problem*, that queries and relevant documents do not necessarily use the same terms, e.g., covid vs. coronavirus, for the same concept.

Recent improvements to term representations have allowed the modeling of terms in a given textual context using pre-trained language models [3] (PLMs) such as ELMo [190] and BERT [58]. These PLMs offer a considerable opportunity for neural IR as they allow models to take advantage of massive amounts of unlabeled natural language to help model terms as they appear in a given context. As opposed to previous neural IR models, known as pre-BERT models, contextualized vector representations further address the *semantic mismatch problem*, that the same term can refer to different concepts, e.g., right hand vs. right answer.

Fine-tuned PLMs achieve state-of-the-art results in ad hoc ranking with huge performance leaps [129], and they grew rapidly to dominate the IR research landscape in a matter of a few years. PLMs are now the de facto building blocks for neural ranking models, mainly applied as black boxes with minimal modifications. This direct application allows IR researchers to take advantage of innovations in NLP (that may not have been intended for ranking) "for free" and fits nicely with the "more data, larger models" strategy [129]. It is interesting to observe how the design of neural IR models evolved from capturing multiple intuitions important for ranking through specialized architectural components (e.g., convolutional and recurrent components) in pre-BERT models to using PLMs where all these specialized components are subsumed in the attention mechanisms of transformers. Thanks to the fine-tuning recipe, the application of PLMs is straightforward. Consequently, the same architecture based on homogeneous transformer layers is employed regardless of the downstream task. There is hardly any distinction between

3. Also referred to as contextualized language models or transformer models.

PLMs for semantic similarity in NLP and relevance matching in IR. The contextualization process in PLMs can either lack important signals for the ranking task or integrate superfluous signals unnecessary for ranking but used for other downstream tasks. The work we present in this dissertation is, therefore, interested in adapting PLMs to the ranking task by injecting traditional intuitions important for ranking or investigating the contextualization process in PLMs by constraining its flexibility to what the ranking task requires only.

Research Issues and Contributions

This dissertation is articulated around the development of neural IR models in the BERT era where large-scale contextualized language models are prevalent. More specifically, we focus on the adaptation of these flexible transformer models for the ad hoc ranking task, and investigate the following two main issues: (1) the impact of emphasizing the traditional exact match intuition on transformer models for document ranking; (2) the impact of constraining the contextualization process in transformer models to simpler aggregation approaches on ranking effectiveness.

Emphasizing Exact Match cues in PLM-based ranking models

PLMs such as BERT and its subsequent variants (e.g., ELECTRA [40], T5 [197]) have achieved unprecedented success on various NLP tasks. The success of these models is largely owed to the heavy pre-training on language modeling objectives on the one hand and learning contextualized representations using the transformer architecture on the other. Thanks to the fine-tuning strategy and the availability of large publicly-released training datasets, applying a PLM to ad hoc ranking is straightforward. Nogueira and Cho [173] proposed the first successful application of BERT to the ranking task, known as monoBERT, kicking off the "BERT revolution" in IR [129]. The authors fine-tune the configuration proposed by Devlin et al. [58] in the original BERT paper for *sentence pair classification* tasks, on the large public MS MARCO dataset [9]. Though this configuration was not specifically designed for capturing the different aspects of query-document relevance, it outperformed, by large margins, pre-BERT neural models, which were carefully designed to capture different signals important for ranking. This gives rise to the following question: *Should we forget about the insights and intuitions accumulated throughout the development of IR models and focus instead on scaling up transformer-based models and improving their supervision?* This might actually be a good way forward, as demonstrated by later work using large-scale generic PLMs [174]. Nevertheless, this might not be the only path

forward. We believe that domain knowledge from decades of developing ranking models can interact with PLMs. In fact, an important insight from pre-BERT model designs is that a robust model should handle both semantic and exact match signals properly [83, 250, 165]. Indeed, researchers emphasized the difference between NLP models centered around semantic matching and IR models focused on relevance matching, where exact and semantic matches are complementary. However, recent designs relying on PLMs are erasing the distinction between the two threads of research [129].

We believe that exact matching is still an essential signal for assessing the relevance of a document to an information-seeking query aside from soft semantic matching. In this dissertation, we examine if recent contextualized LMs such as BERT or ELECTRA can benefit from explicit exact match cues to better adapt to the document ranking task.

The approach we propose is an adaptation of the monoBERT model to the document ranking task through the integration of the traditional exact match intuition [21, 22]. We, therefore, explore *marking strategies* which emphasize exact term matches between queries and documents using *special marker tokens* introduced in the textual input of the model. This way, the overlapping terms between the query and document (up to morphological changes, e.g., ride/riding/rider) are highlighted with special marker tokens to indicate their importance, among other terms, for the ranking task. We explore different marking strategies, including different types of marker tokens.

Our approach is based on augmenting the PLM input with special marker tokens to promote exact term matches. Thus, the PLM learns how these marker tokens interact with other tokens in the input sequence to build meaningful contextual representations for these special tokens. We build variations of the monoBERT model to investigate the contribution of the contextual representations of the marker tokens to relevance scoring. Aside from exact match cues, we further investigate the use of marker tokens to learn implicit query expansion representations with PLMs.

We evaluate the contribution of exact match marking to the ranking performance across different experimental scenarios on three standard TREC benchmarks. We indeed conduct exhaustive experiments and demonstrate the effectiveness of our exact match marking approach on in-domain collections and show its zero-shot generalization ability to out-of-domain collections. We further evaluate its effectiveness in scenarios incorporating techniques for out-of-domain adaptation and conduct comparative evaluations with state-of-the-art PLM-based ranking models. Our findings support that traditional information retrieval cues such as exact matching are still valuable for large pre-trained contextualized models such as BERT and ELECTRA.

Distilling the Contextualization process in PLM-based ranking models into simpler approaches

Contextualized representations produced by PLMs have significantly improved the performance of neural ranking models in the past few years, thanks to their ability to represent terms in their context. Recent developments in neural IR are moving towards representation-centered approaches that focus on applying PLMs for learning representations of texts specifically tailored for ranking [110, 251, 112, 75, 217]. These approaches adopt a bi-encoder design [98] where the query and document are processed independently. This design contrasts with cross-encoders, the standard BERT design adopted in earlier applications such as monoBERT, that benefits from all-to-all attention across tokens from both the query and the document. Simple bi-encoder approaches encode each query and each document into a single vector representation, and relevance is defined in terms of similarity between the two vectors. Alternatively, ColBERT [112] proposes representing each query and each document by the contextual representations of their tokens (for more interpretability), and relevance is given by soft matching all token vectors. The success of these approaches in which query-document matching is modeled with simple vector similarities (e.g., dot product) demonstrates that PLMs can encode information important to relevance estimation in token representations. However, the contextualization process in these models is opaque and complex and can integrate various information about syntax and semantics [230] or how the sense of a term can vary across different contexts [246]. It is yet unknown which of these signals are important for the ranking task and which are expendable. We, therefore, wonder: *Do we need the full flexibility of the complex contextualization process of PLMs for ranking or can we simplify it?*

Findings from recent work on residual compression [217] for reducing the memory footprint of the ColBERT index indicate the possibility of summarizing the semantic space produced by the BERT encoder by a set of centroid vectors along with minor refinements.

In this dissertation, we investigate the contextualization process in a PLM, and examine whether a simpler contextualization process can perform as well as a PLM on the ranking task.

We propose distilling the contextualization process in a PLM, considered as an oracle, into simpler aggregation methods based on static embeddings. More specifically, we devise two aggregation methods:

1. *Intrinsic Aggregation.* Distills the entire semantic space produced by the oracle PLM, for each token, into a combination of a finite set of static sub-embeddings. The static sub-embeddings of a token are meant to capture coarse-grained information relative to this token, such as its senses and general aspects, e.g., bank of river or bank in finance.

This simple aggregation aims to explore whether the contextualization process can be effectively replaced with a combination of a smaller number of static sub-embeddings.

2. *Extrinsic Aggregation.* Distilling the rich semantic space of tokens into a small number of static sub-embeddings is unlikely to capture fine-grained topic information tied to the same high-level aspect of a token (e.g., "bank robberies" vs. "deposit money in the bank" are related to the financial aspect of "bank"). We, thus, investigate if such fine-grained variations can be captured using information from the local context (e.g., one token to the left/right).

We pre-train a combination of intrinsic and extrinsic aggregation modules to approximate, via distillation, the contextualized representations from a ColBERT model, referred to as the oracle. The pre-trained models are then fine-tuned on the ranking task in a supervised setting. We use these models to study how well simplified intrinsic and extrinsic contextualization approaches perform on standard evaluation benchmarks and whether they are complementary. We further conduct a thorough ablation analysis to demonstrate the importance of each architectural component composing the modules and their parameters (e.g., the number of sub-embeddings per token). The results of our experiments show that using our simplified contextualization approach matches and occasionally outperforms the ColBERT oracle. This suggests that the full flexibility of a PLM is not required to create representations suitable for ranking, which could motivate more efficient architectures for ranking.

Thesis Organization

The remainder of this dissertation consists of three parts, a conclusion part, and an appendix.

Part I: Background

This part presents the fundamentals of information retrieval and an overview of the major developments in IR models, in two chapters:

Chapter 1: Core concepts in information retrieval

The goal of this chapter is to present information retrieval and its fundamental concepts, and describe the evaluation protocol of IR models.

Chapter 2: A historical overview of IR models

This chapter presents the major developments in IR over the past three quarters of a century up until the arrival of BERT. We start by presenting early models which were based on exact matching, and the first attempts to alleviate the subsequent vocabulary mismatch problem. Then we briefly cover learning to rank methods based on machine learning approaches using hand-crafted features. Finally, the different models developed in neural IR, which learn continuous vector representations of text to overcome the vocabulary mismatch problem, are reviewed up until the introduction of contextualized language models to IR and the start of the so-called "BERT revolution".

Part II: State Of The Art Overview

This part of the dissertation is dedicated to the applications of contextualized language models to the text ranking task. After presenting the BERT model in chapter 3, we organize its applications to text ranking into three chapters: Chapter 4 covers the applications of BERT and its subsequent variants in multi-stage reranking approaches. Chapters 5 and 6 present the applications of BERT for retrieval using sparse and dense representations, respectively.

We mainly develop work in Chapter 4 and Chapter 6 since they motivate our two contributions. Our first contribution follows the cross-encoder design used in early works (Chapter 4). Our second contribution is motivated by the recent developments covered in Chapter 6, which adopt a bi-encoder design.

Chapter 3: Overview of BERT

This chapter presents in detail the most common pre-trained language model, BERT. We describe its transformer-based architecture and its different configurations. We also overview works from the NLP field that attempt to understand how BERT works, and present important findings about the knowledge encoded in BERT weights.

Chapter 4: BERT in multi-stage reranking

This chapter covers the diverse models that integrate BERT or its variants in the reranking stage(s) of a multi-stage reranking architecture. We present in detail the first application of BERT as a reranker on top of bag-of-words retrieval, namely monoBERT [173], which serves as the starting point for many subsequent works. We cover different works which propose techniques to overcome the length limitation of BERT. We further describe efforts toward

building more efficient reranking models. Finally, we discuss adaptations of pre-trained sequence-to-sequence models to the ranking task.

Chapter 5: BERT for sparse retrieval

This chapter focuses on the applications of BERT for query and document expansion and term-reweighing in order to mitigate the vocabulary mismatch problem. Instead of directly using BERT for ranking, models in this chapter employ the contextual language model to refine query or document sparse representations to bring them into closer alignment. Existing lexical sparse retrievers based on exact matching can then be used for retrieval using efficient inverted indexes. We cover expansion models manipulating term-based or textual representations of queries and documents from the corpus. We also review methods involving non-textual representations, but rather learn sparse expansion via BERT or even directly learning sparse representations.

Chapter 6: BERT for dense retrieval

Models covered in this chapter apply PLMs to learn dense representations of texts suitable for retrieval. This approach to retrieval has the potential to address the vocabulary mismatch problem by directly performing relevance matching in the semantic space created by the PLM. We formally define the so-called dense retrieval framework and its bi-encoder design, which contrasts with the more effective but more expensive cross-encoder design, which is thus limited to reranking (chapter 2). In the same way that sparse retrieval requires inverted indexes, we present the infrastructure supporting dense retrieval. We then present the different dense retrieval models proposed in the literature, which can be categorized based on the output of the encoder into single-vector systems, which encode each query/document into a single vector, and multi-vector systems, which encode each query or document into multiple vectors. Finally, we discuss attempts to enhance the effectiveness of bi-encoders to close the gap to cross-encoders.

Part III: Contributions

This part details our contributions. It comprises two chapters: Chapter 7 presents our first contribution, which examines the impact of emphasizing traditional exact match signals on ranking with PLMs. Chapter 8 presents our second contribution, which investigates whether restraining the flexibility of the contextualization process in a PLM can still perform as well as the original PLM on the ranking task.

Parts of Chapter 7 are reproductions of my jointly authored publications [21, 22]. Given the fast pace of progress in neural IR, results in

each chapter are presented as they were at the time of original publication. That is, new baselines were not added retroactively to studies.

Chapter 7: Highlighting exact matches for ad hoc ranking with transformer models

This chapter presents in detail the context and motivations of our first proposition and the research questions we study. We describe the marking strategies we propose for highlighting exact match signals and the architecture of our model. We then move to the experimental validation of our approach, where we show that explicit exact match signals introduced via marking are beneficial when ranking with PLMs. We also demonstrate the effectiveness of our approach across different scenarios on different standard benchmarks and how they compare to state-of-the-art models.

Chapter 8: Investigating contextualized representations for ad hoc ranking

We present in this chapter the details of our second contribution, in which we first recall the context and motivation behind our proposition. Then, we describe our methodology, including the proposed modules for intrinsic and extrinsic aggregations and the distillation process from the oracle PLM. Next, we present the experimental setup and the evaluation scenarios we use to validate our proposed approach empirically. We demonstrate that contextualized representations produced by the flexible oracle PLM can be approximated with a simpler approach without losing effectiveness on the ranking task.

Conclusion

This part concludes this dissertation and presents future directions for our work on the short and long terms.

Appendices

This dissertation includes two appendices. The first Appendix A extends the results presented in our first contribution (chapter 7). The second Appendix B presents additional details about the reproducibility of our work.

Part I

Background

1

Core concepts in information retrieval

1 *Introduction*

The need for information access assisted by computing machines emerged in the mid-1940s as a response to the exponential growth of scientific publications in the wake of World War II. The term "Information Retrieval", abbreviated IR, was introduced shortly thereafter by Mooers [167]. It is the field of study that is interested in developing search models and systems to satisfy a user seeking information.

The core of IR is to provide relevant information to users in response to their information needs. Usually, a search starts when a user issues a *query* expressing a search intent. The goal of the IR system (IRS) is to first identify material that satisfies the query from large collections of data, such as Web pages, textual documents, images or any other type of document, and then return an ordered list of these materials to the user, according to their relevance degree. Thus, the fundamental problem is to estimate the relevance score of documents in a collection with respect to the user's query. Existing works propose a wide range of models to evaluate query-document relevance on the basis of different strategies, which we overview in Chapter 2.

With the development of IR, the community established standardized evaluation procedures. The Cranfield paradigm is a system-oriented evaluation defined in the Cranfield project by Cleverdon [42]. It has come to dominate the IR research landscape and is at the origin of numerous evaluation campaigns such as TREC (Text Retrieval Evaluation conference) and CLEF (Conference and Labs of the Evaluation Forum). Despite our focus on the Cranfield paradigm, there are other user-oriented evaluation paradigms which are necessary to accurately evaluate particular approaches: interactive evaluations including humans in the evaluation process [111], A/B testing [116] for evaluating online services with substantial numbers of users.

The objective of this chapter is to present the fundamental concepts of information retrieval. It is organized as follows. We start by presenting core concepts and definitions in section 1.2. Then, in section 1.3, we formally characterize the evaluation of IR systems, i.e: evaluation metrics and evaluation campaigns and reusable test collections.

2 *Core Concepts*

Definitions in IR refer to core concepts that have yet to be defined, namely: *Document, document collection, information need and query* and finally *relevance*.

2.1 *Document*

A document is a set of information that is accessible and stored on computers. It constitutes the elementary information unit of a *document collection* in IR. It can have different formats: a text, a fragment of a text, a sound, an image or video. We are, however, only interested in textual documents. We also distinguish different *granularity* of a document considering the integral or a piece of the document (e.g., integral document vs. passage).

2.2 *Document Collection*

The definition of IR assumes the existence of a collection of documents or a corpus $\mathcal{C} = \{d_1, ..., d_n\}$. This corpus is usually large but finite, if we consider a commercial search engine, the corpus is comprised of the countless billions of pages crawled form the Web —rising efficiency concerns.

It assumes furthermore, that the corpus is provided to the IRS prior to the arrival of queries. This implies that offline processing may be conducted on the documents before relevance scoring. A corpus is thus considered *mostly* static, i.e., additions, deletions and alterations to the documents happen at a pace slower than the amount of preprocessing required by the system.

If documents in the same collection address different subjects, it is called a *generic collection*. For example, a Wikipedia collection. On the other hand, if all documents address a specific domain, it is called a *domain collection*. For instance, a medical collection.

2.3 *Information Need and Query*

A query is a sequence of words, usually a set of key words or a longer natural language sentence/question, chosen by the user in an attempt to express his/her information need. Ingwersen [100] identifies three fundamental types of information needs in IR:

1. Verification needs, where users want to verify or locate items that are *known* to them, e.g., search for a resource knowing its URL on the Web: go on the IRIT web site whose URL is `https://www.irit.fr`.
2. Conscious topical needs, where users want to clarify, review or pursue aspects of a *known* topic.

3. Muddled topical needs, where users want to explore some new *unknown* concepts. In this case, the information need is inherently variable and incompletely or ill-defined.

It is important to note that queries are not synonyms with information needs Taylor [228]. Queries are usually ill-formed or incomplete, they don't fully represent the information need that originally compelled the user to seek information, Belkin [13] call this "anomalous state of knowledge". Nevertheless, we usually abuse the terminology and use "query" as a metonym for the user's "information need" since it is the only tangible signal provided to the IRS in order to realized the ranking task. We only consider textual queries in the context of this dissertation even though a query can have different modalities: an image or a spoken query.

2.4 *Relevance*

Relevance is by far the most important concept in IR [100, 212]. It is the notion connecting the query, expressing the "user's information need", to the "goodness" of a document in the collection [129]. While seemingly intuitive, relevance is difficult to precisely characterize.

> *Everybody knows what relevance is. It is a "ya'know" notion, concept, idea —no need to explain whatsoever.* –Saracevic [219]

Debates about the precise meaning of relevance date back to the development of the first IR systems, since it underlies what such systems should return and how to evaluate their effectiveness. We retain the definitions from Saracevic [218] who tried to synthesise decades of investigation about the notion of relevance in information science, and accumulated a list of definitions presented here:

1. The correspondence between a document and a query;
2. The measure of the informativeness of the document to the query;
3. The degree of the relation between the document and the query;
4. The degree of surprise that a document provides, in relation to the user's need.

In practice, the relevance of a document is estimated by the degree of similarity of the query with the document content returned by the IRS. The goal of this latter is then to match two types of relevance:

1. **System relevance**, where the relevance estimation is solely based on the intrinsic characteristics of the queries and the documents —Does the content of the document match the information need?
2. **User relevance**, or *subjective relevance* it is the user-specific assessment about the document contents returned by the IRS that involves complex cognitive processes.

We refer the readers to [129] for a more exhaustive definition of relevance in the text ranking task.

3 *Evaluation in IR*

Evaluating the quality of an IRS is a key task in IR. A wide range of criteria can be considered, such as the capacity of the system to return the most relevant documents that satisfy the user's query, the response time, the storage footprint.

In the following, we present the standard community evaluation protocol and the necessary ingredients required to evaluate an IRS.

3.1 *The evaluation protocol*

The evaluation protocol provides the environment and defines the experimental scenario required to evaluate the quality of IR systems.

The Cranfield paradigm defined in Cleverdon [42] is a reference for evaluation in IR. The Text Retrieval Evaluation Conference (TREC), which has been running for three decades, is the first evaluation campaign based on this paradigm. It is organized every year by the U.S National Institute for Standards and Technology (NIST), and provides a standardized evaluation framework for a wide range of IR tasks such as ad hoc ranking, medical IR or question-answering.

TREC provides all the ingredients necessary to evaluate IR systems in the context of a given task, building what is known as the *test collection*, which is comprised of the three following elements:

1. A *corpus* or a *collection* of documents;
2. A set of information needs called *Topics*. It consists of a particular external representation of information needs. A typical ad hoc retrieval topic comprises three fields:
   - A title, which consists of few keywords that describe the information need, akin to a user query posed to a search engine;
   - A description, which is a longer well-formed natural language description of the information need;
   - A narrative, which is a paragraph that details the characteristics of the sought information that are not described in the title or description.
3. Relevance judgements, also called *qrels* consisting of a set of (q, d, r) triples, where r is the relevance label provided by a human assessor for the document d w.r.t query q. The relevance label r can either be a binary label indicating if d is relevant or not w.r.t q, or a graded-scale

from not relevant to highly-relevant; For instance a three-point scale (not relevant, relevant and highly-relevant). In both cases, they are the result of a costly human annotation process. NIST assessors manually evaluate the submissions of the participating teams in the campaign. For large-scale corpora, however, relevance judgements are obtained using *pooling*, where the assessors annotate only the top documents (without repetition) returned by the participating systems [25]. It is important to note that these relevance judgements represent the specific assessment of what is relevant or not to the assessor (or annotator). Relevance is not a "truth" (in a platonic sense) or an "inherent property" of a piece of text (with respect to an information need) that the assessor attempts to "unlock" [129]. Everyone can have a different notion of relevance, and many factors (e.g., the time of day and day of week that a label is given, fatigue, anchoring, exposure, left-side bias, task switching) can affect the quality of the judgements [233]; see [84] for a discussion of assessor agreement studied across many decades.

Thus, evaluating the effectiveness of an IRS for a target task consists in evaluating the results returned for each topic of the test collection by comparing the *system relevance* (scores) and the *user relevance* as estimated by the assessor. The global performance is estimated on the whole set of test topics based on evaluation metrics which we present in the next section.

3.2 *Evaluation Metrics*

Evaluation metrics are computed from relevance judgements to quantify the effectiveness of an IRS. By having means to quantify the effectiveness of an IRS, it becomes possible to compare different systems and make measurable progress in improving IR models.

During evaluation, each IRS produces a ranked list of results for all provided test topics, called a "run" or "submission" that is submitted for evaluation, in a TREC campaign for example. The qrels and the submitted run are then fed to an evaluation program (e.g., `trec_eval`) that automatically computes the evaluation metrics per topic, and then each metric is aggregated across all topics to obtain a global quality measure of the effectiveness of the IRS.

Below, we describe the commonly used metrics that are used to evaluate IR systems. Considering a ranked list $\mathcal{R} = \{(d_i, s_i)\}_{i=1}^{l}$ of length l of document-score (d_i, s_i) pairs with respect to a specific topic (query), we define the following metrics:

3.2.1 *Precision (P)*

Measures the capacity of a system to reject all documents that are not relevant w.r.t a query q or to only return the relevant documents. In other words, precision indicates the capacity of a system to minimise *noise*. It is given by the fraction of documents in the ranked list $\mathcal{R}$ that are relevant:

$$Precision(\mathcal{R}, q) = \frac{\sum_{(d_i, s_i) \in \mathcal{R}} rel(q, d)}{|\mathcal{R}|} \tag{1.1}$$

where $rel(q, d)$ is the binary relevance judgement of the (q, d) pair. Graded relevance judgements ought to be binarized with some relevance threshold before computing precision, e.g., in a three-grade scale, we can set $rel(q, d) = 1$ for both "relevant" and "highly-relevant" judgements.

Sometimes, precision is evaluated at a cutoff k, noted as *Precision@k* (*P@k*), where only the top-k documents in the ranked list $\mathcal{R}$ are considered.

3.2.2 *Recall (R)*

Measures the capacity of a system to find all relevant documents w.r.t a query q. It is a metric that evaluates the capacity of the system to minimise *silence*. It is given by the fraction of relevant documents, in the entire corpus $\mathcal{C}$, for q that are retrieved in the ranked list $\mathcal{R}$:

$$Recall(\mathcal{R}, q) = \frac{\sum_{(d_i, s_i) \in \mathcal{R}} rel(q, d)}{\sum_{d \in \mathcal{C}} rel(q, d)} \tag{1.2}$$

where $rel(q, d)$ is the binary relevance judgement of the (q, d) pair. Graded relevance judgements are binarized in the same manner presented for precision.

Same as precision, recall is also evaluated at cutoff k, noted *Recall@k* (*R@k*).

Recall and Precision have the advantage of easy interpretation, however they both share the same downsides: First, these metrics do not take into consideration relevance grades, they cannot separate a "relevant" document from a "highly relevant" one since relevance judgements are binarized. Second, they do not take into account the rank positions, i.e, two systems S_1, S_2 each ranking one relevant document only, where S_1 ranks the document first and S_2 ranks it last, both systems will have the same precision/recall. Yet, relevant documents appearing in the first rank positions are preferred by the user.

3.2.3 *Reciprocal Rank (RR)*

It is a measure based on the rank. It evaluates the number of documents to consider before finding the first relevant document in $\mathcal{R}$. It is defined as:

$$RR(\mathcal{R}, q) = \frac{1}{rank_i} \tag{1.3}$$

where $rank_i$ is the smallest rank number of a relevant document, i.e, if a relevant document appears at the first rank, $RR(\mathcal{R}, q) = 1$, $1/2$ if it appears at rank 2, etc. If no relevant document appears in the top-k ranks, then the query receives a score of 0.

RR has an intuitive interpretation, but it only considers the first relevant result returned. While this may be an appropriate metric for tasks where the user might be satisfied by with a single answer (e.g., question answering), it is usually a poor choice for ad hoc retrieval where users desire more than one relevant result (i.e, a ranked list of relevant documents).

3.2.4 *Average Precision (AP)*

Measures the average of precision scores at cutoffs corresponding to the rank of each relevant document, thereby combining aspects of both precision and recall. It is given by:

$$AP(\mathcal{R}, q) = \frac{\sum_{(d_i, s_i) \in \mathcal{R}} P@i(R, q).rel(q, d)}{\sum_{d \in \mathcal{C}} rel(q, d)} \tag{1.4}$$

where $rel(q, d)$ is binarized so that non-relevant documents do not contribute to the average. Similarly to precision and recall, average precision is usually evaluated at cutoff k, noted $AP@k$.

3.2.5 *Normalized Discounted Cumulative Gain (nDCG)*

Measures the quality of a ranked list of results. Unlike the metrics presented above, $nDCG$ was specifically designed for graded relevance judgements. We first define the Discounted Cumulative Gain (DCG):

$$DCG(\mathcal{R}, q) = \sum_{(d_i, s_i) \in \mathcal{R}} \frac{2^{rel(q,d)} - 1}{\log_2(i + 1)} \tag{1.5}$$

The *Gain* of a ranked document depends on both its relevance grade (i.e., the higher the relevance of a document, the higher its gain), and the rank at which this document appears (i.e., a highly relevant document appearing lower in the list is *penalized*—discounted gain). At last, $nDCG$ represents the normalized DCG (range $[0, 1]$) by the ideal gain as follows:

$$nDCG(\mathcal{R}, q) = \frac{DCG(R, q)}{IDCG(R, q)} \tag{1.6}$$

where $IDCG$ represents the gain of the best possible ranked list where all relevant documents appear in the order of their relevance grade (i.e., the documents with the highest relevance grade are ranked first, followed by the next highest relevance grade, etc). $nDCG$ is typically evaluated at cutoff k ($nDCG@k$) usually 10 or 20.

Up to this point, we presented the most common evaluation metrics used in IR to estimate the quality of a ranked list of results w.r.t a single topic. For a given test collection, the arithmetic mean across all test topics, is typically used to characterize the global performance of the IRS.

4 *Conclusion*

In this chapter, we presented an overview of the vast Information Retrieval (IR) domain and defined its fundamental concepts and the evaluation protocol allowing measurable progress in the domain.

We have seen that the core of IR lies in providing an ordered list of *relevant* documents that satisfy a query expressing the user's information need from large document corpora. Ranked textual document retrieval is a classic problem in information retrieval, as in the main task of the Text Retrieval Conference [241], and performed by commercial search engines such as Google, Bing, Baidu, or Yandex. Information retrieval researchers call this the *ad hoc* retrieval task. In order to complete this task, an IRS implements a model which evaluates the relevance score of each document w.r.t the query. Over the years, numerous models have been proposed and used in IR systems.

We present in the next chapter an overview of these models and their development through time. We will present a brief history of IR models from the first IR statistical methods frequently based on Bag-of-Words (BoW) representations, the rise of learning to rank approaches, the advent of deep learning, and the introduction of dense semantic representations, to finally, the arrival of BERT and transformer-based architectures for contextualized representations.

2

A historical overview of IR models

1 *Introduction*

The goal of the *ad hoc* retrieval task is to generate a ranked list of textual documents retrieved from a corpus in response to a user's query. The documents can be full-length documents or parts of a document (e.g., a paragraph). These texts are ranked by their relevance with respect to the query estimated by an IR model, also called the matching model, which provides a theoretical framework and a formal interpretation of the notion of relevance [202, 212].

Over the past three-quarters of a century, the IR community developed a plethora of models for estimating query-document relevance based on different strategies and following advances in related research areas. The context of our thesis focuses on recent neural IR models based on transformers [236], of which BERT [58] is the most popular instantiation. Except for a few tasks, including automatic processing of natural language, these models have revolutionized the fields of natural language processing, information retrieval, and, more generally, human language technologies, including technologies to process, analyze and manipulate human language data. However, in order to better grasp how transformer-based models have revolutionized ad hoc ranking, it is necessary to understand how IR models evolved to reach this point in history. To do so, we provide a high-level overview of the major developments in the information retrieval field in this chapter.

We first introduce a few traditional IR approaches in section 2.2. The decades of insights from these IR models not only inform the design of our new neural-based approaches, but these models also serve as important baselines for comparison [164]. They also highlight the various desiderata that we expect IR models to incorporate. Then, we present learning to rank approaches based on human-crafted features in section 2.3 before introducing deep neural networks to IR models in section 2.4. In the context of deep learning approaches, we further distinguish "pre-BERT" models, which came before the introduction of BERT, and "BERT-based" or, more generally, transformer-based models.

A thorough review of IR approaches that came before the BERT revolution is out of the scope of this thesis. We refer readers to the following books for a detailed overview of traditional IR:

— Modern Information Retrieval, by Baeza-Yates and Ribeiro-Neto [7]

— Information retrieval: the early years, by Harman [85]

Similarly, for neural network fundamentals:

— Deep Learning with Python, by Chollet [38]

— Deep learning, by Goodfellow et al. [80]

Finally, we refer readers to these surveys on pre-BERT neural IR models:

— Neural Information Retrieval: At the End of the Early Years, by Onal et al. [180]

— An Introduction to Neural Information Retrieval, by Mitra and Craswell [164]

2 *Traditional IR*

Traditional (classical) query-document matching methods, also known as "keyword search", encompass a broad class of models that evaluate the relevance between a query and a document based on *exact term matching*. Meaning that only terms matching *exactly* between the query and the document are considered for relevance scoring. Typically, stemming is performed before matching to reduce inflected words to their stem (e.g., "hunting", "hunts", and "hunter" are reduced to the stem "hunt") and thus, ensuring that syntactic variations of the same stem in the queries and the documents match ("hunter" matches "hunting").

Nearly all classical methods suppose query-term independence, that the contribution of each query term to the relevance score is considered independently [1]. This is commonly achieved with Bag-of-Words representation, where a query (document) is represented with a set of independent terms $\{t_1, ..., t_n\}$. Hence, the relevance score of a document d w.r.t a query q is given by the sum of the individual relevance scores of each exactly matching query-document term $t \in q \cap d$, or:

$$R(d, q) = \sum_{t \in q \cap d} f(t) \tag{2.1}$$

where f is a term-weighting function that models the importance of a term — also called "salience"— based on its associated statistics such as *term frequency* (how many times a term occurs in a document), *document frequency* (how many documents the term occurs in), and *document length* (the length of the document the term occurs in). Considering f and these intrinsic statistical factors, both queries and documents can be represented with BoW using *sparse vectors* of the size of the corpus vocabulary $\mathcal{V}_\mathcal{C}$, where each value represents the weight of each term in the vocabulary $< f(t_1), ..., f(t_{|\mathcal{V}_\mathcal{C}|}) >$.

1. We also find in the information retrieval literature works that attempt to capture dependencies between query terms [158].

The relevance score can, then, be cast as the inner product between the query and document vectors. This corresponds to the *vector space* model proposed by Salton et al. [214] with the ubiquitous tf-idf (term frequency- inverse document frequency) weighting function and cosine similarity for relevance scoring.

The vector space model spurred a considerable body of work dedicated to the exploration of different term-weighting schemes based on statistical properties of the terms that are easy to compute [213]. Inverted indexes (inverted files or lists) are nearly always employed to encode these properties: term frequencies, term positions, document structure information, various forms of document metadata (e.g., document length), etc; see [272] for an overview. Enabling low latency keyword search. Therefore, classical methods are valued for their efficiency and are still widely employed today, both in academia and industry. The Okapi BM25 (Best Match 25) is the most widely used and recognized classical model due to its performance [206, 46, 203]. BM25 is based on the theory of probabilities (the probabilistic IR framework) and estimates the relevance of a document d w.r.t a query q as follows:

$$BM25(d,q) = \sum_{t \in q} idf(t).\frac{(k_1+1).tf(t,d)}{k_1.((1-b)+b.\frac{|d|}{avgdl}) + tf(t,d)} \tag{2.2}$$

where $tf(t,d)$ is the term frequency of t in the document d, i.e, the number of occurrences of the term t in d. The model has two free parameters k_1 and b: k_1 controls the term frequency tf, and b is used for the length normalization performed in the denominator of the second component where $|d|$ is the document length, and $avgdl$ is the average document length in the whole collection. Finally, $idf(t)$ is the inverse document frequency of the term t, computed as:

$$idf(t) = \log \frac{|\mathcal{C}| - df(t) + 0.5}{df(t) + 0.5} \tag{2.3}$$

with $df(t)$ being the number of documents that contain t and $|\mathcal{C}|$ is the total number of documents in the corpus.

Beyond the original BM25 formulation by Robertson et al. [207], we find numerous variants in the literature [234, 108] which may lead to differences in effectiveness.

While classical methods can model term importance quite effectively based on statistical factors, relying on exact matching becomes a limitation when terms in queries and documents don't match. It is known that authors often use a large vocabulary when writing their documents, and inevitably they use different terms to describe the *same* concept. Sometimes they use equivalent terms (synonyms, acronyms), sometimes they use similar terms (generic or specific), for example: "Coronavirus", "SARS-CoV-2" and "COVID-19".

As for the user queries, they are usually short or incomplete insofar as the user has a vague idea of the information sought, especially on the web. For example, a user searching for "movies similar to The Matrix", here exact matching is powerless since the expected information is a list of movie titles that will, most certainly, not contain any of the query terms. This is known as the "vocabulary mismatch problem" [70], a long-standing challenge in IR. Three threads of research tackled this problem by: (1) expanding the queries to better match the documents, (2) enriching the document representations to better align with the query representations, or (3) moving away from exact matches.

QUERY EXPANSION The first thread of research proposes to enrich query representations with query expansion techniques [31]. In relevance feedback, the original user query is augmented with significant terms derived from documents *judged* relevant by the user and reinforces the importance (weight) of these terms in the new representation of the query [23]. Formulations by Rocchio [208] and Robertson and Jones [205] are the most popular relevance feedback approaches. In pseudo-relevance feedback [47], also called blind feedback, the user does not intervene, and query expansion is performed by exploiting the top-k documents (*assumed* to be relevant) returned by the IRS in response to the initial query. Query expansion can also be performed using external resources, which can be a thesaurus, an ontology, etc. Voorhees [239] uses lexical-semantic relations from WordNet[163] for query expansion. Query expansion is still an active line of research, and we discuss in Chapter 5 recent works integrating pre-trained language models.

DOCUMENT EXPANSION Similarly, enriching document representations was investigated to bridge the lexical gap between queries and documents. Pseudo-relevance feedback has been successfully employed in specific tasks such as speech retrieval [223] and short-text retrieval [61]. We also find document expansion models [11] based on WordNet concepts, similarly to query expansion. However, this thread of research is less popular than query expansion. Nevertheless, document expansion has recently regained interest with transformers; we discuss this in Chapter 5.

BEYOND EXACT MATCHING A different thread of research attempts to alleviate the vocabulary mismatch problem by going past exact matching. Berger and Lafferty [19] were among the first authors to exploit the statistical translation model to solve the vocabulary mismatch problem. Their approach learns translation probabilities between the terms of the query and the document from a reference (train) corpus based on word co-occurrences. Other approaches attempt to perform matching in some semantic space induced from the data, for instance, based on Latent Semantic Analysis [56] or Latent

Dirichlet Allocation [245]. Despite the fact that these approaches have not been widely adopted, there are clear connections between this line of research and the recent dense retrieval movement that learn dense context-based representations for ranking, which we cover in Chapter 6.

Traditional IR models have served as the workhorse of most modern search systems over the past several decades. They are considered as a mature technology that can be used as reliable infrastructure that can robustly deliver high query throughput at low query latency on large corpora [129]. Nonetheless, these methods have limitations. On the one hand, the widely used BoW representation considers the terms in documents and queries as meaningless strings of characters. It is therefore opposed to the reality of natural (human) language, which requires that terms be linked by semantic relations. The position of the terms and the relationship with other terms in the document are ignored. On the other hand, these models are based on exact matching that considers documents with no query-term occurrences as irrelevant. However, documents can be relevant without having any lexical overlap with the query. For example, synonyms or acronyms can be used instead of the query terms. Hence, soft or semantic matching enabled by continuous representations in neural networks is needed to cope with the limitations of exact matching by addressing a variety of linguistic phenomena, including synonyms, paraphrases, term variations, and different expressions of similar intents. However, soft matching and the deep learning movement are still a phase away in the development of IR models.

3 *The Rise of Learning To Rank*

The next major development in IR models, beginning in the late 1980s [48], is the application of supervised machine-learning models to learn ranking functions. This category of models, referred to as "Learning To Rank", rely extensively on hand-crafted, manually-engineered features, Lin et al. [129] classify these features into statistical properties of terms and intrinsic properties of documents:

— Statistical properties of terms include the usual functions of term frequencies, document frequencies, document lengths, etc, used in classical methods. And even scores computed using these classical approaches, such as BM25, are typical features in a learning-to-rank setup.

— Intrinsic properties of documents include a wide range of measures such as the ratio of HTML tags, the editorial quality, spam scores, hyperlinks in the web collections, and social signals (number of views, number of followers, etc). User-behavior based features and click logs are also valuable features for search systems.

At a high level, learning-to-rank methods can be divided [137, 124] into three major approaches, based on their loss function: *pointwise* approaches, *pairwise* approaches, and *listwise* approaches.

Pointwise approaches. Formulate the ranking problem as classification or regression. The model is fed with a features vector corresponding to a query-document pair and trained to approximate the relevance judgement of the input pair. Consequently, a pointwise approach does not consider the inter-dependency among the returned documents.

Pairwise approaches. The model receives a pair of feature vectors X_i and X_j corresponding to a pair of documents (q, d_i) and (q, d_j), and thus trained to learn a *preference*, i.e, the property wherein d_i is *more relevant than* (preferred over) d_j. Consequently, a pairwise loss function measures the inconsistency between the true *preference* and the model's *preference*, and the objective is to rank the documents according to their relevance grade. However, pairs of documents related to the same query are processed independently.

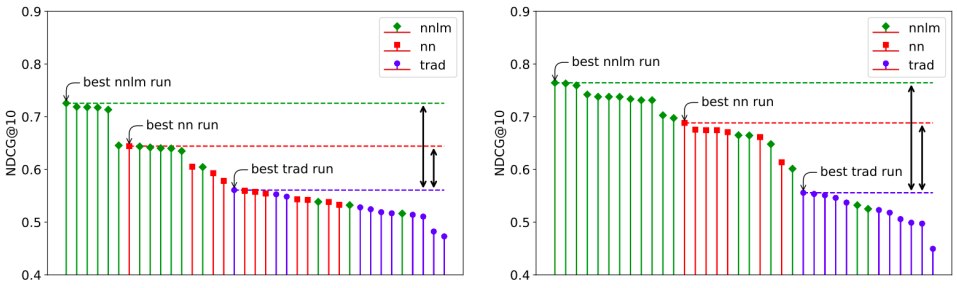
Listwise approaches. Consider the entire set of documents related to a query. The model is trained to approximate the relevance judgements of each document w.r.t to the query. With a listwise approach, a ranking metric such as normalized discounted cumulative gain (Section 3.2) can be directly optimized.

Learning to rank achieved great success in many IR applications [26, 29, 60, 34, 224] and reached its zenith in the early 2010s, with the development of models based on tree ensembles [27]. Nonetheless, learning-to-rank models' effectiveness is closely related to the quality of the relevance features they use, which are mostly hand-crafted. These features are time consuming since their engineering involves expensive human efforts. Moreover, this process is often data-specific and could require domain expertise. Despite the possibility of automated feature selection [78], feature engineering is still an expensive step in the design of learning-to-rank models.

In order to address this issue, it would be of great value if the ranking model could automatically learn relevance features without human intervention. The deep learning revolution that came right after enables, precisely, to learn abstract representations from raw data.

4 *The Emergence of Deep Learning*

In the early 2010s, advances in neural network models using multiple hidden layers sparked the deep learning revolution leading to dramatic



(a) Document retrieval task

(b) Passage retrieval task

Figure 2.1 – nDCG@10 results, broken down by model type: "nnlm" use language models such as BERT, performed best on both tasks, other pre-BERT neural ranking models "nn", and traditional non-neural models "trad" had relatively lower performance in this track. [45]

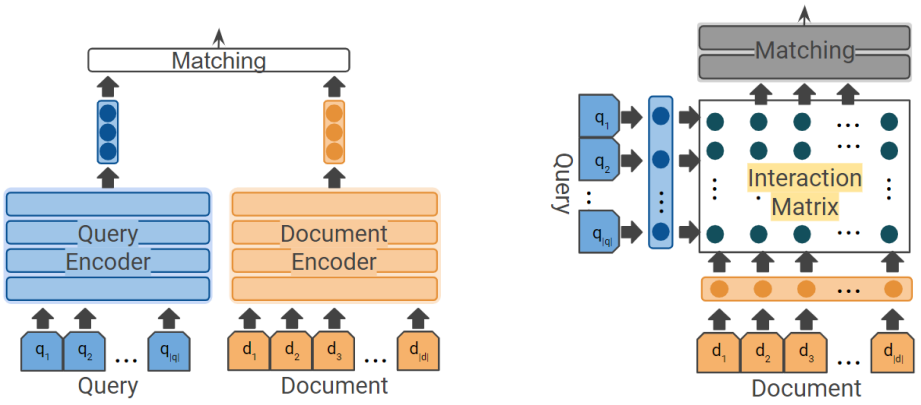
progress in computer vision, speech recognition, and machine translation tasks [119]. Inspired by this success, a new wave of IR models fueled by deep neural networks emerged with the intention of achieving breakthrough performance as in these other fields. The promise of deep learning was, on the one hand, to obviate the need for expensive manually-engineered features, which is a major drawback of learning to rank models. On the other hand, accomplish soft matching by learning distributed representations to overcome the limitations of exact matching.

Following the active growth in the area of deep learning, which is continuously introducing new architectures and training regimes, research in the neural IR field has been advancing at an increasing pace. After about half a decade of model progression based on deep neural networks, a significant leap in performance was finally brought by the introduction of BERT [58] to IR in 2019 by Nogueira and Cho [173], marking the beginning of a new era in the field. Indeed, the evaluation of the submissions to the TREC Deep Learning Track 2019 [45] which was the first large-scale evaluation of ranking models, revealed that BERT-based models achieved a significant jump in performance compared to pre-BERT models, as shown in Figure 2.1.

We start by presenting an overview of pre-BERT ranking models in section 4.1, to then discuss the start of the "BERT revolution" in section 4.2.

4.1 *Pre-BERT neural ranking models*

The computation of the relevance score of a document w.r.t a query, which is the core problem in ad hoc retrieval, can be formalized as a text matching problem. Following the notation in [83], the degree of matching is measured



(a) Representation-based (b) Interaction-based

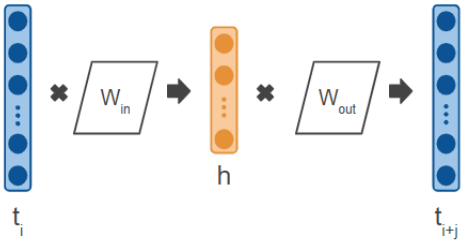
Figure 2.2 – Two classes of pre-BERT neural ranking models: (a) Representation-based models learn vector representations of queries and documents that are compared using simple metrics, such as cosine similarity to compute relevance scores, and (b) Interaction-based models explicitly model term interactions in a similarity matrix that is further processed to compute relevance scores

as the score produced by a scoring function F based on the representation Φ of the query q and the document d:

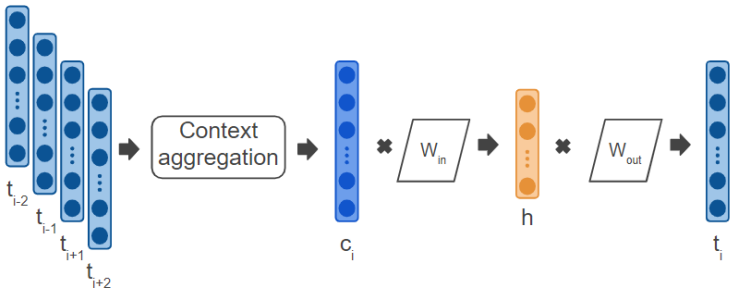
$$match(q, d) = F(\Phi(q), \Phi(d)) \tag{2.4}$$

where Φ is a function that maps each of the query and document to a vector representation, and F is a scoring function based on interactions between the two representations. This text matching problem is closely related to the problem of semantic matching of two sentences in Natural Language Processing (NLP), such as paraphrase identification. Models for these tasks share many architectural similarities, and there has been cross-fertilization between the NLP and IR communities in this regard. However, relevance matching and semantic matching are not synonyms, and one major difference is that inputs for computing semantic similarity are symmetric, i.e., $match(s_1, s_2) = match(s_2, s_1)$, whereas query/document inputs cannot be swapped. Nevertheless, recent developments in learned dense representations for ranking are bridging the two threads of work closer , as we will see in Chapter 6.

Depending on the choice of the two functions F and Φ, pre-BERT neural ranking approaches can be categorized into two main types: *representation-based* and *interaction-based* models. The general model architecture for both types are illustrated in Figure 2.2. The first category of models focuses on *independently* learning good distributed (dense) vector representations for queries and documents, which can be compared using a simple vector



(a) Skip-Gram



(b) Continous Bag Of Words (CBOW)

Figure 2.3 – The two configurations of Word2Vec: (a) Skip-Gram and (b) CBOW. The architecture is a neural network with a single hidden layer whose size is much smaller than that of the input and output layers. Both models use one-hot representations of terms in the input and the output. The learnable parameters of the model comprise the two weight matrices W_{in} and W_{out} that correspond to the embeddings the model learns for the input and the output terms, respectively. The Skip-Gram model trains by minimizing the error in predicting a context term t_{i+j} given the central term t_i. The CBOW model, in contrast, predicts the central term t_i from a bag of its neighbouring terms; we consider a context window of size 5, including 2 terms before and after the central term $j \in \{-2, -1, +1, +2\}$

similarity function (e.g., cosine similarity) to compute the relevance scores. In these models, Φ is a complex representation mapping function, whereas F is a relatively simple matching function. In contrast, the second category of models focuses on capturing term interactions using an *interaction matrix* based on term-to-term similarities between the query and document terms. This interaction matrix is then fed through a neural network to produce a relevance score. Here, Φ is generally a simple representation function, while F is a complex deep neural network.

Both types of neural ranking models consider only the embeddings of query and document terms as input and importantly no additional hand-crafted features are needed as opposed to learning-to-rank methods. For term representation, distributed word embeddings have been widely adopted

in pre-BERT IR models. Words are represented using real-valued vectors to capture fine-grained semantic and syntactic regularities in order to overcome the limitations of BoW representations. Word2Vec [161, 162], is certainly the most popular method for computing distributed word representations, where each word embedding is learned from its neighbours within a fixed-size window over the text, using a is a simple one hidden layer neural network. Mikolov et al. [162] proposes two configurations of this model, namely CBOW (Continuous Bag of Words) and Skip-Gram, as shown in Figure 2.3. Both input and output terms are one-hot encoded, meaning that every word in the vocabulary $\mathcal{V}$ is represented by a binary vector $\vec{v} \in \{0,1\}^{|\mathcal{V}|}$ of vocabulary size —each position corresponds to a word in the vocabulary— where only one value —corresponding to the position of the word in the vocabulary— is set to one while all other values are set to zero. The CBOW configuration is trained to predict a term t_i based on the terms in its symmetric context c. First, input representations of the terms in the context c are aggregated (sum or average) and then fed to the model. After training, the vector representation of the central term t_i is provided by the hidden layer h. On the other hand, Skip-Gram is trained to predict the surrounding symmetric context from the central term t_i.

In addition to the local context window, GloVe [189] additionally uses a *word-to-word* matrix to capture global word co-occurrences. GloVe leverages statistical information about a text dataset by training the model only on the nonzero elements of the co-occurrence matrix rather than on the entire sparse matrix or on individual context windows of a large corpus. It combines the global and the local contexts during training to learn word embeddings.

4.1.1 *Representation-based models*

This category of models (Figure 2.2a) adopt a Siamese-type of neural architecture to learn vector representations for queries and documents separately. This separation in representation allows document representations to be computed offline. The pioneering work of Huang et al. [95] proposes Deep Structure Semantic Model (DSSM) for ad hoc web search. DSSM uses two symmetric deep stacked MLPs —with shared parameters— to produce vector representations from the character n-grams of each of the query and document inputs. At search time, relevance scores are computed using cosine similarity between query-document representations. Shen et al. [222] extends the DSSM model by incorporating convolutional neural networks (CNNs) to capture context. Rather than learning text representations as part of the model, the Dual Embedding Space Model (DESM) [168] computes relevance scores by aggregating cosine similarities across all query-document term representations obtained with pre-trained Word2Vec embeddings [162]. In the

context of transformers, the representation-focused paradigm has regained the attention of the IR community, as covered in Chapter 6.

4.1.2 *Interaction-based models*

In this type of models (Figure 2.2b), query-document term interactions are explicitly captured in order to extract matching signals early in the model [83]. The term-level similarities are typically represented within an interaction matrix where the rows correspond to the query terms and columns correspond to document terms. Each cell $m_{i,j}$ in the matrix contains, then, the similarity between the embedding of the i-th query term and the embedding of the j-th document term, usually computed in terms of cosine similarity. Building on the interaction matrix, the general approach consists in extracting relevance signals from the term similarities, which are then combined and processed, often using pooling operations and/or stacked feed-forward networks, to produce relevance scores.

Guo et al. [83] propose DRMM, which cast the similarity matrix interactions into matching histograms. Similarly, KNRM [250] uses Gaussian kernels to learn differentiable "softer" histograms that allow the embeddings to be learned during training. Other models consider specific assumptions about the input. We find, position-aware models such as MatchPyramid [186, 185], PACRR [96], CO-PACRR [97], and ConvKNRM [54] which use additional architectural components (e.g., CNNs, Max Pooling) to extract hierarchical matching signals from the term-level interaction matrix. Instead of a position-shared weighting scheme, ANMM [254] proposes a value-shared weighting scheme for combining different matching signals, with an *attention* gating function for learning question term importance. Other models propose to improve the input embeddings using LSTM-based contextualization in POSIT-DRMM [156], or by incorporating entity embeddings in EDRM [139]. In order to handle long documents, Fan et al. [65] propose a hierarchical neural matching model (HiNT) which splits documents into passages and uses two stacked components: a *local* matching layer that captures passage-level matching patterns and a *global* decision layer that performs interactions between the different passage-level signals to compute document-level relevance features.

4.1.3 *Hybrid models*

Mitra et al. [165] combines the representation-based and interaction-based paradigms in a new hybrid model DUET. The authors emphasize the importance of *exact lexical matching* in deep neural models for IR, and argue that web search requires both exact and soft (semantic) matching. The DUET model, thus, comprises two parallel components: the *local* module for exact matching that focuses on learning interaction signals between the query-document

inputs (interaction-based) and the *distributed* modules for semantic matching based on the distributed representations of the inputs (representation-based). The relevance score is obtained by aggregating the scores from the local and distributed modules.

Nie et al. [172] show in their empirical study that the interaction-based neural architectures generally lead to better results than the representation-focused architectures in information retrieval tasks. The promising performance of interaction-based models is partly attributed to their capacity to learn the local interaction patterns rather than learning global representations in representation-based models. Though interaction-focused models can be computationally expensive, as they require pairwise similarities between embeddings of query and document tokens, they have the advantage of learning the matching signals from the interaction of two inputs at the very beginning stages.

Most of the models covered in this section, up to this point in history, operate under the data regime where large corpora of documents or queries are available but only limited (or even no) labelled data. However, neural IR models with a significantly large number of parameters require large amounts of training (labelled) data for supervision. Alternative training schemes, e.g., using weak supervision [57] or adversarial learning [242, 43], were developed in an attempt to address this challenge.

After nearly a decade of active research on the neural IR field throughout the 2010s, Lin [126] suggests to pause in a moment of self-reflection with respect to the hype surrounding neural ranking approaches: Under the data regime, are neural IR models making concrete progress? In other words, can neural IR models beat traditional IR models in the absence of vast quantities of training data available from behavior logs that are only accessible to researchers in the industry (with rare exceptions)?

In response to Lin's skepticism, Yang et al. [256] conducts a rigorous evaluation of several neural ranking models in comparison to a strong Bag-of-Words search baseline with well-tuned RM3 query expansion on the standard TREC Robust04 benchmark [240]. At least under the data regime, the authors find that most neural ranking models were unable to outperform the traditional baseline. They claim that at least some of the gains reported in the literature are illusory due to comparisons to weak baselines. While many of the papers cited above report significant improvements when trained on large, proprietary datasets (many of which include behavioral signals), the results are difficult to validate, and the benefits of the proposed methods are not broadly accessible to the community [129]. However, BERT was about to enter the research scene.

Table 2.1 – State-of-the-art results on the MS MARCO passage ranking leaderboard, in January 2019, showing the effectiveness of the newly introduced BERT model compared to pre-BERT models.

	MS MARCO Passage	
	Development	Test
Method	MRR@10	MRR@10
BM25 (Microsoft Baseline)	0.167	0.165
KNRM [250]	0.218	0.198
Conv-KNRM [54]	0.290	0.271
IRNet [173]	0.278	0.281
BERT_{large}	0.365	0.358

4.2 *The introduction of BERT*

The release of BERT [58] in October 2018 was an event described as marking the beginning of a new era in NLP. BERT broke performance records on many language-based tasks. The public availability of the model versions, that were already pre-trained on massive datasets, enabled researchers to use this powerhouse as a readily-available component in their solutions — saving time, knowledge, and especially resources that would have been spent on training a language-processing model from scratch, which are unattainable for small research groups. This ready-to-use availability paved the way for BERT to dominate the most popular NLP leaderboards (e.g., GLUE benchmark). In January 2019, Nogueira and Cho [173] reports the first application of BERT to IR. The authors combined the power of BERT with the availability of large training data from the MS MARCO passage ranking collection [9] to break records on the the MS MARCO passage ranking leaderboard [2]. The task consists in ranking passages (document extracts) from web pages in response to user queries, in the form of natural language questions, sampled from Bing search logs. Table 2.1 shows state-of-the-art (SOTA) performance on the MS MARCO passage ranking leaderborad with pre-BERT models, by January 4th, 2019, compared to Nogueira's submission. IRNet, previous pre-BERT SOTA on the MS MARCO leaderboard, submitted on January 2nd, was outperformed, 5 days later only, by BERT with about 30% improvement.

This big leap in performance was a long-awaited breakthrough in the neural IR field, kicking off what Lin et al. [129] call the "BERT revolution" for document ranking. BERT was, indeed, met with instant enthusiasm by the IR community, and research building on the insights of Nogueira and Cho [173] grew rapidly to apply, extend, adapt and understand this

2. https://microsoft.github.io/msmarco/

new model. Going back on the skepticism expressed in [126] one year earlier, Lin [127] acknowledges that deep transformer models, heavily pre-trained via language modeling tasks, have "significantly" and "substantially" improved the effectiveness of document retrieval, *even in the absence of vast amounts of training data*. Besides, the availability of MS MARCO to researchers today —thanks to the generosity of Microsoft, since the creation of such a dataset is well beyond the resources available to NIST, other TREC-like campaign organizers, and academic research groups [127]— alleviates the data availability issues for academic researchers, allowing model exploration in a "high-resource data regime"; only available to researchers in the industry until then.

The next part of this thesis is dedicated to the products of the so-called "BERT revolution" for IR.

5 *Conclusion*

This chapter focused on the major developments in IR over the past three-quarters of a century. We first presented early IR models based on exact matching and the subsequent expansions that attempted to address the limitations of exact matching. These classical methods are known and valued for their efficiency and are still widely used in both academia and industry. Then, we moved to the next family of IR methods based on supervised machine-learning techniques. This family is known as Learning to Rank (LTR) and relies heavily on manually-engineered features. The next significant development in ad hoc ranking was the advent of Deep Learning, with the promise of (1) obviating the need for laborious hand-crafted features addressing the major issue of LTR approaches and (2) building continuous vector representations enabling soft semantic matching in contrast to exact matching. Finally, the arrival of BERT in 2018 achieved unprecedented success in many NLP and IR tasks, starting the "BERT revolution".

Four years later, BERT —which remains today the most popular instantiation– and related models such as XLNet [258], RoBERTa [138], T5 [198], and many more that have followed, still dominate the IR research landscape. In the next part of this dissertation, we discuss the progress of BERT-based IR models.

Part II

State Of The Art Overview

3

Overview of BERT

1 *Introduction*

The **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) is a product of Google research [58] released at the end of 2018 and by far the most famous pre-trained language model.

Like nearly all scientific advances, BERT was not developed in a vacuum, but built on top of several previous innovations that were bubbling in the NLP community, including but not limited to the transformer architecture [236] and the idea of self-supervised pre-training based on language modeling objectives previously explored by ULMFiT (Universal Language Model Fine-Tuning) [93] and ELMo (Embeddings from Language Models) [190]. Both ideas were already combined in the OpenAI Transformer used in GPT (Generative Pre-trained Transformer) [196], and the additional idea of bidirectional encoding resulted in BERT.

We present in this chapter a high level overview of BERT in Section 3.2 where we define the concept of *contextual representation* and the mechanisms BERT leverages to attain it. We then present the original *Transformer* architecture, in Section 3.3 from which BERT borrows its core context encoding component. We move, in Section 3.4, to the *pre-train then fine-tune* paradigm used in BERT to first gain general language understanding from pre-training on unlabelled data and then transferring it to down-stream tasks with fine-tuning. Next, we detail the composition of BERT's input, in Section 3.5, and present an overview of the different pre-trained BERT configurations which are publicly available in Section 3.6. Finally, we present some studies which tried to understand BERT's inner workings in Section 3.7.

2 *BERT architecture*

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a deep neural network for building contextualized representations. Its architecture consists of a multi-layer *bidirectional transformer encoder*, as illustrated in Figure 3.1. It takes as input a sequence of vector embeddings –representing a sequence of tokens– denoted with:

$$[E_{[CLS]}, E_1, E_2, ..., E_{[SEP]}] \tag{3.1}$$

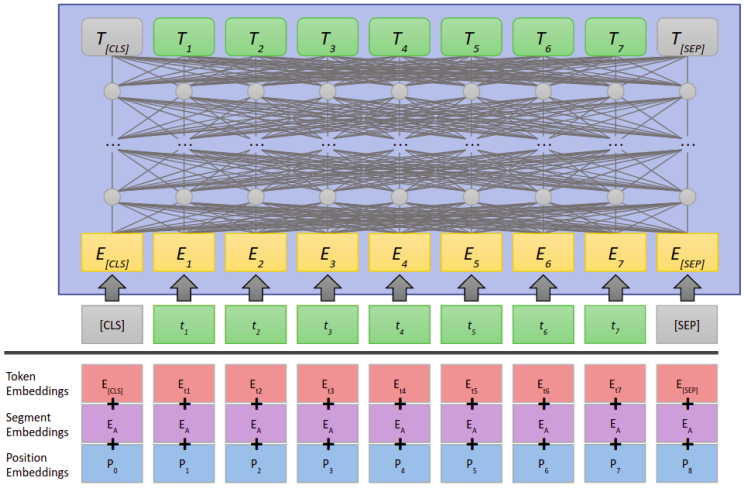


Figure 3.1 – The general architecture of BERT. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. Modified from a diagram by Jimmy Lin (https://twitter.com/lintool/status/1285599163024125959).

including two special tokens [CLS] and [SEP] added at the start and the end of the input sequence, respectively. After going through the stacked transformer-encoder layers, the model, finally, outputs a sequence of contextual embeddings, denoted as:

$$[T_{[CLS]}, T_1, T_2, ..., T_{[SEP]}] \tag{3.2}$$

BERT followed the recent advances in NLP which carry with them a new shift in token representation. Not long before, static word-embeddings were a major force in how NLP and IR models deal with language. Methods like Word2Vec [162] and GloVe [189] have been widely used in pre-BERT models (section 4.1). Such word representations provide context-independent representations, i.e., a word is assigned the same embedding independently from the context it appears in. Yet, if we consider the word "bank" appearing in "a river bank" and "bank of America", the first occurrence carries a different meaning than the second. Consequently, they should have different embeddings. This idea of context-dependent representations was implemented in ELMo [190] from which BERT draws many ideas: the goal of contextual embeddings is to capture complex characteristics of language (e.g., syntax and semantics) as well as how meaning varies across linguistic contexts (e.g., polysemy).

BERT implements these ideas by taking advantage of the powerful transformer architecture [236], as opposed to ELMo's use of LSTMs (Long Short Term Memory).

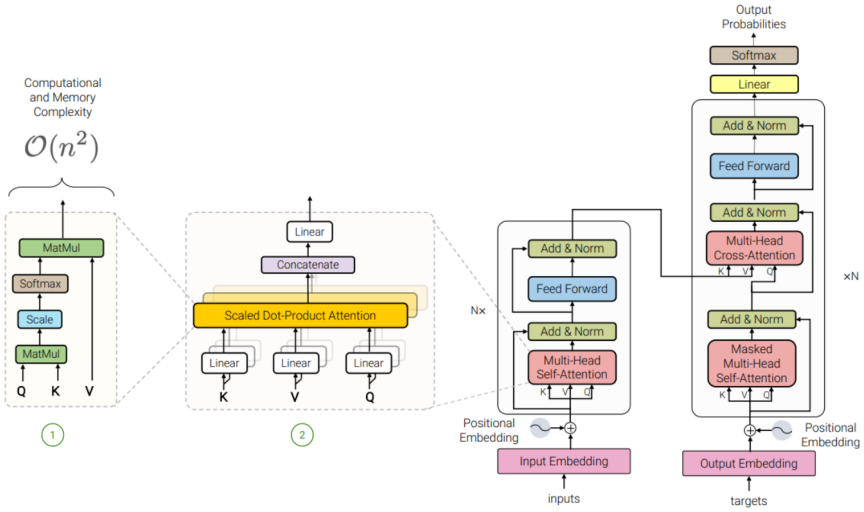


Figure 3.2 – Architecture of the original transformer [236]. Diagram by [226].

3 *The Transformer architecture*

The Transformer architecture was proposed by Vaswani et al. [236] as a new sequence-to-sequence architecture, which transforms an input sequence into another output sequence of tokens, based on *attention mechanisms*. Transformers proved to be more effective than the popular LSTM architecture on many tasks such as machine translation —by better handling long-term dependencies— and grew to replace them over time.

Figure 3.2 illustrates the full transformer architecture with the encoder part (left) and the decoder part (right) that can be stacked multiple times, which is indicated by $(N\times)$ in the figure. We see that the modules consist mainly of *multi-head attention* and feed forward layers. The input and output sequences are first embedded into an n-dimensional space (token embeddings). As opposed to recurrent networks (e.g., LSTMs) which can remember how the sequence is fed into the model, the transformer architecture needs an additional mechanism to map each token in the sequence to a position. This is realized by adding positional encoding to the token embeddings.

On the encoder side, the input sequence token embeddings are fed through a multi-head self-attention layer. For each token, self-attention look at all positions in the input sequence for cues to better represent this token. Considering a single attention head h and $X \in \mathcal{R}^{l\times n}$ the n-dimensional embeddings corresponding to the l tokens in the input sequence (or output vectors from the lower encoder layer), the embeddings are first projected via the three layers of the attention head h: the *Query* layer W_h^Q, the *Key* layer W_h^K and the *Value* layer W_h^V to the query, key and value vectors. The query and key inputs are used to compute the scaled dot-product attention scores which estimates

the "importance" of each token in the sequence to the current token. These scores are then used to weight the value vectors to obtain the new token representations as follows (see diagram ① in Figure 3.2):

$$Attention_h(X) = softmax\left(\frac{(X \cdot W_h^Q) \cdot (X \cdot W_h^K)}{\sqrt{D_K}}\right) \cdot W_h^V \tag{3.3}$$

where D_K refers to the dimension of the key vectors.

This is computed for each head $h \in H$, H being the total number of attention heads. The multi-head mechanism refines the self-attention layer in two ways: (1) It expands the model's ability to focus on different positions, and (2) it gives the attention layer multiple "representation subspaces" since there is a total of H sets of Query/Key/Value weight matrices (in the original Transformer formulation $H = 8$). Each of these sets is randomly initialized. Then, after training, each set is used to project the input embeddings (or vectors from lower encoders/decoders) into a different representation subspace. The final output of the multi-head self-attention is obtained by concatenating the outputs of all the heads and projecting the result through the output layer W^O (see diagram ② in Figure 3.2):

$$Y = concat_{h=1}^{H}(Attention_h(X)) \cdot W^O \tag{3.4}$$

The resulting representation is further processed with a feed forward network along with residual connections and normalization layers. Finally, the output of the last encoder of the stack is then transformed to a set of attention vectors key $K_{encoder}$ and value $V_{encoder}$. These are then transmitted to the decoder.

On the decoder side we have the *masked* multi-head self-attention, which is equivalent to the encoder's multi-head self-attention with causal masking: the scaled dot-product scores are masked to prevent tokens at position i from attending to future positions $j > i$ as follows:

$$Attention_h(X) = softmax\left(\frac{(X \cdot W_h^Q) \cdot (X \cdot W_h^K)}{\sqrt{D_K}} \cdot Mask_{causal}\right) \cdot W_h^V \tag{3.5}$$

where

$$(Mask_{causal})_{i,j} = \begin{cases} 0 & \text{if } j > i, \\ 1 & \text{else} \end{cases} \tag{3.6}$$

The encoder's key and value vectors ($K_{encoder}$ and $V_{encoder}$) are used in each decoder layer at the multi-head cross-attention level, also known as encoder-decoder attention, which helps the decoder focus on relevant positions in the input sequence, altering the original self-attention mechanism as follows:

$$Attention_h(X) = softmax\left(\frac{(X \cdot W_h^Q) \cdot (K_{encoder})}{\sqrt{D_K}}\right) \cdot V_{encoder} \tag{3.7}$$

HALF A TRANSFORMER IS ENOUGH OpenAI GPT [196] is another intellectual ancestry of BERT, while BERT uses only the "encoder half" of a full transformer, GPT is a "decoder-only" transformer. GPT is pre-trained to predict the next word in a sequence based on what it has already seen (language modeling). Thus, the decoder is a natural choice for language modeling since it's built to mask future tokens —a valuable feature when it's generating a sequence word by word. In contrast, BERT uses a different objective leading to an important distinction discussed bellow.

4 *Pre-train then fine-tune*

The conventional workflow for BERT consists of two stages: *pre-training* and *fine-tuning*. Pre-training uses two self-supervised tasks: masked language modeling (MLM) and next sentence prediction (NSP). In fine-tuning for downstream tasks, one or more fully-connected layers are typically added on top of the final encoder layer.

4.1 *Pre-training*

Besides using only half a transformer, both BERT and GPT present a significant advance over the original transformer by using *self-supervision* in *pre-training*, whereas Vaswani et al. [236] starts with a random initialization of the transformer weights and then proceeds directly with supervised training on labelled data.

The idea of pre-training from unlabelled data then fine-tune for a supervised downstream task has a long history [49, 93]. Lin et al. [129] traces back the intellectual origins of pre-training even further to the computer vision domain, dating back to the last decade in Erhan et al. [62]. The advantage of these approaches is that few parameters need to be learned from scratch. Moreover, pre-training based on self-supervision is, at least partly, behind the improvements achieved by GPT and BERT on language processing tasks. The advantages of self supervision are as follows [129]:

— Data provides the *supervision* for model pre-training without needing any external labelling. In GPT for example, the next token in the input sequence provides the supervision— "label". The availability of annotated training data is not longer a limitation, thus the amount of data that can be used for training the model can be largely expanded. Instead, computing resources and the amount of data available become the bottleneck [109].

— Pre-training based on self-supervised objectives provides a good base capable of reasonably handling language, that can be further fine-tuned for supervised downstream tasks. This is the "pre-train then fine-

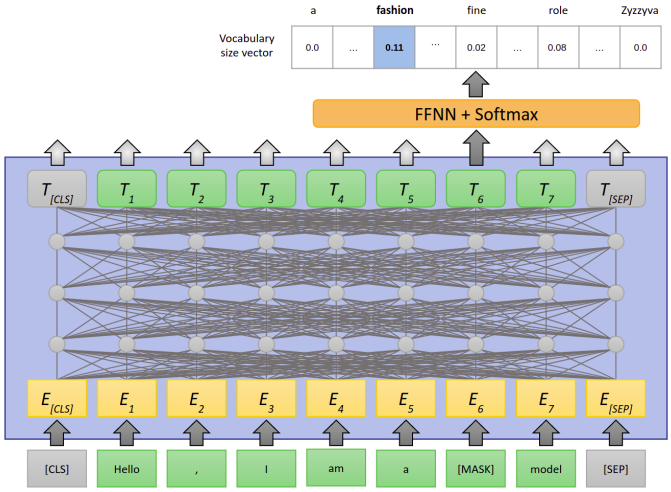


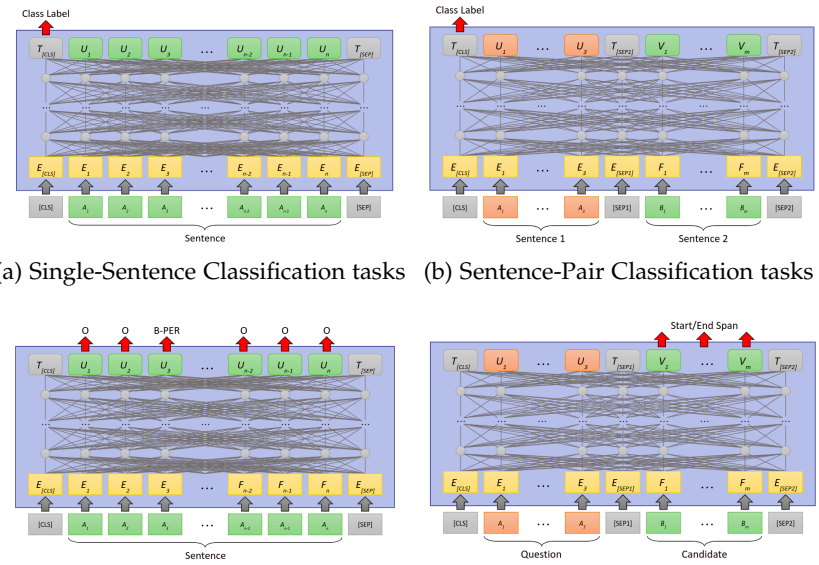
Figure 3.3 – BERT's Masked Language Modeling (MLM). Masks a percentage of the input sequence tokens at random, and trains the model to predict the masked tokens. In this example, the word "fashion" is masked from the input sequence by replacing it with the special token [MASK]. The last hidden vectors are fed through a feed forward network (FFNN) and a softmax over the vocabulary. The output vector contains the probability that the masked token corresponds to the i-th token in the vocabulary.

tune" recipe that became universal today for working with pre-trained language models such as BERT and GPT. The availability of open-source checkpoints and the lightweight fine-tuning process, that requires much less resources than training from scratch, contributed to the wide spread of BERT.

4.1.1 *Masked Language Modeling (MLM)*

In contrast to left-to-right language modeling that has been used for pre-training previous models [49, 93, 196], BERT introduces the "Masked Language Model" (MLM) self-supervised objective that can use both left and right contexts of a token. During pre-training, a percentage of the input tokens are randomly masked (i.e., replaced with the special [MASK] token), and the model is then optimized to predict those masked tokens using cross entropy. This procedure is also known as the *Cloze* task in the literature [229]. As illustrated in Figure 3.3, the final hidden representations corresponding to the masked tokens are fed into an output softmax over the vocabulary, as in a standard LM.

However, masking creates a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, the authors of BERT propose to not replace masked tokens with the [MASK] token. The training data generator chooses 15% of the token positions



(a) Single-Sentence Classification tasks (b) Sentence-Pair Classification tasks

(c) Single-Sentence Token Labeling (d) Sentence-Pair Token Labeling

Figure 3.4 – Illustrations of fine-tuning BERT on different NLP tasks. The model inputs are not limited to sentences, an input is a textual segment that can be a question, a paragraph, etc. Diagrams by Jimmy Lin (https://twitter.com/lintool/status/1285599163024125959)

randomly for prediction, if the *i*-th token is chosen, it is replaced with: (1) the [MASK] token 80% of the time , (2) a random token 10% of the time, and (3) remains unchanged 10% of the time.

4.1.2 *Next Sentence Prediction (NSP)*

Considering that many downstream tasks, such as Question Answering (QA) and Natural Language Inference (NLI), are based on understanding the *relationship* between two sentences, BERT is also pre-trained on the Next Sentence Prediction task. Pairs of sentences *A* and *B* are extracted from a monolingual corpus, where *B* is, in 50% of the time, the actual sentence following *A*, and 50% of the time it is a random sentence from the corpus. Devlin et al. [58] argue that the NSP task is beneficial to both QA and NLI. However, following work by Liu et al. [138] observed no drop in performance in models that lacked such pre-training, thus questioning the necessity of NSP.

4.2 *Fine-tuning*

Devlin et al. [58] proposed four task-specific models augmenting BERT with a single additional output layer, that is learned from scratch during fine-

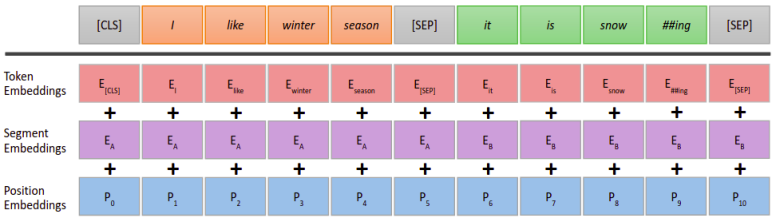


Figure 3.5 – BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

tuning. As illustrated in Figure 3.4, tasks (a) and (b) are sentence-level tasks where the whole input is assigned a label, while (c) and (d) are token-level tasks where labels are assigned to tokens. We use the term "sentence" to be consistent with the terminology used in the original BERT paper, however input sequences are not limited to "sentences", and can be a question, a paragraph, a passage from a document, etc.

— Single-sentence classification tasks, used for classification tasks over single-segments of texts such as sentiment analysis from movie reviews.

— Sentence-pair classification tasks, for example, identifying if two sentences are paraphrases, or semantically similar.

— Single-sentence token labeling, for example, named-entity tagging.

— Sentence-pair token labeling tasks, for example, question answering where the task consists in labeling the begin and end positions of the answer span in a candidate paragraph (the second sentence) given a question (the first sentence).

A special classification token [CLS] is added to the start of the input, and its final hidden state $T_{[CLS]}$ is typically used as input for the additional output layer in classification tasks; $T_{[CLS]}$ is considered as an aggregate representation of the entire input (single or pair of token-sequences). An other special delimiter token [SEP] is appended to the end of the input. For tasks involving more than one sequence input, [SEP] is additionally used to separate non-consecutive token sequences.

5 *Input representation*

Now that we have defined the different tasks on which BERT can be applied, we can define its input representation being the element-wise summation of the following three components (see example in Figure 3.5):

— **Token embeddings.** Learned token representations for each token in the sequence, fetched from a lookup table. BERT uses a WordPiece

Configuration	Layers	Hidden size	Attention heads	Parameters
Tiny	2	128	2	4 M
Mini	4	256	4	11M
Small	4	512	4	29M
Medium	8	512	8	42M
Base	12	768	12	110M
Large	24	1024	16	340M

Table 3.1 – BERT configurations: The commonly used Base and Large configurations were introduced in the original BERT paper [58], while the remaining configurations were proposed later by Turc et al. [235] for exploring effectiveness/efficiency tradeoffs.

tokenizer [248] capable of modeling large corpora, comprising millions of unique words (space separated), with a relatively small vocabulary size of only 30,000 tokens [1]. Such reduced vocabulary is attainable by splitting words into "subwords", e.g., the word "snowing" in Figure 3.5 is split into "snow" and "##ing", where "##" indicate that the current token is connected to the previous subword. It is however important to note that the splitting process is usually unsupervised and the resulting subwords are not necessarily linguistically meaningful. As opposed to "playing", words such as "thinking" or "working" are not split, while "exhilaration" is split into ("ex", "##hila", "##ration"), which do not correspond to morphemes.

— **Segment embeddings.** A learned embedding indicating whether a token belongs to sentence *A* or sentence *B* in tasks involving two inputs. This used in addition to the [SEP] token placed between the two input sequences.

— **Position embeddings.** A learned embedding determining the position of each token in a sequence, giving BERT a sense of linear order and relative positions between tokens.

6 *BERT configurations*

In the original paper, Devlin et al. [58] proposed two standard BERT configurations, namely: $\text{BERT}_{\text{Large}}$ and $\text{BERT}_{\text{Base}}$, with 24 and 12 transformer-encoder layers, respectively. In a later work, Turc et al. [235] conducted extensive experimentation to gain understanding of how knowledge distillation (discussed later in Section 5.1) and the pre-train then fine-tune process

1. Byte-Pair-Encoding (BPE) [220] is also a subword-based tokenizer worth mentioning since it was used in GPT as well as RoBERTa [138] which proposes a more robust version of BERT.

work in isolation, and how they can interact. This work unveiled the power of pre-trained distillation, a general yet simple algorithm for building compact models in three steps:

1. Pre-taining a compact model with the MLM objective, capturing linguistic phenomena from a large corpus.
2. Knowledge-distillation from the soft labels produced by a large teacher on unlabeled transfer data.
3. (Optional) Fine-tuning the compact model on labelled data to mitigate eventual mismatches between the distribution of the transfer and labeled sets.

Table 3.1 shows all model configurations which resulted from this study in addition to the original Base and Large configurations, where a configuration is characterized by its number of layers, the hidden dimension size and the number of attention heads. In general, larger configurations tend to achieve better effectiveness on downstream tasks, consequently these configurations are useful for exploring effectiveness/efficiency trade-offs.

7 *BERTology*

The recent dominance of pre-trained contextualized representations has served as the impetus for exciting and diverse interpretability research attempting to unveil the knowledge encoded in BERT weights. Popular approaches include fill-in-the-gap probes of MLM, analysis of self-attention weights, and probing classifiers with different BERT representations as inputs. The neologism *BERTology* was specifically coined to describe this flurry of interpretability research. We only report a small fraction of the findings in the context of NLP hereafter, we refer the reader to Rogers et al. [209] for a more complete overview.

A number of researchers focused on the syntactic knowledge of BERT. Probing classifiers have been widely used to determine whether something can be predicted from BERT's internal representations. For example, Lin et al. [132] showed that BERT representations are hierarchical showing resemblance to syntactic tree structures. Tenney et al. [231] also showed that BERT embeddings encode information about parts of speech, syntactic chunks and roles. Others have investigated BERT's behavior using fill-in-the-gap probes of MLM. Since BERT was pre-trained with MLM objective, it is possible to feed the masked token [MASK] to the model and ask it to predict the masked term, as a way to probe what the model has learned. Ettinger [64] found that BERT does not understand negation and is insensitive to malformed input (e.g., shuffled word order or truncated sentences). Meaning that BERT's syntactic knowledge is either incomplete or it is not needed for solving its

tasks. Glavaš and Vulić [79] shows that the latter is more likely since additional fine-tuning with supervised parsing does not make much difference for downstream task performance. Aside from syntactic knowledge, studies showed that BERT has some semantic knowledge about entity types, relations, semantic roles, and proto-roles [231].

Other researchers have examined BERT's attention heads and characterized their behavior. Clark et al. [39] distinguish between attending to previous/next tokens, [CLS], [SEP], punctuation, and "attending broadly" over the sequence. Kovaleva et al. [117] found that a limited set of attention patterns are repeated across different heads, suggesting that the model is over-parameterized. Indeed, manually disabling attention in certain heads leads to effectiveness improvements in some NLP tasks [237].

While these studies offer some insight about the general knowledge encoded in BERT, we are more interested in how BERT behaves in ranking. We will come back on studies attempting to understand how BERT models relevance in the following chapter; see Section 3.7.

8 *Conclusion*

We presented in this chapter an overview of BERT, a sophisticated model that brought together many crucial innovations to achieve unprecedented performance on a broad range of NLP tasks. We briefly covered BERT's relevant intellectual ancestries, important concepts and architectural components employed in BERT, and other pre-trained language models. We also discuss key findings about BERT's inner workings in the general context of NLP.

The remainder of this part is dedicated to the many applications, adaptations, and attempts to go beyond BERT to build better IR models. Considering the timely evolution of BERT applications to IR, Lin et al. [129] organize the literature into three main bodies of work. Following this timely organization, we start the next part of this dissertation with the straightforward applications of BERT and variants as rerankers in multi-stage reranking architectures initiated by Nogueira and Cho [173]. Then, we move to the applications of BERT for query and document expansion for improving first-stage sparse retrieval. Finally, we present the applications of BERT for learning dense representations suitable for text ranking (i.e., dense retrieval).

4

BERT in multi-stage reranking

1 *Introduction*

Deep transformer models pre-trained with language modeling objectives (PLMs), exemplified by BERT [58], have proven to be highly effective in a variety of classification tasks in NLP. Inspired by this success, Nogueira and Cho [173] were the first to demonstrate the effectiveness of PLMs for ad hoc ranking. The authors adopted the simplest and most straightforward formulation of ranking, in which the ranking task is converted into a classification problem. The proposed model, known as monoBERT, was deployed as a binary relevance classifier that estimates the probability each document belongs to the "relevant" class. At inference time, these estimates are then used to rank the documents for every user query. This is a direct implementation of the *Probability Ranking Principle*, which states that documents should be ranked in decreasing order of the estimated probability of relevance with respect to the information need [204].

From the computational perspective, applying inference to every document in a corpus for every single user query is impractical, considering both the time and memory complexity of transformer models, but also the linear growth of query latency w.r.t the corpus size. Despite the active exploration of architectural alternatives (discussed in Chapter 6), most applications of BERT for document ranking today focus on reranking a small list of candidates per query. In a typical end-to-end system, these candidates are the result of an efficient keyword search, usually with bag-of-words queries against inverted indexes (see Section 2.2). This gives rise to the standard "retrieve-then-rerank"

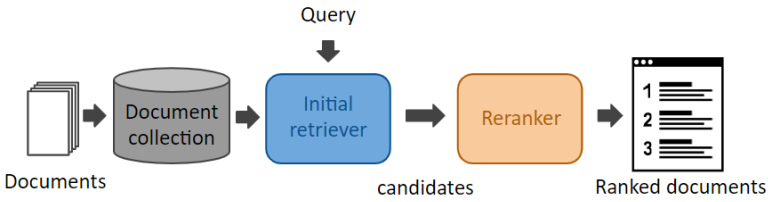


Figure 4.1 – The retrieve-then-rerank architecture, which is the simplest instance of multi-stage ranking architecture. The initial retriever (also called first-stage retriever) retrieves a list of candidate documents for each query, typically with bag-of-words queries against inverted indexes. These candidates are then reranked with a PLM such as monoBERT.

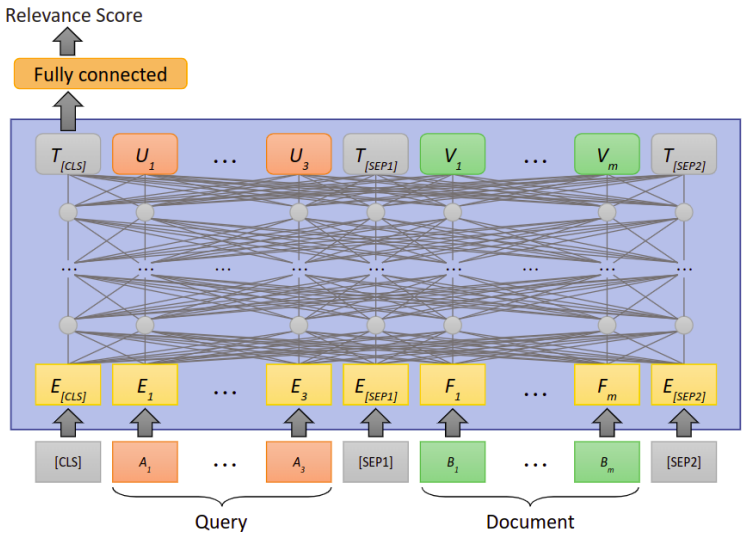


Figure 4.2 – The monoBERT architecture [173].

approach, illustrated in Figure 4.1. The initial retriever, also called first-stage retriever, returns a list of candidates for each query, typically scored using a traditional exact-match based method such as BM25. This retrieve-then-rerank approach is the simplest and most adopted instanciation of the more general multi-stage reranking architecture, which can include more than one reranker on top of the initial retriever (see Section 4.4).

We present in this chapter the first applications of BERT and its variants as part of multi-stage reranking architectures. We present in Section 4.2 a detailed description of the monoBERT model which is the starting point of many other models and provides a solid baseline for subsequent BERT-based IR models. The inability to handle long input sequences is inherent to BERT's design, posing a major limitation when it comes to full-length document ranking collections (e.g., scientific papers and web pages). We discuss the different approaches devised to overcome the length limitation of BERT for long document ranking in Section 4.3. In Section 4.4, we present how these approaches are deployed in multi-stage reranking architectures with more than one reranker. Finally, we present some attempts to go beyond BERT to explore more efficient ranking models (in Section 4.5), or different relevance modeling alternatives using generative PLMs (in Section 4.6).

2 *Relevance Classification with monoBERT*

Nogueira and Cho [173] propose the first application of BERT for ranking passages in the MS MARCO passage ranking task [9]. Their model,

monoBERT, follows the retrieve-then-rerank approach where BM25 is used as the initial retriever.

Ranking is formulated as a relevance classification task where monoBERT computes a score $R(d_i, q)$ estimating how relevant the candidate document d_i is to the query q. That is:

$$R(d_i, q) = P(Relevant = 1 | d_i, q) \tag{4.1}$$

2.1 *MonoBERT architecture*

The architecture of monoBERT is represented in Figure 4.2. Using the same notation as Devlin et al. [58], the query q is fed as *Segment A* and the candidate document d_i as *Segment B*. The special classification token $[CLS]$ is prepended to the input sequence, and the special delimiter token $[SEP]$ is placed at the beginning and end of the document segment to build the input sequence S as follows:

$$S = [[CLS], q, [SEP], d_i, [SEP]] \tag{4.2}$$

where q and d_i are represented with their tokens obtained after applying the WordPiece tokenizer.

The sequence S is then passed through BERT which produces contextualized vector representations for each token. The final contextual representation $T_{[CLS]}$ of the standard classification token $[CLS]$, that captures the interaction between the query and the document, is then used as input to a single fully-connected layer that estimates the probability that the document d_i is relevant to the query q, as follows:

$$R(d_i, q) = P(Relevant = 1 | d_i, q) \triangleq softmax(T_{[CLS]} \cdot W + b)[1] \tag{4.3}$$

where $T_{[CLS]} \in \mathcal{R}^D$, D being the hidden dimension size from BERT, $W \in \mathcal{R}^{D \times 2}$ is the weight matrix of the fully-connected layer and $b \in \mathcal{R}^2$ its bias term. $softmax(.)[1]$ denotes the softmax output corresponding to the "relevant" class (i.e., binary classification with two output probabilities for the "non-relevant" and "relevant" classes).

The monoBERT model including the BERT core, which is pre-trained, and the additional classifier layer, which is randomly initialized, is trained for relevance classification using cross-entropy loss:

$$L = -\sum_{j \in J_{pos}} \log R(d_j, q) - \sum_{j \in J_{neg}} \log \left(1 - R(d_j, q)\right) \tag{4.4}$$

where J_{pos} is the set of indexes of the relevant documents and J_{neg} is the set of indexes of non-relevant documents. Since the loss function takes into account only one candidate text at a time, this can be characterized as belonging to the family of pointwise learning-to-rank methods; see Section 2.3.

2.2 *Understanding BERT behavior in ranking*

Despite their success, little is understood about *why* pre-trained language models such as BERT are so effective for ranking. What new aspects of the task do they solve that previous approaches do not? A large of body of work was dedicated to help shed the light on the mechanisms, strengths and weaknesses of BERT-based ranking models.

Padigela et al. [183] study a set of hypotheses and found that BERT retrieves passages with more novel terms as opposed to BM25 which is biased towards high query term frequencies. The authors also found that BERT fails at capturing the context of long queries. Nonetheless, Dai and Callan [51] demonstrate that unlike traditional models –which prefer short keyword queries–, BERT can leverage stop words and punctuation thanks to its capacity to model language structure. Qiao et al. [194] argue that BERT should be understood as an "interaction-based sequence-to-sequence matching model" that prefers semantic matches between paraphrase tokens. While Zhan et al. [266] argue that the lower layers of BERT focus mainly on building semantic representations meanwhile upper layers capture interaction signals to predict relevance (i.e., typical design of representation-focused models).

Because analytic methods are impractical given the models' large number of parameters, Rennings et al. [201] propose diagnostic datasets which reformulate traditional ranking axioms (e.g., that documents with a higher term frequency should receive a higher ranking score [66]) as empirical tests. Rennings et al. [201] studied pre-BERT neural ranking architectures that predate the rise of BERT, and focused on just four axioms. Câmara and Hauff [28] extend this work to a distilled BERT model [216] and adds five more previously-proposed ranking axioms [67]. However, the authors find that these axioms are inadequate to explain the ranking effectiveness of their model. In the same line of research, MacAvaney et al. [146] introduce novel "textual manipulation tests" and "dataset transfer tests" in addition to the previous diagnostic tests. The authors gather these tests under a new framework for analysing the behavior of neural IR models including pre-BERT models, monoBERT as well as subsequent PLM-based models such as monoT5 (described in Section 4.6). Their study show that monoBERT is better than BM25 at estimating relevance when term frequency is constant, supporting the finding from Câmara and Hauff [28] that monoBERT does not satisfy the term frequency axioms. Using the newly introduced text manipulation tests, MacAvaney et al. [146] find that shuffling the order of words within a sentence or across sentences (altering the syntactic structure) has a large negative impact on PLMs, while shuffling the order of sentences within a document has a modest negative impact. The most surprising discovery in this study, is that appending non-relevant sentences to the end of a document tricks monoBERT into increasing the relevance score of the document, while

adding relevant terms – generated by docT5query (described in Chapter 5) – decreases its score. Using the dataset transfer tests, where two versions of the same documents are compared w.r.t a query, the authors find that monoBERT scores informal text slightly higher than formal text, and fluent text slightly higher than documents written by non-native speakers.

Despite all the efforts put into understanding how BERT works for ranking, a lot is yet to uncover and explanations are far from complete. Previous work indicates that BERT show evidence of combining elements from both representation-based and interaction-based models. Empirical analysis from text manipulations further show that BERT leverages soft semantic match, as well as term position signals. However, PLMs can also exhibit unexpected behaviors when additional content is added to documents, or when documents are expressed with different levels of fluency or formality. At the end, the inner workings of pre-trained language models remain unclear and behavior across different queries, corpora, and architectures is variable.

2.3 *Training BERT for ranking*

"Pre-train then fine-tune" is the de facto workflow for training BERT. It is first pre-trained on "general domain" corpora such as Wikipedia using self-supervision to gain general and transferable knowledge about the language. The obtained checkpoint can then be fine-tuned on task-specific labeled data drawn from the same distribution as the target task. Variations of this general recipe can be explored to better adapt BERT to the ranking task at the pre-training or fine-tuning levels.

2.3.1 *Pre-training*

While there may be some overlap between the pre-training and target corpora, they may nevertheless differ in some properties like vocabulary distribution, genre, and numerous other factors [129]. Aside from the corpora, the core of IR is to model the notion of relevance [218], which is not considered in BERT pre-training objectives (MLM and NSP). To address these issues, researchers in the IR community have started exploring additional pre-training on the target corpus and rethinking new pre-training objectives tailored for the ranking task.

Nogueira et al. [176] investigate the benefit from additional pre-training on the ranking corpus, named target cropus pre-training (TCP), in the context of their multi-stage reranking architecture (described in Section 4.4). The authors take the original pre-trained BERT checkpoint (on general domain data), and further pre-train it on the MS MARCO passage corpus using the same self-supervision objectives, i.e., Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) (see Section 3.4). They found that TCP

brings modest improvements over the original general-domain pre-training, nonetheless the gain does not require any labeled data, and thus TCP might be worthwhile in certain scenarios.

Inspired by the query likelihood model (QL) [191], Ma et al. [142] propose a novel pre-training task named Representative Words Prediction (ROP) specifically tailored for IR. QL assumes that the query is a piece of representative text generated from the "ideal" document. Hence, ROP samples pairs of term sets according to the multinomial unigram language model [262], and then pre-trains BERT to predict the pairwise preference. The resulting model named PROP has the same architecture as monoBERT, where the query segment is replaced by a term set representative of the document segment. Ma et al. [142] show that ROP improves effectiveness over pre-training with MLM when reranking BM25. In a later work, Ma et al. [143] propose leveraging BERT to replace the classical unigram language model for the ROP task in the B-PROP model. Inspired by the divergence-from-randomness idea [3], they propose a contrastive method to leverage BERT's [CLS]-token attention to sample representative words. Experiments show that B-PROP performs better than PROP on the downstream document ranking task. Furthering this idea, Ma et al. [145] propose HARP that introduces hyperlinks and anchor texts to replace the sampling method which outperforms PROP.

Recently, Chen et al. [32] incorporate IR axioms into model pre-training using a new method named ARES. They generated training samples with a number of existing IR axioms to guide the training of neural ranking models. Compared to existing pre-training approaches, ARES is more intuitive and explainable. Experimental results show the effectiveness of ARES especially in low-resource settings, e.g., zero-shot and few-shot learning which we discuss in the following section.

2.3.2 *Fine-tuning*

Fine-tuning a pre-trained language model requires access to labelled data drawn from the same distribution as the target task. However, annotated data is usually limited (costly) or even unavailable for the domain and/or task at hand. To overcome this limitation, researchers explored the idea of leveraging out-of-domain or out-of-task labelled data to benefit the related target task where labelled data is limited. The model is first fine-tuned on *out-of-distribution* labelled data before fine-tuning on *in-distribution* labelled data. This strategy is known as "multi-phase" fine-tuning, where fine-tuning spans multiple "phases". Analogous to how pre-training on general-domain data provide a model with *general language modeling knowledge*, multi-phase fine-tuning attempts to provide the model with *general knowledge about the task* from available (free of annotation cost) out-of-distribution labelled data.

In order to rank news articles, Akkalyoncu Yilmaz et al. [2] exploit annotated data from an out-of-domain Microblog collection and calls this cross-domain relevance transfer. Dai and Callan [51] leverage log-data from a search engine to fine-tune BERT, providing it with general search knowledge before fine-tuning on TREC collections. Zhang et al. [269] coined "pre-fine-tunning" when investigating the benefits of fine-tuning on the large MS MARCO passage ranking collection before fine-tuning on limited collection-specific labelled data. Pre-fine-tuning is now widely adopted and applied to monoBERT as well as subsequent models [122, 22].

Despite the gains brought by pre-fine-tuning, sequentially learning from multiple labelled datasests can cause the model to lose patterns acquired from the previous dataset. This phenomenon, known as "catastrophic forgetting", has been studied by Zhang et al. [267] which revealed that BERT-based ranking models seem to better retain effectiveness on the pre-fine-tuning dataset despite further fine-tuning, compared to pre-BERT neural ranking models.

Pushing the limits of training BERT on out-of-distribution data, researchers investigate "zero-shot" transfer settings (the extreme instantiation of few-shot learning) where a model is exclusively fine-tuned on out-of-distribution annotated data and directly applied to the target task. Examples include Birch [2] (in Section 4.3) and monoT5 [174] (in Section 4.6).

Instead of pre-fine-tuning on out-of-distribution labelled data, Zhang et al. [267] propose a reinforcement weak supervision method with monoBERT, called ReInfoSelect. ReInfoSelect trains a selector model to select some constructed anchor document pairs for training the monoBERT via reinforcement learning. It takes the ranking performance (i.e., nDCG) as the reward. Experiments show that the neural ranker trained by ReInfoSelect can match the effectiveness of neural rankers trained on private commercial search logs. Alternatively, MacAvaney et al. [148] explore whether monoBERT can benefit from a training curriculum which provides a systematic approach for ordering training instances from simple to hard [17]. The authors assign a weight for each training example through a difficulty heuristic based on BM25 scores. Experimental results show that this weighted curriculum learning approach can significantly improve the effectiveness of monoBERT. Compared to multi-phase fine-tuning which generally spans multiple datasets, the fine-tuning process in a curriculum follows a multi-step path which can be applied to a single dataset.

In conclusion, researchers have investigated different extensions to the standard "pre-train then fine-tune" workflow to train BERT for the ranking task. These extensions leverage data from related domains and tasks in both self-supervised (i.e., pre-training) and supervised (i.e., fine-tuning) settings. While these extensions have been proven to improve BERT's ranking

effectiveness, there are no clear guidelines to properly apply them (e.g., number of epochs, order of out-of-domain fine-tuning phases, etc.). Further exploration is needed to fully understand how exactly these pre-training and fine-tuning extensions work in order to achieve measurable and predictable gains.

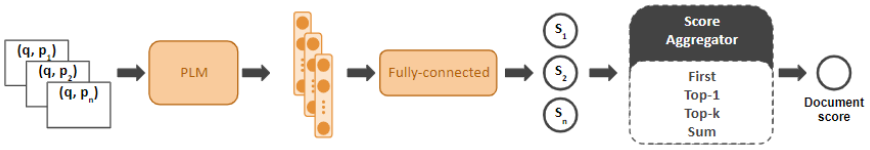
3 *Full-length document ranking with BERT*

Considering the quadratic time and memory complexity of the self-attention mechanism in Transformers [236] (see Section 3.3), PLMs such as BERT were pre-trained with a limited input length (i.e., 512 tokens). Past this limit, position embeddings are not available. Henderson [86] point out the importance of position embeddings and argue that BERT can be seen as a "bag of vectors", where cues about the linear structure of the language are *exclusively* provided by these learned position embeddings. Thus, past the pre-trained limit of 512 tokens, BERT has no sense of token order (position) and so the input text will essentially be treated as a bag of tokens.

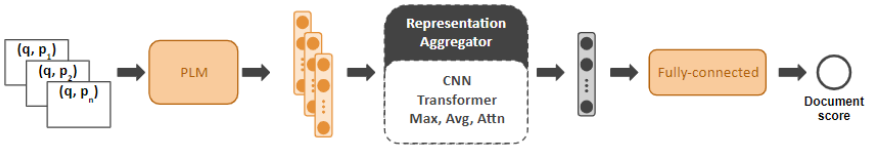
This length limitation restricts the application of BERT-based models such as monoBERT to short paragraph-length documents like MS MARCO passages used in [173]. However, in traditional ad hoc retrieval, documents can contain thousands of tokens (especially with WordPiece tokenization) in standard TREC collections [240, 45, 44]. Besides, relevance judgements are provided at the document level. In other words, if a document is labelled "relevant" as a whole, it is unclear which part of it contains the "relevant material". Assuming the *Scope Hypothesis* [206]: a long document consists of a number of unrelated short documents concatenated together. In this way, the relevance matching could happen in any part of a relevant document. Building on this assumption, a majority of applications are to segment the long document text into smaller chunks that can be processed by BERT, individually, and then do an aggregation over chunks to produce document-level relevance scores. Based on the aggregation type, these methods can be broadly categorized into two classes: (1) Passage Score Aggregation: aggregate the relevance score of the query and segmented passage; and (2) Passage Representation Aggregation: aggregate the representations of segmented passages to document representations first and then compute the relevance between query and the aggregated document representation.

3.1 *Passage Score Aggregation*

Passage score aggregation is a post-processing method that aggregates the relevance scores between the query (q) and the segmented passages ($\{p_1, p_2, ..., p_n\}$ of a document d) provided by the PLM as shown in Figure



(a) Passage Score Aggregation



(b) Passage Representation Aggregation

Figure 4.3 – Score aggregation vs. representation aggregation approaches [123]

4.3a. Previous work explored various segmentation and score aggregation methods. Akkalyoncu Yilmaz et al. [2] propose a segmentation method as well as a training approach in Birch to overcome the length limitation of monoBERT. On the one hand, Birch leverages out-of-domain tweet data (short passages with relevance judgements) to fine-tune monoBERT for the ranking task, then inference is directly performed on TREC newswire collections without further in-domain fine-tuning. Birch is one of the first instanciations of zero-shot cross-domain learning (see Section 2.3.2), since the model was not trained on labelled data drawn from the target dataset. Experimental results reveal that monoBERT has strong domain transfer capabilities for relevance matching. On the other hand, Birch splits documents (i.e., news articles) into their sentences during inference to feed them through monoBERT. Inference is applied to each individual sentence in the document w.r.t the query, and then the top-k scores are aggregated with the original document score s_d obtained by the first-stage retriever (i.e., BM25) to obtain the document score s_f, as follows:

$$s_f \triangleq \alpha \cdot s_d + (1 - \alpha) \cdot \sum_{i=1}^{k} w_i \cdot s_i \tag{4.5}$$

where s_i is the score of the i-th top scoring sentence according to monoBERT. The parameters α and $w_{1:k}$ are tuned via cross-validation.

Results show that the top-scoring sentence is a good proxy for the relevance of the entire document.

Dai and Callan [51] propose a different approach where documents are split into overlapping passages using a fixed-size sliding window. Fine-tuning is performed on the segmented passages drawn from the target-task (i.e., in-domain training data as opposed to Birch out-of-domain tweet data) in a cross-validation setting. The relevance judgments of segmented passages

are consistent with the document, that is, if the document is relevant to a query, all the segmented passages are also relevant to the query and vice versa. However, according to the Scope Hypothesis, the document could be partially relevant to a query, and thus not all passages are relevant to a query. In other words, the training instances are expected to contain noise. During inference, the relevance score of each passage is predicted independently with monoBERT, and the document score is obtained by aggregating the passage scores. The authors explore three aggregation methods:

1. BERT-firstP only uses the score of the first passage, i.e., $s_f = s_1$;
2. BERT-maxP uses the maximum score of the passages, i.e., $s_f = \max_{i=1}^{n} s_i$;
3. BERT-sumP sums all the relevance scores of passages, i.e., $s_f = \sum_{i=1}^{n} s_i$.

Experimental results indicate that BERT-maxP and BERT-sumP perform better than BERT-firstP on traditional TREC ad hoc retrieval tasks since all passages are taken into account. But these two methods require more computational cost as all the query-passage pairs need to be trained and predicted while BERT-firstP only considers the first passage of each document. BERT-maxP remains the best performing aggregation method (i.e., it respects the Scope Hypothesis by considering all passages, and taking the best scoring as proxy for the whole document score) and is widely adopted and replicated in subsequent work [122, 269, 22].

Besides the score aggregation exploration, Dai and Callan [51] propose the first exploration of using different query representations with monoBERT. Previous IR models, including Birch, only use one representation of the query, namely "title" in TREC topics, which typically comprises a set of keywords, akin to user queries posed to web search engines. However, TREC topics represent the information need with different external representations including a "title", a "description" and a "narrative" (see Section 1.3). Dai and Callan [51] find that ranking documents using "description" queries is more effective than using the standard "title" queries. While BoW IR models perform better on short keyword queries, BERT is able to leverage the linguistically richer description queries to improve ranking effectiveness. Further experiments using only keywords (i.e., remove stop-words and punctuation) present in descriptions and narratives show a degradation in the ranking performance of BERT, as opposed to the BoW initial-retriever which is able to regain its lost effectiveness. These results demonstrate the important role that non-content words play in BERT contextualization.

Most applications of BERT for ranking use the standard contextualized representation of the [CLS] token to predict the relevance scores and discard the contextualized representations of remaining tokens. Alternatively, MacAvaney et al. [149] explore the application of contextualized representations from BERT in neural ranking models as a replacement for static embeddings. The proposed CEDR model leverages the contextualized em-

beddings of BERT to build a similarity matrix (chunk by chunk), which is then fed into an existing interaction-focused neural ranking model such as DRMM [83] and KNRM [250]. The [CLS] representation is also incorporated in CEDR to enhance the model's signals. By combining BERT and pre-BERT models, CEDR is significantly better than vanilla BERT.

3.2 *Passage Representation Aggregation*

Instead of only aggregating passage scores, another line of research proposes to aggregate the representations of all passages in a long document so that the relevance score is estimated by considering all the passages together. PAssage Representation Aggregation for Document rEranking (PARADE) [122] segments long documents into passages and performs representation aggregation on the [CLS] representation of each query-passage pair, as illustrated in Figure 4.3b. The authors propose two types of passage aggregation methods: (1) Using a mathematical operation such as the element-wise mean, max and sum on the representation vectors; Or (2) using a deep neural network including a multi-layer perceptron (MLP), convolutional neural networks (CNN) and Transformer layers. By aggregating the passage representations with more complicated architectures, PARADE $_{\text{Transformer}}$ can significantly improve the performance over passage score aggregation methods like BERT-maxP and other passage representation aggregation methods like PARADE $_{\text{max}}$.

Similarly, Wu et al. [249] propose the Passage-level Cumulative Gain model (PCGM), which represents query-passage pairs with BERT (i.e., the contextualized representation of [CLS]), and then uses an LSTM to aggregate the sequence of query-passage [CLS] vectors. PCGM is trained end-to-end to predict the cumulative gain after each passage, that is, the amount of relevant information a reader would encounter after having read a document up to this passage. At inference time, the document-level gain (i.e., relevance score) is given by the cumulative gain of its last passage (i.e., after reading the entire document). This approach suggests the availability of passage-level gain labels, in contrast to PARADE where the document label is used to label all its passages as suggested by Dai and Callan [51].

In order to demonstrate the superiority of capturing fine-grained passage-level relevance signals, the authors annotated a corpus of Chinese news articles with passage-level cumulative gain. These human annotations reveal that relevant documents are typically longer than other documents, and the higher the document-level gain, the more passages need to be read to reach this gain. In other words, the relevance of a document can be accurately estimated by its most relevant passage, which is consistent with previous work such as BERT-maxP. Nonetheless, distinguishing between different relevance grades (e.g., relevant vs. highly-relevant) might require the model

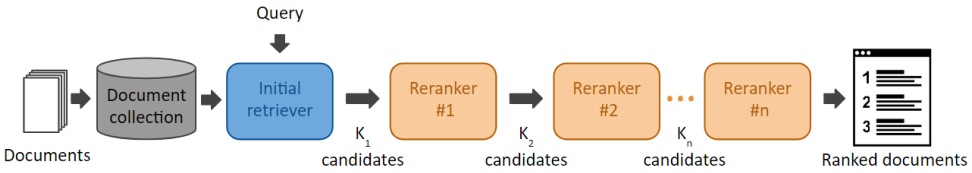


Figure 4.4 – The multi-stage reranking architecture with a first-stage (initial) retriever followed by one or more reranking stages ($K_1 \geq K_2 \geq ... \geq K_n$). According to the number of rerankers (n): the retrieval process can be defined as a Single-stage retrieval ($n = 0$), two-stage reranking or retrieve-then-rerank ($n = 1$), or Multi-stage reranking ($n \geq 2$).

to accumulate evidence from multiple passages, which suggests that BERT-maxP might not be sufficient.

3.3 *Alternative Transformer architectures for long sequences*

In order to avoid processing long documents in chunks, researchers explore alternative transformer architectures specifically tailored for long input sequences. The Reformer [114] restricts the standard all-to-all attention (see Section 3.3) to the most similar tokens only, based on locality-sensitive hashing. Beltagy et al. [14] design Longformer, which reduces the computational cost of attention using sparse patterns through a sliding window to capture local context, and global attention tokens that can be specified for a given task. Jiang et al. [105] propose QDS-Transformer, an application of Longformer to document ranking where the query tokens are global attention tokens attending to all query and document tokens.

While these new transformer architectures reduce the computational complexity, it is not clear whether these alternatives can match the ranking effectiveness of aggregation-based approaches.

4 *Multi-stage rerankers*

Most applications of BERT to document ranking are deployed as rerankers in a retrieve-then-rerank architecture. This is the simplest instanciation of the general multi-stage reranking architecture depicted in Figure 4.4. In this setup, the initial retrieval is followed by a number of n reranking stages, where the output of each reranker feeds the input of the following stage. To be more specific, each stage $S_i, i \in \{1..n\}$ receives a ranked list consisting of K_i candidates from the previous stage and provides a ranked list comprising K_{i+1} candidates to the following stage, with $K_{i+1} \leq K_i$. The final ranked list produced by the last stage S_n is the output of the multi-stage reranking architecture.

Multi-stage reranking evolved in the mid-2000s as a means for deploying systems exploiting expensive learning-to-rank (see Section2.3) models while managing trade-offs between effectiveness and efficiency [5, 30, 41, 33, 136, 94]. Because more effective ranking methods leverage computationally expensive designs, inference can be prohibitively slow since query latency increases linearly with the number of candidates considered. In multi-stage architectures, however, early stages can exploit "cheaper" methods to discard candidates that are easy to detect as non-relevant, so that later stages can operate "expensive" models on fewer and fewer candidates. This setup can thus provide systems with good trade-offs between the ranking quality (i.e., use of effective models) and retrieval latency (i.e., expensive models are only used on few candidates).

In the context of PLMs, Nogueira et al. [177] propose the first multi-stage reranking architecture as a solution for mitigating the quadratic complexity of BERT-based ranking models. The authors propose duoBERT, a "pairwise" extension of monoBERT where the input sequence is comprised of a query and two documents . DuoBERT is trained to estimate the relevance of a document relative to another by leveraging signals from a document pair. However, due to the length limitation of BERT, the whole sequence is truncated to 512 tokens and each document can have at most 223 tokens. In order to manage the effectiveness/efficiency trade-offs, the authors propose a multi-stage ranking setup where monoBERT reranks the initial K_1 candidate documents retrieved by BM25, and then provides K_2 candidates to duoBERT. The ranked list returned by duoBERT is the output of the multi-stage architecture. By introducing two rerankers in a pipeline, the effectiveness/efficiency tradeoff space is greatly expanded, which could lead to a setting with both better ranking quality and faster inference than what can be achieved in a single-stage reranker. However, K_1 and K_2 need to be properly set, empirically, to achieve the desired balance, as the pipeline is not end-to-end differentiable.

An obvious extension of the pairwise approach in duoBERT is the "listwise" approach, in which the relevance of a document is determined jointly with a list of other candidates. Given the large number of possible permutations that can be considered, a multi-stage ranking architecture is a natural choice. Zhang et al. [268] devise a "listwise" ranking model, in which all candidates are fed to BERT simultaneously, such that self-attention can enable joint inference to rank the document list. Due to the length limitation of BERT, documents are truncated to fit the 512 token sequence. Though results are encouraging, feeding multiple documents into BERT remains a major challenge, and the superiority of listwise approaches on their pairwise counterparts is not clearly established.

5 *Towards more efficient transformer-based ranking*

BERT, and transformer-based PLMs in general, brought a huge performance boost to the state-of-the-art ranking models, however these performance gains came with high computational costs. Hofstätter and Hanbury [89] show that real-world applications of BERT are impractical or prohibitively expensive considering its slow query latency, as opposed to its neural predecessors (i.e., pre-BERT models). Combining this with Kovaleva et al. [117] observation –that revealed a repetition of a limited set of attention patterns across different heads– that BERT appears to be over-parameterized, the question of trading some of BERT effectiveness for smaller and more efficient models arose.

Two major lines of research attempt to answer this question: The first is knowledge distillation from larger into smaller BERT models, and the second is clean-slate redesigns of transformer models specifically for ranking.

5.1 *Knowledge Distillation*

Knowledge distillation is a general approach where a smaller model named the *student* is trained to mimic the behavior of a larger model called the *teacher* [87]. The practical motivation behind distillation is to compress the teacher's knowledge into a smaller and more efficient student while maintaining the same effectiveness on a target task.

Tang et al. [225] explore knowledge distillation from BERT into simpler RNN-based neural networks. The authors report a loss in effectiveness, which is expected considering the simpler architecture of the student models. Nonetheless, the inference is accelerated by an order of magnitude. On the other hand, Jiao et al. [106] and Sanh et al. [216] both investigate distilling the general trained BERT model into smaller versions of BERT (i.e., same transformer-based architecture) with only six transformer layers, namely: TinyBert and DistilBERT, respectively.

In the context of reranking, researchers explored how smaller and more efficient variants of BERT can be effectively trained using distillation from a larger trained BERT reranker [74, 122, 268, 35]. Notably, Gao et al. [74] study the distillation procedure and its impact for ranking using TinyBERT. The authors identify three distillation procedures:

1. **Ranker Distill.** A randomly initialized student model is trained to mimic the behavior of the BERT teacher that has already been fine-tuned for ranking;
2. **LM Distill + Fine-tuning.** Distill the pre-trained teacher (LM knowledge) into the student model then fine-tune, normally, the student for ranking;

3. **LM Distill + Ranker Distill.** Both LM and ranking-specific knowledge are taught to the student model via distillation from the teacher (i.e., LM distillation followed by ranker distillation).

While Ranker distillation alone yield significant loss in effectiveness, the remaining distillation procedures lead to student models comparable to the teacher in effectiveness while being more efficient.

Li et al. [122] investigate knowledge distillation applied to their PARADE model using smaller BERT variants distilled by Turc et al. [235]. The authors explore on the one hand training directly PARADE models based on smaller BERT variants, and on the other hand applying ranker distillation to train student PARADE models based on smaller BERT variants from a teacher PARADE model based on $\text{BERT}_{\text{Base}}$. Experiments show that the student models trained under the ranker distillation procedure are more effective than smaller models trained directly.

Knowledge distillation has emerged as a promising approach to train smaller variants of BERT with negligible losses in the ranking quality compared to a larger BERT model. Evidence from previous investigations suggest that distilling a teacher model that has already been fine-tuned on the ranking task into a smaller pre-trained student model appears to be the best *distillation procedure*. We will discuss more applications of knowledge distillation for ranking in the context of dense retrieval in Chapter 6.

5.2 *Rethinking transformers for ranking*

Knowledge distillation has proven to be effective for managing effectiveness/efficiency trade-offs with PLMs such as BERT. Nevertheless, using a smaller model with the same architectural design might not be the only solution for building efficient ranking models. A second line of research explores whether redesigning transformers, from scratch, specifically for ranking can achieve better effectiveness/efficiency trade-offs.

Hofstätter et al. [92] propose Transformer Kernel (TK) that uses separate transformer blocks to encode query and document terms which are then used to compute a similarity matrix fed to KNRM [250]. In a later work, Hofstätter et al. [91] propose TKL, a variant of TK with *local self-attention* for handling long documents. In other words, attention with distant tokens (outside the local attention range of 50 tokens) is not set to zero. Mitra et al. [166] extend the TK design further by assuming query term independence and replacing the transformer layers with novel and more efficient "conformer" layers. Unlike BERT, these models were trained from scratch (i.e., without pre-training on LM tasks) which might explain the effectiveness gap with BERT.

It is unclear, for the time being, if clean-slate redesigns of transformers such as TK/TKL/CK can benefit from the same self-supervised pre-training that is the hallmark of PLMs such as BERT. This same pre-training that is considered the main source of the big leaps in effectiveness witnessed on both NLP and IR tasks. Lin et al. [129] support that whether pre-training (i.e., the quality of contextualized representations) or the model architecture (i.e., the relevance matching machinery) is more important is still an open question and represents a research direction worth exploring.

6 *Generative Ranking Models*

According to the two schools of relevance modeling, that are, discriminative modeling or generative modeling, in the IR literature [191, 203], PLM applications for reranking can be categorized into two classes:

1. **Discriminative Ranking Models.** In which the ranking task is converted to a relevance classification task. This includes all PLM-based models that we discussed so far in this chapter;
2. **Generative Ranking Models.** Which approximate the true relevance distribution by modeling the generative process between queries and documents.

It is no surprise that researchers explored the usage of PLMs in the generative ranking framework [179, 174]. Such applications are either based on the (1) Query Generation process, which is inspired by the query likelihood model [191], or the (2) Relevance Generation process, which generates a specified relevance token given the query and the document.

6.1 *Query Generation*

The intuition behind query likelihood is to rank the documents based on the probability that the query is generated by a model of the document. Query generation is a direct implementation of query likelihood using generative PLMs such as GPT [196] or BART [120] which are fine-tuned to generate a query given a relevant document. At inference time, the relevance score of a document d_i is the probability estimated by the model for generating the query q:

$$R(d_i, q) = P(q|d_i) = \prod_{j=1}^{|q|} P(q_j|q_{<j}, d_i) \tag{4.6}$$

where q_j is the j=th query token and $q_{<j}$ are all query tokens occurring before q_j. Note that, at inference time, the model also uses the "Teacher Forcing" strategy like in the fine-tuning process. That is, for each step (i.e., token

generation), the oracle query token (i.e., ground truth) is used as input for generating the next, instead of model output from a prior time step.

$$L = \sum_{(q,d^-,d^+)\in C} \max(0, -\log P(q|d^+) + \log P(q|d^-)) \tag{4.7}$$

where d^+ and d^- are relevant and non-relevant documents for the query q, respectively, and C is the set of training triples.

Experimental results show that query generative models are as effective as simple discriminative ranking models for answer selection.

6.2 *Relevance Generation*

Relevance generation is focused on generating specified relevance tokens (e.g., "relevant" vs. "non-relevant") by feeding the concatenation of the document and the query into a generative PLM, and the probabilities of these relevance labels are treated as relevance scores. In essence, the relevance generation is a classification task as the model is trained using pointwise loss function on relevance tokens and ranks documents by the probability of predicting the target relevance token (e.g., "relevant").

Nogueira et al. [174] propose using the sequence-to-sequence (i.e., encoder-decoder) T5 [197] model for modeling relevance generation. In T5, Raffel et al. [197] introduce a novel idea where every task is *cast* as generating some output text given some input text. For example, the translation task from English to French can be modeled with the following text-to-text template:

translate English to French: [English input] (4.8)

where "translate English to French" is a literal string, which T5 was fine-tuned to associate with the translation task from English to French. In other words, the task to be performed is included as part of the input. Following this formulation, Nogueira et al. [174] devise a text-to-text template for the ranking task where the input is :

Query: [q] Document: [d_i] Relevant: (4.9)

and the model, named monoT5 (analogous to monoBERT), is fine-tuned to generate the relevance tokens "true" for relevant documents and "false" for non-relevant ones. The probability of the "true" token is used to represent the document relevance score, which is normalized with the softmax function over the logits of "true" and "false" tokens.

Experimental results indicate that monoT5 is overall more effective than monoBERT, but also more data efficient to train as well. In other words, monoT5 appears to excel at few-shot learning compared to monoBERT. Pushing the experiment further, Nogueira et al. [174] explore the zero-shot setting

with monoT5. Unsurprisingly, the effectiveness of monoT5 in increases with the model size [1], and the T5-3B model outperforms *supervised* discriminative models such as BERT-maxP in a *zero-shot* manner.

Compared to BERT, which has an encoder-only design, T5 is an encoder-decoder design leveraging the full transformer architecture proposed by Vaswani et al. [236]. Nogueira et al. [174] argue that the decoder part of the model makes important contributions to relevance modeling. The authors investigate the impact of the relevance tokens choice on monoT5 effectiveness. They investigated swapping the relevance tokens so they mean their opposite, using alternative tokens such as "yes/no" and even arbitrary token combinations, e.g., "apple/orange". When the model is fine-tuned on sufficient training data, the choice of the tokens does not matter. However, in a low-resource regime, the authors observe that the choice of the relevance tokens is quite important, as it becomes more difficult for the model to learn how to associate arbitrary tokens with the relevance labels. This suggests that the model is leveraging the decoder part of the transformer to build the relevance matching machinery. Nonetheless, how exactly so remains an open question.

Mirroring monoBERT, a pairwise approach using T5, similar to duoBERT (see Section4.4), is a possible extension of monoT5. This approach, named duoT5 is proposed by Pradeep et al. [193] where the input template is extended to include a pair of documents:

$$\text{Query: } [q] \text{ Documento: } [d_i] \text{ Document1: } [d_j] \text{ Relevant:} \tag{4.10}$$

The model is fine-tuned to predict "true" if d_i is more relevant than d_j w.r.t the query q, and "false" otherwise.

Similarly to duoBERT, Pradeep et al. [193] deploy monoT5/duoT5 in a multi-stage reranking architecture achieving top results in the TREC Deep Learning passage ranking task [44].

The advent of PLMs spawned a resurgence of generative approaches for document ranking. Nogueira et al. [174] and Nogueira dos Santos et al. [179] works demonstrate the effectiveness of sequence-to-sequence transformer models for ranking, and provide a fresh perspective on well-studied language modeling approaches to IR that date back to the late 1990s.

Finally, it is interesting to note that a hybrid discriminative-generative approach to ranking can be considered. Liu et al. [133] propose a multi-task learning approach to jointly learn the discriminative and the generative relevance modeling in a unified pre-trained model. The underlying assumption is

1. T5 has three configurations: T5-base with 220M parameters, T5-large with 770M parameters and T5-3B with 3B parameters.

that joint discriminative and generative retrieval modeling leads to more generalized, and hence more effective retrieval models. The authors cross-train a retrieval task and one or more complementary language generation tasks. For the generative PLM (i.e, BART), they feed the document and the query into the encoder and the decoder respectively. Then, the query is generated in a sequence-to-sequence manner and the relevance score is calculated by the last token of the entire sequence using a feed-forward layer. Since the bidirectional attentions in the BERT discriminative PLM cannot fully adapt to the sequence-to-sequence training strategy, they implement a mix of attention mechanisms including bidirectional attention, unidirectional attention and cross attention to support sequence-to-sequence tasks. Their experiments show that jointly learning discriminative tasks and generative task leads to significant improvement on the MS MARCO passage ranking task.

7 *Conclusion*

We reviewed in this chapter the applications of PLMs in the context of reranking approaches. We first covered the basic relevance classification approach of the monoBERT model and the subsequent extensions proposed to address the model's input length limitation. These extensions include aggregation-based models such as Birch, BERT-maxP, CEDR and PARADE, but also attempts to redesign the transformer architecture for long input sequences. Aside from the modeling aspect, we also present extensions to the conventional pre-train the fine-tune training workflow to improve effectiveness on the ranking task. In addition to BERT applications for reranking, we discuss the body of work investigating the behavior of these applications.

Considering the high computational cost of transformer-based rerankers, various approaches were investigated to build more efficient models. While one line of research explores how to train smaller BERT models using knowledge distillation, another rethinks the transformer architecture for the ranking task (i.e., TK, TKL, and CK).

Nogueira et al. [174] and Nogueira dos Santos et al. [179] take a different direction and explore generative approaches for ranking and highlight the potential of sequence-to-sequence PLMs for document ranking.

With all these diverse research directions, we have covered the various applications of BERT in the reranking component. In the next chapters, we review the applications of BERT for first-stage retrieval. We start with applications for improving sparse retrieval by enriching and expanding the representations of queries and documents in the following chapter. We move afterward to direct applications of BERT for dense retrieval in Chapter 6.

5

BERT for sparse retrieval

1 *Introduction*

State-of-the-art search systems, use multi-stage reranking pipelines in which an efficient first-stage retrieves an initial set of candidates from the document corpus, and one or more reranking models improve and prune the ranking. Typically the first-stage retriever is a bag-of-words (BoW) retrieval method that fetches information from an inverted index, and estimate relevance based on exact matches between queries and documents. Therefore, initial retrieval is impacted by the *vocabulary mismatch* problem [70], introduced in Section 2.2 as a core problem in information retrieval (IR). As a consequence, relevant documents with no overlap with the query, or even missing a key term from the query (e.g., use of synonyms), will not be retrieved during the initial retrieval. Hence, they will never be presented to the downstream reranking stages.

On the other hand, ranking models based on neural networks, such as PLMs, are able to accomplish soft or semantic matches by learning distributed representations, and can thus alleviate the vocabulary mismatch problem. However, as we have seen so far, these models are deployed in the reranking stages, and therefore suffer from a strict upper bound imposed by any recall errors in the first-stage retrieval – which still relies on exact matches.

A potential solution would be to use PLMs to learn dense representations to address the vocabulary mismatch problem directly in the initial retrieval stage. This is, however, the subject of the next chapter. In this chapter, we focus on another line of research which proposes to enrich or learn query and/or document sparse representations based on PLMs, in order to achieve better alignment. Such techniques revisit older expansion ideas discussed in Section 2.2, which have a rich history in IR, dating back many decades [31]. We start by presenting query expansion techniques based on pseudo-relevance feedback in Section 5.2. In Section 5.3, we move to document expansion methods which propose to expand or re-weight document terms to alleviate the vocabulary mismatch problem. Instead of focusing on the textual content (i.e., terms), models covered in Section 5.4 manipulate query and document representations produced by neural encoders to learn better sparse representations.

2 *Query Expansion*

The basic idea behind query expansion is to augment a query with additional terms that are likely to appear in relevant documents, and hence mitigating the vocabulary mismatch problem. For example, the query "Covid symptoms" can be augmented with the term "coronavirus". The augmented query increases the likelihood of matching relevant documents from the corpus which use terms not present in the original query.

Recently, researchers revisited query expansion for improving the ranking effectiveness of BERT-based models. Padaki et al. [182] propose generating better queries for BERT-based rerankers [1]. The authors leverage Google's query reformulation suggestions for the topic titles (keywords) to obtain natural language question candidates. They only retain well-formed questions, semantically similar to the original topic descriptions. While reranking using these suggestions was not as effective as reranking using the original topic descriptions, they still improved over reranking with titles.

Alternatively, Naseri et al. [169] investigate how BERT can be leveraged for query expansion through pseudo-relevance feedback (PRF). PRF leverages the top-k documents from an initial retrieval in response to the initial query, which are *assumed* to be relevant, to derive significant terms to add to the query (see Section 2.2). Naseri et al. [169] devise an unsupervised contextualized query expansion model, namely CEQE, leveraging BERT to improve the selection of expansion terms. CEQE combines contextual representations, produced by monoBERT for the top-k initial candidates, with probabilistic language models to select expansion terms. Results show that CEQE improves substantially over expansion with static embeddings, demonstrating the superiority of contextual embeddings. However, improvements over the traditional RM3 expansion method are smaller and come at a much higher computational cost, since it requires BERT inference over the top-k candidates from the initial retrieval.

On the other hand, Zheng et al. [270] propose an intuitive approach that leverages the strength of BERT to select relevant document chunks for query expansion. The authors devise BERT-QE which extends the pre-BERT neural pseudo-relevance feedback (NPRF) model proposed by Li et al. [121]. While NPRF uses feedback documents directly for query expansion, BERT-QE refines the expansion using only relevant chunks from the feedback documents. To do this, the model exploits monoBERT to rerank the initial retrieved documents in three sequential phases. First, the top-k_d documents returned by monoBERT are used as PRF documents. Then, these PRF documents are segmented into fixed-length chunks, and the relevance of each chunk is evaluated independently w.r.t the query to identify the top-k_c feedback chunks.

1. Lin et al. [129] argue that this work can be better qualified as query reformulation than query expansion.

Finally, the relevance of a document is computed as the interpolation of its score w.r.t the original query, and its score w.r.t the feedback chunks. The effectiveness of BERT-QE expansion has been proven empirically on standard TREC document collections (i.e., Robust04 and GOV2).

3 *Document Expansion and Term Re-weighting*

Mirroring query expansion, document expansion augment the original document text with additional terms that are representative of their content and topic or query terms for which the document can be relevant. While document expansion dates many decades, it did not encounter the same success as query expansion. However, this thread of research has recently regained interest.

3.1 *Query Prediction for Document Expansion*

Nogueira et al. [178] propose the first successful application of a transformer-based sequence-to-sequence model for document expansion, namely doc2query. This last was trained, from scratch (i.e., not pre-trained), on the MS MARCO collection to generate the query given its relevant passage. At inference time, doc2query is used to generate a list of queries for which the input document could be relevant. The generated queries are then appended, sequentially, to the end of the original document as expansion terms without any special markup to distinguish them from the original content. Finally, the expanded documents can be indexed as usual, and used directly in a first-stage retrieval in a multi-stage reranking architecture without any further modification.

In a follow up work, Nogueira et al. [175] explore the use of the pre-trained sequence-to-sequence model, T5 [197] as replacement for the transformer model in doc2query (which was not pre-trained), and called the resulting document expansion model docT5query.

Experimental results show that BM25 results improve significantly when using the expanded document index, on the MS MARCO passage ranking test collection. The results also demonstrate, clearly, the superiority of using the pre-trained T5 model compared to the non pre-trained transformer. The authors also show that the improvements from document expansion and query expansion using PRF (with RM3) are additive.

Manual inspection of the generated expansion queries reveal that doc2query/docT5query tend to copy some terms from the input document. In other words, the models perform term re-weighting by increasing the importance (e.g., term-frequency for BM25) of key terms. Nevertheless, the models also produce novel expansion terms (i.e., synonyms or semantically related terms) not present in the original document, which represent 31% of the generated

terms. These new terms are learned from the training data to mitigate the gap between queries and relevant documents that might not contain query terms. That is, doc2query/docT5query are learning how to bridge the vocabulary mismatch.

Lin et al. [129] investigate further the impact of the novel expansion terms compared to copied terms from the original content on the MS MARCO passage ranking test collection. Their results show that expanding with copied terms alone yield bigger gains, suggesting that term re-weighting has more impact than attempts to enrich the vocabulary with novel terms. Nonetheless, combining both types of terms lead to a big jump in ranking effectiveness, indicating that both types are complementary, and the gain from both is greater than the sum of their individual contributions. This observation suggests complex interactions between copied and novel expansion terms that are yet to be uncovered. An additional interesting result show that discarding the original document content and using only the expansion terms yields surprisingly high effectiveness, only slightly worse than the full combination of content and both expansion types. This suggests that the original content can, to a large extent, be replaced by the predicted queries for BoW search [129].

3.2 *Term Re-weighting based on Contextualized Representations*

Alternatively to adding term copies to the original document content to achieve term re-weighting, Dai and Callan [53] propose to directly estimate the weight of a term from its context to improve first-stage retrieval. They devise the Deep Contextualized Term Weighting (DeepCT) model, that is, a new term weighting model based on BERT's contextualized representations to replace the traditional frequency-based term weighting in first-stage retrieval, e.g., BM25 (se Section 2.2). The motivation behind this new weighting model, is that term frequency does not necessarily indicate whether a term is important or central to the meaning of the text, especially when the frequency distribution is flat, such as in sentences and short passages. In contrast, BERT have been shown to capture the characteristics of a term's semantics and syntax, and more importantly, how they vary across linguistic contexts (see Section 3.7), offering a "better" way to estimate term importance.

DeepCT formulates term weighting as a regression problem, where the contextualized representation $T_{t,d}$ produced by BERT for a term t in a document d is used to estimate the importance $y_{t,d}$ of the term t:

$$\hat{y}_{t,d} = W \cdot T_{t,d} + b \tag{5.1}$$

where W and b are the linear combination weights and bias.

In order to train DeepCT, the model requires proper target term weights which should reflect whether a term is essential to the document or not. Dai

and Callan [53] propose *query term recall* as an estimation of the ground truth document term importance, that is:

$$QTR(t,d) = \frac{|Q_{d,t}|}{|Q_d|} \tag{5.2}$$

where Q_d is the set of queries for which the document d is judged relevant, and $Q_{d,t}$ is the subset of Q_d that contains t.

Query term recall is based on the assumption that search queries can reflect the key idea of a document. Terms that appear in relevant queries are more important than other terms in the document.

The training is then conducted on MS MARCO query-passage pairs. DeepCT takes the passage as input and outputs term weight estimates using Eq.5.1. The model is trained end-to-end to minimize the mean square error (MSE) loss to the target QTR weights:

$$L = \sum_d \sum_t (\hat{y}_{t,d} - QTR(t,d)) \tag{5.3}$$

At inference time, DeepCT is applied on all documents in the corpus to produce term weights. Then, Dai and Callan [53] propose a simple trick to integrate DeepCT weights directly in existing indexing algorithms without any modification. They first rescale term weights from $[0..1]$ to integers in $[0..100]$ so they can be interpreted as term frequencies. Then, they create pseudo-documents in which terms are repeated according to their term weight. For example, if the term "apple" is assigned a weight of 3, it is repeated 3 times "apple apple apple" so its term-frequency is equal to its DeepCT integer-weight. The new corpus of pseudo-documents are then indexed as usual, and retrieval can be performed on this index using any BoW query.

Experimental results on the MS MARCO passage ranking test collection show improvements over doc2query. However, gains using the subsequent docT5query (which appeared after DeepCT) are more important. Nevertheless, DeepCT is limited to term re-weighting whereas docT5query and doc2query can further add novel expansion terms to bridge the vocabulary mismatch gap. In this regard, DeepCT can actually achieve better gains than docT5query when restricting its expansion to copied terms (i.e., re-weighting only). This suggests that DeepCT weighting approach is more effective than re-weighting terms based on duplicates in the query predictions.

While not as effective as docT5query, it is important to note that DeepCT uses $\text{BERT}_{\text{Base}}$ with an encoder-only architecture while docT5query uses T5-base with the full encoder-decoder architecture and twice the size of $\text{BERT}_{\text{Base}}$. Hence, processing entire corpora with DeepCT is much faster. Furthermore, DeepCT uses only one pass to compute term weights whereas docT5query requires n (i.e., 40 in the original paper) passes to generate all

n query predictions to be added to the original document. This addition of terms raises another efficiency issue where keyword search latency is increased due to the increased length of the documents. In contrast, DeepCT is more efficient [150].

Later, Dai and Callan [52] propose HDCT, an extension of DeepCT for handling long documents. Similarly to previous BERT applications covered in the previous chapter, HDCT splits long documents into passages using a sliding window of about 300 terms. Each passage is independently processed as in DeepCT following the regression model defined in Eq.5.1. The term weights are then rescaled to term-frequency like integers tf_{BERT}, and we obtain a bag-of-words vector representation for each passage p_i:

$$P\text{-}BoW_{HDCT}(p) = [tf_{BERT}(t_1, p), ..., tf_{BERT}(t_m, p)] \tag{5.4}$$

At the end, HDCT generates a sequence of bag-of-words passage vectors:

$$\{P\text{-}BoW_{HDCT}(p_1), ..., P\text{-}BoW_{HDCT}(p_n)\} \tag{5.5}$$

The importance weight of each term t in a document d is finally determined by:

$$D\text{-}BoW_{HDCT}(d) = \sum_{i=1}^{n} pw_i \times P\text{-}BoW_{HDCT}(p_i) \tag{5.6}$$

where pw_i is the weight associated to the passage p_i. Dai and Callan [52] explore two passage weighting schemes: summing across passages $pw_i = 1$, and weight decay over passage position in the document, i.e., $pw_i = 1/i$. The authors find that the decay scheme is slightly more effective for news articles, however, the summing scheme appears to be more robust across news articles and web pages.

At its core, HDCT uses a passage-level pre-token regression task to train the parameters of BERT and the regression layer parameters (i.e., W and b in Eq.5.1). However, it is impractical to manually label the importance of every term in every passage of a document. Thus, Dai and Callan [52] propose three strategies to automatically generate training labels using: (1) a content-based approach using metadata for cases where only the documents are available, (2) a relevance-based approach for cases where rich query-document relevance assessments are available, and (3) a pseudo-relevance based approach using existing retrieval systems (e.g., BM25) for cases where search queries can be collected but the relevance labels or user activities are not accessible.

Experimental results show that the content-based weak-supervision strategy achieve significant and robust improvements over standard term-frequency based retrieval models. Using PRF-based weak-supervision from in-domain queries is more effective than using document metadata whereas PRF from

out-of-domain queries is worse. Finally, using search-specific labels (i.e., relevance-based) such as queries and clicks can improve HDCT by aligning it with the user search intents.

3.3 *Combining Term Expansion with Term Re-weighting*

While DeepCT has proven to be effective, it has a number of drawbacks. In fact, the context-based weighting in DeepCT only learns the term-frequency component of term weights, but still relies on the remaining components of the BM25 scoring function (see Section 2.2) via the generation of pseudo-documents. Moreover, DeepCT only assigns weights to terms that appear in the document, which limits retrieval to exact matches. In other words, it does not address the vocabulary mismatch problem, as opposed to the expansion in doc2query which, however, relies on unchanged BM25 scoring.

Mallia et al. [155] combine the strengths of doc2query and DeepCT, and propose DeepImpact which brings together the two key ideas: expansion and context-based term-weighting. DeepImpact first expands documents using docT5query, and then uses a BERT-based weighting model to estimate the importance of terms in the expanded documents (i.e., both existing and expansion terms). The term weights are produced by feeding the contextualized representations of the expanded documents produced by BERT to a two-layer MLP. Differently from the regression task in DeepCT, DeepImpact weighting model is trained with a pairwise loss between relevant and non-relevant documents with respect to a query. To be more specific, the model weights are optimized to maximize the difference between the scores of the relevant and non-relevant documents w.r.t the query.

The trained model is then used to predict the weights of each term in the expanded documents. Instead of using the real-valued weights produced by the weighting model, the authors propose to quantize these weights to a range of $[1, 2^b - 1]$, where b is the number of bits used to store the weight which is set to $b = 8$. These quantized term-weights (i.e., integer weights) are then used to index the expanded documents. Lin and Ma [128] argue that it would be more accurate to call these weights *learned impact scores*, since at inference time, query–document scores are simply the sum of weights of document terms that are found in the query. This draws an explicit connection to a thread of research in IR dating back two decades which exploits query evaluation optimization based on integer arithmetic [4], as opposed to floating-point operations used in BM25.

Experimental results on the MS MARCO passage test ranking collection demonstrate the effectiveness of DeepImpact which is better than both docT5query and DeepCT. The results also show that the ranking performance of DeepImpact gets closer to the effectiveness of BM25 + monoBERT reranking pipeline which is impressive considering that it is more than an

order of magnitude faster. Interestingly, even if DeepImpact is more effective than docT5query, they both yield the same performance when combined with monoBERT in a reranking pipeline, since both models have similar recall (i.e., Recall@1000).

4 *Learning Sparse Expansions and Representations*

The methods we reviewed so far involve augmenting the textual content of queries of documents or re-weighting existing terms. In contrast to these text-based approaches, other directions explore how to learn sparse representations from the dense vector outputs of PLMs. The first direction investigates approaches to learning term weights in the whole vocabulary. These approaches can be categorized as learned sparse-expansions as we will see in Section 4.1. Different from the above methods that improve sparse representations in explicit symbolic space (i.e., terms of the vocabulary), methods in the second direction focus on learning sparse representations for queries and documents in the latent term space (i.e., dimensions in the sparse vector have no clear concepts).

4.1 *Learning Sparse Expansions*

Bai et al. [8] propose SparTerm which aims at improving the representation capacity of BoW methods for semantic-level matching, while still keeping its advantages. SparTerm directly learn sparse term representations in the full vocabulary space. It first predicts the term importance distribution in the vocabulary space based on contextual token representations produced by BERT. Based on this, it re-weights existing and expand terms simultaneously, as opposed to DeepImpact which leverages docT5query for expansion prior to term weighting. Then, a gating controller is used to ensure the sparsity of the final representation.

To be more specific, SparTerm predicts term importance in the WordPiece vocabulary space V based on the logits of the Masked Language Model (MLM) layer. The final sequence-level representation is then obtained by summing importance predictors over all tokens conditioned with a binary activation from the gating controller. The authors explore two sparsification controllers: (1) Literal-only where only tokens appearing in the input sequence are activated, and (2) expansion-aware where the activation is *learned* to include tokens that have the potential to bridge the vocabulary mismatch gap.

Later, Formal et al. [69] propose SPALDE to improve SparTerm, by introducing a logarithmic activation to prevent some tokens from dominating the representation, and a FLOPS regularization [187] loss for learning sparse

representations. In a follow-up work, Formal et al. [68] further improve the effectiveness of SPLADE in their "v2" model using a better pooling mechanism for generating sequence-level representations, on the one hand. On the other hand, the authors apply the training improvements developed in the context of dense retrieval [2], including knowledge distillation from a more powerful cross-encoder teacher and sophisticated hard negative mining. We present these techniques in detail in the following chapter in Section 6.2.

4.2 *Learning Sparse Representations*

In contrast to weighting terms in the symbolic space, sparse representation learning methods focus on building sparse vectors for queries and documents, where representations are expected to capture semantic meanings of each input text. In this way, queries and documents are represented in the latent space. This thread of research dates back only a few years ago to 2018 right before the "BERT revolution".

SNRM [260] is the pioneer to learn sparse representations for first-stage retrieval. SNRM learns standalone sparse representations for each query and document to capture semantic relationships between them, which shows better retrieval effectiveness over traditional baselines. We refer the reader to the original paper for more details.

Recently, Jang et al. [102] proposed UHD-BERT, which learns extremely high dimensional representations with controllable sparsity based on PLMs. More specifically, it uses a BERT encoder to generate dense token representations for both queries and documents, and maps them to sparse high-dimensional representation using a *Winner-Take-all* (WTA) model [153]. WTA is fundamentally a linear layer that only preserves top-k activation and sets the others to zero. This means that the outputs' sparsity can be controller through the parameter k. Finally, UHD-BERT generates the sparse query/-document representation by token-wise max pooling.

5 *Conclusion*

Query and document expansion techniques have been explored for many decades in an attempt to mitigate the vocabulary mismatch problem in IR. With the arrival of BERT, the community regained interest in expansion techniques to investigate how the strength of pre-trained contextualized representations can be leveraged to improve the first-stage retrieval effectiveness.

2. It is important to note that sparse retrieval approaches (such as SPLADE) were developed recently in parallel with dense retrieval models. Notably, these models share the same bi-encoder core design, which we present in Chapter 6, for generating the dense contextualized representations of input tokens.

As opposed to pre-BERT expansion techniques, document expansion with models such as doc2query, DeepCT, or DeepImpact, using contextualized representations, has shown to be effective and easy to plug and use. These models shift the computationally expensive inference with PLMs from query time to indexing time. On the other hand, query expansion techniques such as CEQE show modest gains in effectiveness (e.g., compared to traditional RM3) and come with a high computational cost.

More recent sparse representation learning approaches such as SPLADE, fueled by traditional inverted indexing techniques, have seen a growing interest, inheriting from desirable IR priors such as explicit lexical matching.

The next chapter covers a research thread that moves beyond sparse representations to learned dense representations for retrieval. Retrieval is formulated as a representational learning problem where the task is to learn (transformer-based) encoders that map queries and documents into (single or multiple) dense vectors, and relevance is computed based on semantic matches between these vectors. The main motivation behind dense retrieval is to mitigate the vocabulary mismatch problem by directly capturing semantic matches from continuous dense representations.

6

BERT for dense retrieval

1 *Introduction*

Classical bag-of-words (BoW) information retrieval (IR) models, such as BM25, have served as the workhorse of most modern search systems over the past several decades (see Section 2.2). These models rely on exact matches between the query and document terms, and carry out search efficiently with inverted indexes. Such indexes encode statistical term properties (e.g., term frequencies, term positions, etc), however they do not capture their semantics. Instead, queries and/or documents can be expanded, prior to retrieval, to bridge the *vocabulary mismatch* gap.

Over the past few years, advances in representation learning [16] resulted in a shift away from sparse signals, mostly limited to exact matches, towards continuous dense representations that are able to capture semantic meanings of input texts for better relevance evaluation. This shift instigated a new wave of neural models which directly perform first-stage retrieval using learned dense encodings of documents and queries. This is known as *dense retrieval*, and it is the focus of this chapter. Dense retrieval has the potential to overcome the vocabulary mismatch problem that is known to plague exact matching-based systems, by preforming semantic matching. Moreover, it explores alternative architectures which encode queries and documents, *separately*, into dense representations, and relevance is then computed as simple vector similarities (e.g., inner product) between the query and document representations. This design is called *bi-encoder* [1] as it involves two encoders, in contrast with the *cross-encoder* design exemplified by monoBERT [2] where both query and document are fed together through one encoder, which performs all-to-all attention across tokens in the input [98]. We present this general design in more detail in Section 6.2.

As opposed to cross-encoder models which require extensive computation on each candidate document, dense retrieval models can be easily applied for full-collection retrieval on large corpora thanks to efficient algorithms for inner product search. We present an overview of these algorithms, which try to solve the nearest neighbour search problem, in Section 6.3.

Once we have defined the basic formulation of dense retrieval and supporting infrastructure, we move to the presentation of its different approaches.

1. Also called dual encoder, twin/two-tower architecture, or Siamese architecture.
2. All models presented in Chapter 4 are cross-encoders.

We first present "single-vector" approaches in Section 6.4, which encode each query and each document into a *single* dense vector, and relevance is modeled as a simple similarity function such as dot product between both vectors. Then, we present "multi-vector" approaches, which attempt to improve the effectiveness of the basic single-vector design, in Section 6.5. Multi-vector bi-encoders encode queries and/or documents at a finer-granularity into multiple dense vectors, and relevance is estimated using more complex similarity functions. Finally, we present in Section 6.6 an overview of techniques devised for improving the effectiveness of dense retrieval models including knowledge distillation from more expressive architectures, hard negative mining, and improved training.

2 *Dense Retrieval*

A typical dense retrieval model consists of two encoders to learn dense representations for queries and documents independently, denoted Φ_q and Φ_d, respectively. Then, a similarity function F (e.g., dot product or cosine similarity) is used to calculate the relevance scores based on the learned representations. In this way, the relevance of a document d_i from a corpus $\mathcal{C}$ w.r.t a given query q is formulated as follows:

$$R(d_i, q) = P(Relevant = 1|d_i, q) \triangleq F(\Phi_q(q), \Phi_d(d_i)) \tag{6.1}$$

Each encoder is trained to map the sequence of query or document tokens into dense representation vectors (usually, of fixed width), such that the similarity, computed with F, is maximized for documents relevant to a query and minimized for non-relevant documents to a query. The dense representations typically consist of hundreds of dimensions, each with a non-zero value, in contrast to sparse representations where the number of dimensions is higher (typically, equal to the vocabulary size), with most dimensions being zero.

The similarity function F between the query and document dense representations is usually symmetric, i.e., $F(\Phi(u), \Phi(v)) = F(\Phi(v), \Phi(u))$. Importantly, F needs to be computationally efficient to allow fast inference at query time. Consequently, the similarity function is typically defined in terms of vector similarity such as inner product or cosine similarity between the query and document representations. Even though, F can be defined by a deep neural network such as transformer layers or even a monoBERT model, this would make full-collection retrieval impractical, and hence bind the bi-encoder to a reranking setting. Nevertheless, we count a few designs in the literature which involve a deep-neural based similarity mechanism, as we will see in Section 6.5.

Considering this formulation, dense retrieval approaches are architecturally similar to pre-BERT representation-based models discussed in Section 4.1,

except that more powerful transformer-based encoders are used to encode query and document texts. For the similarity function inner product is still widely used, however richer query-document interaction mechanisms are also explored, as we will cover in Section 6.5. We have already discussed the thin line between semantic similarity and query-document relevance task in Section 4.1. Even though, these tasks are fundamentally different, e.g., relevance is not symmetric, queries are usually shorter than documents, etc, still they are addressed in the same manner with learned dense representations [129] using the same formulation in Eq. 6.1. Nevertheless, the task differences can manifest in the design choices. For instance, should we learn distinct weights for Φ_q and Φ_d to reflect the differences between queries and documents? Should we even use the same architecture for both encoders? As we will see, different design choices are used in the literature.

As opposed to multi-stage reranking models presented in Chapter 4, dense retrieval models, including both encoders and similarity function, can be fine-tuned *end-to-end* on labelled data, as well as out-of-domain datasets and used in zero-shot transfer settings (even though their out-of-domain generalization abilities are limited [232]). Moreover, the bi-encoder design encodes documents independently from queries. This allows the precomputing and storing of document representations prior to query time, hence pushing the expensive transformer-based document inference into the preprocessing stage. As a result, bi-encoders offer a more efficient alternative for using PLMs. Although the query representation needs to be computed at query time, only a single inference is required, and over a relatively short text sequence. Additionally, the similarity function is designed to be fast, and retrieval from a large (precomputed) collection of document representations can be efficiently implemented based on nearest neighbour search algorithms, as we will discuss in the following section.

3 *Nearest Neighbour Search*

Despite all the progress bought by dense representation learning, dense retrieval is no different from statistical sparse retrieval (e.g., BM25) developed many decades ago in terms of retrieval process. This last boils down to: (1) building an efficient queryable index for each document in the corpus, (2) retrieving a set of k candidates for a given query, and (3) computing a relevance score for each candidate. This index-retrieve-then-rank blueprint has withstood the test of time and has rarely been challenged [159]. The only difference though, is the use of dense vector indexes based on nearest neighbour search, in place of inverted indexes.

Considering a corpus $\mathcal{C} = \{d_i\}$ comprising a large number of documents, the dense representations $\Phi_d(d_i)$ of all documents $d_i \in \mathcal{C}$ can be precomputed

and stored prior to query time. Inferring over the entire corpus is computationally expensive, but this inference is parallel and can be distributed on a large cluster. When a query q arrives, its dense representation is generated through $\Phi_q(q)$, and then the retrieval task is about finding the top-k most similar document representations measured in terms of F (Eq. 6.1). This is known as the nearest neighbour search problem [221, 1].

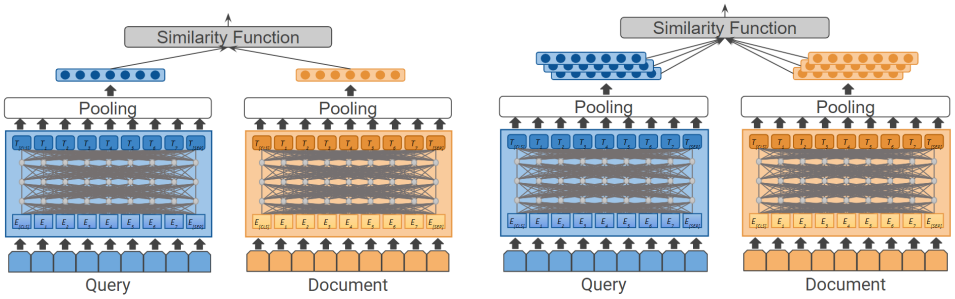
The simplest approach to the nearest neighbor search is brute-force search, which scans all the candidates and computes similarity scores one by one. However, the brute-force search becomes impractical for corpora beyond a certain size. Thus, most research resorts to an *approximate nearest neighbour* (ANN) search [6, 125], which trades off a slight loss in precision for multiple orders of magnitude improvements in speed. Guo et al. [82] categorize existing ANN search algorithms into four major types: tree-based [12, 18], hashing-based [55, 99], quantization-based [77, 103], and proximity graph approaches [115, 154]. The earliest solutions to ANN search were based on locality-sensitive hashing [99], but currently, methods based on hierarchical navigable small world (HNSW) graphs yield state-of-the-art performance based on a popular benchmark [3]. Graph-based methods build the index by retaining the neighborhood information for each individual data point toward other data points or a set of pivot points. Then, various greedy heuristics are proposed to navigate the proximity graph for a given query point. A popular open-source library for ANN is Faiss [107], which is widely adopted in the models discussed in this chapter.

4 *Single-vector Bi-Encoders*

We present here the first class of dense retrieval approaches which represent each query and document with a single fixed-size dense vector, and the similarity function is defined as a simple operation such as inner product. Given these two characteristics, the retrieval task can be cast as a nearest neighbour search problem with computationally efficient ANN solutions, as discussed in Section 6.3. The high-level architecture of single-vector bi-encoders is illustrated in Figure 6.1a.

Reimers and Gurevych [199] propose "sentence-bert", a direct instansiation of this basic bi-encoder design for sentence similarity tasks. The authors use the same PLM for both encoders – since the sentence similarity task is symmetric – and explore three pooling strategies to produce a single vector representation per sentence: (1) take the representation of the [CLS] token; (2) mean pooling across all contextualized output representations; and (3) max pooling across all contextualized output representations. At inference time, the trained encoder is applied to both sentences, and the cosine similarity

3. http://ann-benchmarks.com/



(a) Single-vector Bi-Encoder (b) Multi-vector Bi-Encoder

Figure 6.1 – Two classes of bi-encoders for dense retrieval: (a) Single-vector models encode queries and documents into single dense vectors with a simple similarity function such as inner product, and (b) multi-vector models encode queries and/or documents into a set of vectors and use a richer similarity mechanism to capture relevance

between the two resulting vectors gives the similarity score. This model provides an overview of the basic bi-encoder design, and empirical results indicate that it is less effective than its cross-encoder counterpart, however it is much faster.

Based on sentence-bert design, a number of bi-encoders were specifically developed for dense retrieval. We present hereafter the dense passage retriever (DPR) [110], the approximate nearest neighbour negative contrastive estimation (ANCE) [251], and their variants. It is interesting to note that these models come from the NLP community (DPR) and IR community (ANCE), both communities came closer to tackle dense retrieval [129].

Karpukhin et al. [110] present DPR in the context of a "retriever-reader" architecture for question-answering. In this architecture, DPR is used to retrieve candidate passages from a corpus, which are then processed by a reader to identify the exact answer span to a given question. To accomplish the retrieval task, DPR is trained with separate encoders for the question and the passages from the corpus, and use the contextualized representation of the [CLS] token produced by BERT as the output representation. The relevance between a question and a passage is then computed in terms of inner product between their dense representations.

DPR is fine-tuned end-to-end to maximize the similarity between questions and their relevant passages (i.e., containing the answer span to the question), and at the same time minimize the similarity between questions and and non-relevant passages. To achieve this, the authors train DPR with negative mining, where n negative passages are sampled for each positive passage w.r.t a question. In the experiments, negative passages are sampled from three different sources: (1) random, selected randomly from the corpus; (2) BM25, selected from the passages returned by BM25 that do not contain the

answer (also known as hard BM25 negatives); and (3) in-batch negatives, selected from other training instances in the same batch together with a mix of passages retrieved by BM25. The results of the experiments indicate clearly that approach (3) is the most effective negative sampling method, that is efficient as well since the negative passages are already in the training batch. As a consequence, training can be improved using larger batch sizes. Qu et al. [195] explored exactly this possibility in RocketQA, and find that a larger batch size (i.e., 4096 pairs) benefits effectiveness and performs as well as a monoBERT$_{\text{Large}}$ deployed on top of BM25 retrieval.

A later study by Ma et al. [144] show that a dense-sparse hybrid (linear) combination of DPR with BM25 yields interesting gains over dense retrieval alone. Similarly, Gao et al. [76] propose CLEAR which uses a bi-encoder to complement the lexical model (BM25). Unlike DPR, CLEAR uses the same encoder for both queries and documents with an additional special token inserted before the usual [CLS] token to distinguish queries ([QRY]) from documents ([DOC]). The final vector representation of the query/document is obtained via mean pooling the output representations, and inner product is used as the similarity function. The final relevance score is given by the linear combination of the lexical and dense retrieval scores. Experimental results confirm that sparse-dense hybrid models yield significant gains over dense retrieval alone.

Building on DPR results, Xiong et al. [251] hypothesize that non-relevant documents ranked highly by an exact-match BoW model such as BM25 are likely to be different from non-relevant documents ranked highly by a PLM-based bi-encoder. Thus, sampling hard negatives from BM25 results might not be the best strategy. Consequently, the authors propose ANCE to identify better negative samples that are highly ranked by the bi-encoder being trained.

ANCE adopts the same design as DPR, and takes the [CLS] output representation from RoBERTa [138] as the representation of the input query or document. However, ANCE opts for the same model for both its encoders (i.e., $\Phi_q = \Phi_d$) as opposed to DPR. During training (fine-tuning), hard negatives are sampled via ANN search over an index built from the representations generated by the encoder being trained. Instead of constantly updating the index after each batch, which is computationally impractical, the ANN index is instead rebuilt after each m batches. To avoid a cold start, the authors use BM25 negatives to begin the training process.

ANCE is fine-tuned on positive query-document pairs from the MS MARCO passage ranking collection, and negative sampled from the ANN index (built from a checkpoint of the encoder dating back m batches) using a negative likelihood loss function.

Experimental results on the MS MARCO passage ranking task demonstrate that ANCE training scheme is more effective than DPR's. The authors also

report results on the MS MARCO document ranking task from the 2019 TREC Deep Learning Track [45]. In order to handle long document, ANCE is extended following the same splitting technique used in Dai and Callan [51]: FirstP, which takes the first 512-token passage of the document, and MaxP, which takes the best-scoring passage. Results show that MaxP is more effective than FirstP which is consistent with cross-encoders [51].

Zhan et al. [264] propose an extension of the ANCE model, coined ADORE, which additionally fine-tunes the query encoder. To be more specific, ADORE first follows a similar training scheme as ANCE to fine-tune the query/document encoder (shared weights). While the document encoder weights are frozen after this training stage, the query encoder is further fine-tuned to directly optimize IR metrics with dynamically sampled hard negatives. The results demonstrate the effectiveness of this additional query encoder fine-tuning.

DPR and ANCE provide a good overview of a key issue in the design of bi-encoders for dense retrieval, that is selecting negative samples for training. Empirical results from both approaches suggest that the basic single-vector approach to dense retrieval is less effective than cross-encoders, but generally more effective than lexical BM25 retrieval. Nevertheless, this loss in effectiveness was to be expected with the ablation of cross-attention between queries and documents. More importantly, bi-encoders have the advantage of performing full-collection retrieval as opposed to multi-stage reranking with cross-encoders. When it comes to the choice of same query/document encoder or separate encoders, the best choice is still unclear and each model adopts a different approach. Finally, empirical evidence indicate the potential of dense-sparse hybrids compared to dense retrieval alone.

5 *Multi-vector Bi-Encoders*

The main motivation behind the bi-encoder design is to trade off the high effectiveness of cross-encoders for efficiency gains that would enable full-collection retrieval with learned dense representations. As we have seen in the previous section, single-vector systems are the most basic instansiation of the bi-encoder design. Owing to the use of simple similarity functions such as inner product, the retrieval task is easily cast as a nearest neighbour search problem, with efficient scalable off-the-shelf solutions (see Section 6.3). While this can be much faster compared to reranking, single-vector bi-encoders are still less effective than cross-encoder rerankers (except for RocketQA) which benefit from cross-attention (i.e., interaction) between the query and the document tokens across all transformer layers in the PLM. Thus, researchers

started to explore different effectiveness/efficiency trade-offs by relaxing the constraints of the single-vector design.

MacAvaney et al. [147] investigate how to enhance the efficieny of document reranking by precomputing the token representations of documents. The proposed model, called PreTTR, is a hybrid model between the bi-encoder and cross-encoder architecture. PreTTR is a modified monoBERT model, in which lower layers are used for encoding the query and the document *separately* using masking to avoid query-document cross-attention, while the upper layers are used as usual with all-to-all attention between query and document tokens. Considering that monoBERT has L transformer layers and following the bi-encoder terminology, the lower $L - k$ layers represent the encoders (Φ_q and Φ_d) separated via masking, and the upper k layers represent the similarity function F. This means, that the document representations from the $L - k$ first layers can be precomputed and stored (offline part), while the query needs to be encoded online. Then, the precomputed document and online query contextual token representations are fed into the k last layers of monoBERT to estimate the relevance score. The number k of interaction layers is configurable to a trade-off between the model effectiveness and efficiency.

Similarly, Gao et al. [73] propose the MORES framework where queries and documents are encoded separately with two encoders, but instead of a simple similarity function, the authors devise transformers blocks with one-way attention from query to document representations for relevance estimation.

While these hybrid designs offer some gains in efficiency compared to cross-encoders, their transformer-based similarity functions, however, restrain them to the reranking setting, as existing ANN solutions do no support these neural functions. In the remainder of this section, we focus on alternative bi-encoder architectures relaxing the single-vector representation constraint by representing queries and/or documents with *multiple* representation vectors as illustrated in Figure 6.1b.

5.1 *Multiple Query Representations*

Humeau et al. [98] proposed Poly-encoders, an architecture with an additional learned attention mechanism to represent more global features for the task of response selection. Poly-encoders use two separate models to encode contexts and candidates. The candidate is encoded into a single vector, while the input context, which usually includes more information than a candidate, is represented with m vectors instead of just one. The m vectors are then attended using the candidate vector to get the final score (one-way attention). The value of m gives a trade-off between inference accuracy and speed.

It is important to note that different from general retrieval tasks where retrieved texts (documents) are usually longer than input texts (queries), the

response selection task in the work of Humeau et al. [98] has longer input texts than retrieved texts, and thus the multi-vector representation model is actually employed for the "query encoder".

5.2 *Multiple Document Representations*

Based on the same intuition, Luan et al. [141] raise the limited capacity of single-vector representation to support retrieval of documents which are often lengthy and have diverse aspects in them, while queries are usually short and have focused topics. The authors propose Multi-Vector BERT (ME-BERT), a bi-encoder model which encodes queries into single vectors and document into multiple vectors. The query representation is defined as the contextualized embedding of the special token [CLS], and the multi-vector document representation as the contextualized vectors of the first m tokens in the document. The value of m is always smaller than the total number of tokens in the document. Finally, the relevance score is calculated as the largest inner product yielded by each document vector with the unique query vector.

ME-BERT was trained using a combination of BM25 negatives as well as in-batch negatives, as in DPR. Experimental results show that the ME-BERT model yields strong performance compared to its DE-BERT single-vector case (with $m = 1$, meaning the [CLS] token is used for the document representation), as well as DPR. However, it is as performant as ANCE (which came after) which uses single-vector representations.

Additionally, Luan et al. [141] combine the dense retrieval results with sparse BM25 retrieval using a linear combination of scores to build a more effective dense-sparse hybrid model; similarly to [144].

5.3 *Per-Token Representations and Late Interactions*

In Khattab and Zaharia [112], the authors push the multi-vector idea to its logical extreme and generate a dense vector for each token in both queries and documents in their proposed ColBERT model. For relevance estimation, the authors devise "late interactions" a rich yet scalable similarity function to model fine-grained matching signals between the query-document token-level dense vectors, as shown in Figure 6.2.

Formally, given a query q with n tokens $q_{1:n}$ and a document d comprising m tokens $d_{1:m}$, ColBERT first encodes them by feeding them through the same BERT encoder ($\Phi_q = \Phi_d = \Phi$), but distinguish input sequences that

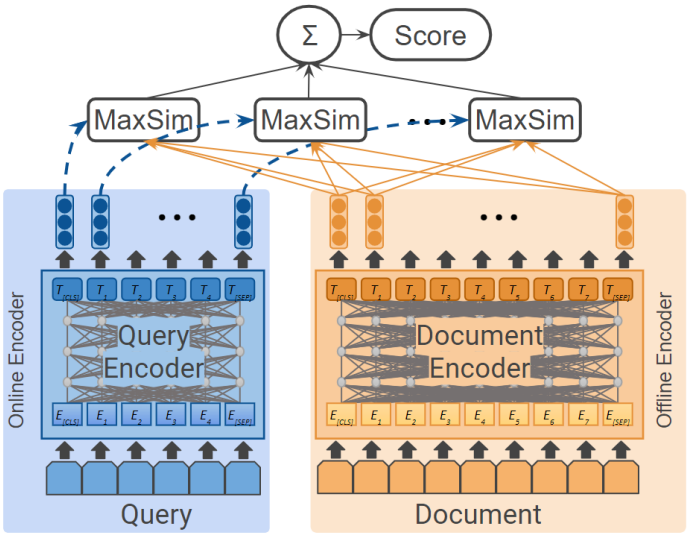


Figure 6.2 – The architecture of ColBERT [112]

correspond to queries and documents by prepending a special token [Q] to queries and another token [D] to documents, after the [CLS] token:

$$\Phi(q_{1:n}) = BERT([CLS][Q]q_1...q_n[MASK]...[MASK]) \tag{6.2}$$

$$\Phi(d_{1:m}) = BERT([CLS][D]d_1...d_m) \tag{6.3}$$

When encoding queries, if the query has fewer than a predefined number of tokens, it is padded with BERT's special [mask] tokens. This padded sequence of input tokens is then passed to the BERT encoder as shown in Eq. 6.2. This padding is to be considered as *query augmentation*, a step that allows BERT to produce query-based embeddings at the positions corresponding to these masks. Query augmentation is intended to serve as a soft, differentiable mechanism for learning to expand queries with new terms or to re-weigh existing terms based on their importance for matching the query [112].

The relevance (similarity) score is then computed using the *MaxSim* operation, which identifies the closest-matching document token among $\Phi(d_{1:m}) \in \mathbb{R}^{m \times D}$ for each query token in $\Phi(q_{1:n}) \in \mathbb{R}^{n \times D}$, D being the vector dimension:

$$R(d,q) \triangleq F(\Phi(q_{1:n}), \Phi(d_{1:m})) = \sum_{i=1}^{n} \max_{j=1}^{m} \Phi(q_i) \cdot \Phi(d_j) \tag{6.4}$$

Since the MaxSim operation is not practical for full collection retrieval, ColBERT adopts an efficient two-step process in order to perform top-k ranking. The authors use the Faiss library [107], to build a dense index of the token-level representations for each document in the corpus. Then at query time, ANN search on this index is employed, in a first step, to fetch

the top k' (e.g., k' = k/2) most similar token representations for each query token representation in $\Phi(q_{1:n})$. Then, each of the retrieved representations is mapped to its document of origin, producing $n \times k'$ document IDs, only $K \leq n \times k'$ of which are unique. These K documents likely contain one or more representations that are highly similar to the query embeddings. For the second step, the retrieved candidates list is refined by exhaustively reranking only those K documents according to the MaxSim operator in Eq. 6.4. In the end, this two-step ranking process resembles the multi-stage reranking architecture, where the nearest neighbour search on the dense-index search replaces the usual first-stage retrieval on inverted indexes.

ColBERT has, however, the advantage of being differentiable end-to-end. The whole model is fine-tuned using triples of $\langle q, d^+, d^- \rangle$ with query q, positive document d^+, and negative document d^-, to produce a score for each document individually and is optimized via pairwise softmax cross-entropy loss over the computed scores of d^+ and d^- w.r.t q.

By decomposing relevance modeling into token-level computations, ColBERT offers an efficient alternative to cross-encoders with modest degradation in effectiveness compared to monoBERT reranking. However, this added expressivity comes at a cost: late interactions impose an order-of-magnitude larger space footprint than single-vector models, as they must store billions of token-level vectors for Web-scale collections.

Similarly to ColBERT, Gao et al. [75] devise a new model, named COIL, but query-document interactions are restricted to exact matching tokens only. With COIL, the main contribution is a new lexical matching scheme that uses vector similarities between query-document overlapping token contextualized representations to replace heuristic scoring used in classical methods. To support full-collection retrieval, the authors devise new contextualized inverted lists, in which representations are grouped by their surface tokens to facilitate exact match search of query tokens. Experimental results show that COIL performs on par with the richer all-to-all matching in ColBERT on the MS MARCO passage ranking task, provided that the inner product between the [CLS] representations of the query and document is added to the MaxSim score from the exact matches. However, results on long document ranking show that adding the [CLS] matching does not bring additional gains over considering only the score from the exact matches.

UniCOIL [128] further simplifies the approach in COIL by learning a single weight per term (i.e., the dimension of the dense vector is compressed to 1) to build a sparse retrieval model, extending previous methods like DeepCT and DeepImpact which we discuss in Chapter 5.

Recently, Santhanam et al. [217] address the challenge posed by the index space-requirements of multi-vector models, and propose ColBERTv2 which improves the quality of ColBERT while reducing its space footprint. The authors make the observation that late interactions naturally produce a

lightweight semantic space and show that token representations produced by ColBERT localize in a small number of regions corresponding to contextual "senses" of a token. Hence, this semantic space can be summarized, with high precision, by a set of cluster centroids along with minor refinements at the dimension level. ColBERTv2 leverages this observation to greatly reduce storage requirements and achieve state-of-the-art dense retrieval quality.

To be more specific, ColBERTv2 follows the same architecture as ColBERT, but it is trained using a combination knowledge distillation and hard negative mining to boost its quality (we discuss these techniques in the next Section 6.6). Moreover, ColBERTv2 uses a *residual compression* mechanism on top of the encoder outputs to reduce the space footprint of late interaction. Given a set of cluster centroids $C_{1:N}$, ColBERTv2 maps each token representation v (i.e., $\Phi(t)$) to its closest centroid C_i, and adds a *quantized* vector $\tilde{r}$ that approximates the residual $r = v - C_i$. At query time, the centroid index i, and the residual $\tilde{r}$ are used to recover an approximate $\tilde{v} = C_i + \tilde{r}$.

To support fast nearest neighbour search, ColBERTv2 precomputes all document (i.e., passages) representations and oraganizes them as follows:

1. The set of centroid representations $C_{1:N}$ is obtained by applying k-means clustering to the embeddings produced by invoking the BERT encoder over only a sample of the entire corpus;
2. The token-level representations of all documents in the corpus are produced using the BERT encoder and applying the centroid-based compression described in the previous paragraph;
3. The representations are grouped by their corresponding centroid ID to create inverted lists, similar to COIL. This allows fast query-token nearest neighbour search.

At query time, the retrieval follows the same two-step process defined in ColBERT, but this time the inverted lists are used to retrieve the candidate documents. That is, for every token representation in the query, the nearest centroids are found. Then, the inverted lists are used to identify the token representations, in the corpus, that are closest to these centroids. From here, the second step is used to refine the candidate documents containing these "nearest" representations using the MaxSim operator.

The multi-vector extension of the basic single-vector bi-encoder design explores different efficiency/effectiveness trade-offs. Notably, the late interaction mechanism proposed in ColBERT via the MaxSim operator supports rich token-level interactions while remaining compatible with efficient nearest neighbour search solutions available in existing libraries. The result is a dense retriever whose effectiveness is comparable to monoBERT reranking, but at a fraction of the query latency. However, the expressivity of late interaction imposes a large space footprint making retrieval impractical for large-scale

corpora. This challenge has been tackled in the recent ColBERTv2 using residual compression. In general, the progress in dense retrieval has been accompanied with a growing interest in vector compression methods to reduce the dense index space footprint. In the context of single-vector systems for example, BPR [252] learns to directly hash embeddings to binary codes using a differentiable tanh function. JPQ [263] and its extension, RepCONC [265], use Product Quantization (PQ) [81, 104] to compress embeddings, and jointly train the query encoder along with the centroids produced by PQ via a ranking-oriented loss. The centroid-based encoding in ColBERTv2 can be viewed as an extension of PQ to multi-vector representations (i.e., compression via splitting single vectors vs. matrix representations – one per token – into sub-vectors).

6 *Enhancing the Effectiveness of Bi-Encoders*

Bi-encoders allow for more efficient inference owing to the ability to precompute document representations, which enables fast similarity search using efficient nearest neighbour solutions, when presented with a candidate query. However, the basic single-vector approach to dense retrieval while being much faster, suffers losses in ranking quality. Menon et al. [157] explore the reasons behind this performance gap between bi-encoders (single vector) and cross-encoders: Does the re-ranking performance gap reflect a limitation in the inherent *capacity* of the bi-encoder's factorized (separate) representation, or in its *training*? The authors demonstrate that bi-encoders suffer from *over-fitting* and exhibit, thus, a poor *generalization* ability rather than *capacity*. This is consistent with the results obtained in the zero-shot evaluations on the BEIR benchmark [232].

Several works explored means of improving bi-encoder effectiveness using more expressive similarity functions such as PreTTR, MORES or the more efficient ColBERT late interaction presented in Section 6.5. However, the richer interaction mechanisms with multi-vector approaches present important drawbacks: Inadequacy of existing nearest neighbour search with transformer-based similarity functions (e.g., MORES and PreTTR), and the large space footprint of multi-vector dense indexes even if the similarity function is amenable to nearest neighbour search (e.g., ColBERT).

Considering these challenges, another body of work find it more fruitful to focus instead on addressing the fragility of single-vector models by introducing new supervision paradigms for negative mining, better pre-training for dense retrieval, and distillation from more expressive architectures. We present these works by their intervention: in the pre-training stage in Section 6.1, or the fine-tuning stage in Section 6.2.

6.1 *Enhancing pre-training*

In Gao and Callan [71], the authors study the internal structure of PLMs and find that models such as BERT directly out of pre-training have a non-optimal attention structure. In particular, they were not trained to aggregate sophisticated information into a single dense representation (i.e., [CLS] output representation). To resolve this discrepancy, they propose to pre-train – on the usual MLM objective – towards dense encoder with a novel Transformer architecture, Condenser, where MLM prediction CONditions on DENSE Representation.

Specifically, the transformer layers in a PLM (e.g., BERT) are divided into L^e early layers and L^l later layers, and an additional Condenser head comprising L^h more layers is added on top. The design includes a short circuit (skip-connection) from the output of the early layers directly to the Condenser head, and only the final [CLS] representation from the later layers is fed to the Condenser head. Since the later layers can refine the token representations (i.e., produced by the early layers) but can only pass new information through the [CLS] representation. The authors claim that the late [CLS] representation is therefore required to aggregate newly generated information in the later layers, and the Condenser head then *condition* on [CLS] to make the MLM predictions. Meanwhile, skip connecting the early layers, removes the burden of encoding local information and the syntactic structure of input text, focusing [CLS] on the global meaning of the input text. Layer numbers L^e and L^l control this separation of information.

After pre-training, the Condenser head is discarded and the weights of the early and late layers can be a drop-in weight replacement for a typical LM like BERT. Experiments show that Condenser improves over standard pre-trained BERT by large margins on various text retrieval and similarity tasks.

In a follow-up work, Gao and Callan [72] propose coCondenser, which adds a self-supervised target-corpus pre-training using a contrastive loss to address the fragility of bi-encoders to training-data noise. The authors also propose better negative mining through a two-round training: In the first round, the retriever is trained with BM25 negatives. The first-round retriever is then used to mine hard negatives to complement the negative pool. The second round retriever trains with the negative pool generated in the first round.

6.2 *Enhancing fine-tuning*

In order to improve the fine-tuning of bi-encoder retrievers which are often very sensitive to the specifics of supervision, a line of work propose two

directions: distillation from more expressive architectures [88, 130], including explicit denoising [195], and hard negative sampling [251, 131].

We have already discussed negative mining in DPR and ANCE in Section 6.4. In RocketQA [195], the authors design an optimized fine-tuning pipeline that not only includes large batch training, but also a sophisticated denoised negative sampling. They use a cross-encoder to remove the top-retrieved documents that are likely to be mislabelled (i.e., false negatives), and use high-confidence labelled examples from this cross-encoder for data augmentation. While this is very effective, the entire pipeline is very heavy in computation.

Another interesting direction is knowledge distillation. We have already presented distillation in Section 5.1 as a means for training smaller student models with reduced inference costs while keeping the same effectiveness as the teacher model. However, in this current thread of research, researchers explore knowledge distillation as a means for distilling "more expressive" cross-encoders into "less expressive" bi-encoders.

Hofstätter et al. [88] propose the first distillation method from cross-encoders to bi-encoders through a three-step process:

1. A cross-encoder teacher is fine-tuned using standard (query, relevant document, non-relevant document) triples (e.g., from the MS MARCO passage ranking collection);
2. The fine-tuned teacher is used to score all the training triples to generate new soft labels (i.e., this inference is done once and the labels are cached for later use);
3. The bi-encoder student is fine-tuned with the soft teacher labels via standard knowledge distillation techniques.

In their experiments, BERT $_{\text{Base}}$ is used to initialize the monoBERT teacher, and the bi-encoder student is based on a DistilBERT encoder.

The authors devise a new distillation loss for step (3), called Margin Mean Squared Error (MarginMSE) which optimizes the margin between the scores of relevant and non-relevant documents w.r.t a query. Given a training triple $\langle q, d^+, d^- \rangle$, the soft labels from the teacher model for both relevant and non-relevant documents d^+ and d^-, respectively, are used to optimize the student model as follows:

$$L(q, d^+, d^-) = MSE(M_s(q, d^+) - M_s(q, d^-), M_t(q, d^+) - M_t(q, d^-)) \quad (6.5)$$

where $M_s(q, d)$ and $M_t(q, d)$ are the scores of the document d w.r.t the query q given by the student and teacher model, respectively. MSE is the standard Mean Squared Error between the predicted scores S and the targets T, that is:

$$MSE(S, T) = \frac{1}{|S|} \cdot \sum_{s \in S, t \in T} (s - t)^2 \quad (6.6)$$

Interestingly, this distillation process can easily be extended to ensembles of teachers (e.g., mean average their scores to create the soft teacher labels) .

Experimental results on the MS MARCO passage ranking test collection demonstrate the effectiveness of distilling from more powerful cross-encoder architectures into less powerful bi-encoders. This is consistent with reranker distillation results presented in Section5.1.

Instead of precomputing teacher scores, Lin et al. [130] compute the teacher soft labels on the fly during knowledge distillation. Because of the high inference costs of cross-encoders, the authors use the faster but sufficiently effective ColBERT model as a teacher. The authors call this model TCT-ColBERT, where TCT stands for Tightly Coupled Teacher. The student model is optimized using a loss function comprised of two terms: The fist term is defined as a the softmax cross-entropy loss over the original gold labels, and the second term captures the KL-divergence between the distribution of the teachers soft labels and the student scores. As opposed to Hofstätter et al. [88] where the student is based on DistilBERT, TCT-ColBERT uses the larger BERT $_{\text{Base}}$ for the student model (i.e., which is the same size as the encoder model of the ColBERT teacher). In the experiments on the MS MARCO passage collection, TCT-ColBERT yields on-paer performance with ANCE. The authors also explore hybrid combinations with sparse retrieval results from either BM25 or docT5query which bring gains over dense retrieval alone.

In a follow-up work, Lin et al. [131] improve TCT-ColBERT in their "v2" model. The authors leverage an initially trained TCT-ColBERT to sample hard negatives for improving the effectiveness of the ColBERT teacher. This ColBERT model is then distilled into a bi-encoder student model.

Later, Hofstätter et al. [90] argue that training batches commonly assembled randomly are likely to contain many low information training examples. The authors thus propose a principled approach to building training batches. The training queries are first clustered using k-means based on an initial bi-encoder representations. Then, instead of randomly selecting queries to include in a batch, queries are sampled from the same topic cluster for the same batch so that contrastive examples are more informative. This query sampling method is called "Topic Aware Sampling" (TAS). Furtheremore, the authors propose to organize the training samples from "easy" to "dfficult" as defined by the margin from the teacher label. The authors call this "balanced" sampling, combined with TAS this gives the full TAS-B technique.

Hofstätter et al. [90] experiment with an ensemble of teachers including a cross-encoder and ColBERT, and the student model remains a DistilBERT-based bi-encoder. Results indicate that the TAS-B batch construction improves significantly over random batch construction, and gains are additive with the sparse docT5query method. Beyond this experimental setting, the proposed

TAS-B technique can be viewed as a general approach to constructing training batches for various dense retrieval models.

Contemporaneously, Zeng et al. [261] propose a generic *curriculum learning* based optimization framework called CL-DRD that controls the difficulty level of training data produced by the cross-encoder teacher model. CL-CRD is used to optimize the bi-encoder student by increasing the difficulty of the training samples. To be more specific, for each training query, the top-200 documents returned by the student dense retrieval model are reranked by the teacher model, and then divided into three groups: (1) the pseudo-relevant group comprising the first K ranked documents, (2) the hard negative group comprising the next K' documents, and (3) the remaining K'' documents in the ranked list produced by the teacher model. Then for each training interaction in the curriculum, a fixed number $L = K + K' + K''$ documents are selected with increasing numbers of documents from the pseudo-relevant group (i.e., increase K) after each iteration since this group represents the most difficult instances.

The student model is optimized with a listwise loss function to enforce it to learn preferences between documents sampled: Within the group (1) (high difficulty), from the group (1) compared to group (2) and group (3), and from the group (2) compared to group (3) (low difficulty). This means that, when K is small, the number of difficult fine-grained preferences in the group (1) is low, and since K increases after each iteration more difficult training instances are considered. In their experiments, the authors experiment with both the single-vector TAS-B model and the multi-vector ColBERTv2 model. For both models, applying the CL-CRD framework yields significant gains in performance.

In Section 6.5, we introduced ColBERTv2 and how it addresses the high storage requirements of ColBERT, however it also uses knowledge distillation with negative mining to improve its effectiveness. Starting from the original ColBERT checkpoint [112], the authors index all documents in the training corpus with ColBERTv2 compression. Then, for each training query, the top-k documents are retrieved from the index and a cross-encoder is used to rerank them. Finally, training triples are constructed form the top-k candidates by selecting a highly-ranked document, and lower-ranked document w.r.t the query. These training examples, along with in-batch negatives, are used to distill the cross-encoder scores into the ColBERT student architecture using KL-divergence loss.

7 *Conclusion*

Over the past few years, there has been a lot of progress in learning dense representations for retrieval which we have covered in this chapter. Dense

retrieval approaches adopt a bi-encoder architecture, in which queries and documents are encoded independently. We first presented the simplest dense retrieval approaches which use a single-vector bi-encoder, that represents queries and documents with single dense representations, and defines relevance in terms of inner product between the two representations. Owing to this simple formulation, efficiency is greatly improved, however single-vector systems suffer from important losses in ranking quality compared to cross-encoders.

Then, we review works which explored different efficiency/effectiveness trade-offs, and proposed to relax the single-vector design. Notably, ColBERT propose to represent queries and documents with their token-level dense vectors, and propose the rich yet scalable late interaction mechansim which is amenable to a two-step retrieval process with existing nearest neighbour search solutions. But, as we have seen, decomposing relevance modeling into token-level computations imposes an order-of-magnitude larger space footprint than single-vector models. While compression techniques can be used to address the space challenge, a lot of works prefer to focus instead on addressing the fragility of single-vector models by improving their training.

If dense retrieval yields promising performance exceeding sparse retrieval, dense-sparse hybrids appear to be more effective than either alone, suggesting that they provide *complementary* relevance signals. Nevertheless, zero-shot evaluations of various PLM-based models conducted in Thakur et al. [232] on the BEIR benchmark, reveal that dense retrievers were overall less effective than BM25. Dense retrieval models which are usually fine-tuned on MS MARCO data (i.e., in a supervised manner) appear to suffer from poor generalization ability to out-of-distribution queries and documents. Addressing this generalizability is an interesting area of research.

On the other hand, it is important to highlight the fact that most bi-encoder models report results on passage retrieval tasks, and extensions to long document collections in the BEIR benchmark truncate the document to its first passage (e.g., the first 256-512 tokens). We have also seen the use of the MaxP strategy from Dai and Callan [51] for evaluating ANCE on long document retrieval tasks. However, there has not been as much investigation in this direction as we have seen in Chapter 4 for reranking.

Finally, despite all of this progress, most IR models today follow, with a different infrastructure stack indeed, the same *index-retrieve-then-rank* process. Metzler et al. [160] stop to rethink this standard blueprint, and propose a vision to build model-based IR systems by exploiting powerful PLMs. Within this proposed framework, the index is embedded into the model itself during the training process, and the retrieval and reranking components are implemented with model inference. Recently, Tay et al. [227] implemented this new IR paradigm based on the T5 model. The significant performance is achieved by training the model with indexing (i.e., documents to docids) and

retrieval (i.e., queries to docids) in a multi-task setup. Alternatively, Zhou et al. [271] presented DynamicRetriever, which implements the model-based IR paradigm using BERT. This new IR paradigm seems exciting, however these works are only preliminary explorations and there are still many challenges to be addressed.

Part III

Contributions

7

Highlighting exact matches for ad hoc ranking with transformers

1 *Introduction*

The vocabulary mismatch problem is a major limitation of IR models based on classical BoW representations whose modeling capabilities are restricted to exact matching. We have shown in the state-of-the-art overview, in Chapter 2, that with the advent of deep learning, IR models evolved to integrate continuous representations which mitigate the vocabulary mismatch problem by enabling semantic-based matching.

Developments in neural models based on dense representations brought the task of *relevance matching* in IR closer to the task of *semantic matching* in NLP (for example, to detect if two sentences are paraphrases of each other). Models for these tasks share many architectural similarities, and there has been cross-fertilization between the two communities [129]. Nevertheless, relevance matching cannot be reduced to only semantic similarity. Guo et al. [83], Mitra et al. [165], Mitra and Craswell [164] emphasized the importance of modeling lexical matches using deep neural networks for document ranking. For queries containing rare terms such as "pekarovic land company", it is easier to estimate the relevance based on exact matches of the rare term "pekarovic". On the other hand, for a query like "What day is the winter solstice?", relevant documents are more likely to contain "21st december" than the term "day". Consequently, assessing the relevance of a document w.r.t a query requires the model to both (1) check for strict, exact term matches (e.g., key entities in the query) and (2) compute semantic similarity generalizing across related concepts.

Pre-BERT models emphasize the difference between semantic and relevance matching, such as DRMM [83] and KNRM [250], which integrate exact match counting in their neural network via histograms and gaussian kernels, respectively, or DUET [165], which identifies good patterns for lexical and semantic matches jointly. In contrast, recent developments with contextual pew-trained language models (PLMs) are now erasing the distinction between the semantic similarity and document ranking threads of work. The same generic model is used for both relevance and semantic matching tasks.

We propose a refined ranking approach emphasizing exact matching signals by introducing special marker tokens to promote the exact term overlap

Table 7.1 – Extracts from top ranked passages by Vanilla BERT for the query: "causes of left ventricular hypertrophy" from MS MARCO [9]

ID	Passage
47203	Causes of **Right** Ventricular Hypertrophy. There are four usual causes of **right** ventricular hypertrophy...
5197133	The last common cause of **right** ventricular hypertrophy is the ventricular septal defect...
7504775	The most common causes of **right** ventricle hypertrophy (RVH) are diseases that damage the lung...

between the query and the document in the textual input. We believe that exact matching signals are still important for effective relevance matching. Therefore, the approach we propose incorporates the traditional exact matching intuition into recent PLMs to precisely improve their effectiveness on the document ranking task.

The remainder of this chapter is organized as follows. Section 7.2 presents the motivation of our proposition and introduces the research questions we study. Section 7.3 details the approach and the marking strategies we investigate. In Section 7.4 we describe the experimental setup and then discuss the evaluation results in Section 7.5. We end up with a conclusion.

2 *Motivation and Research Questions*

Pre-trained Language Models such as BERT [58], ELECTRA [40] and T5 [197], have become the core components for building highly effective ranking models. Applying PLMs to document ranking is straightforward thanks to the pre-train then fine-tune recipe. The same architecture based on homogeneous transformer layers is employed regardless of the downstream task in contrast to pre-BERT neural ranking models [83, 165, 250] which design specialized neural architectural components to capture different aspects of relevance between a query and a document. In their study of BERT behavior for ranking, Qiao et al. [194] revealed that BERT prefers document terms similar to the query in search since its pre-training on surrounding contexts favors text sequence pairs that are closer in their semantic meaning. The authors conclude that BERT can be considered as an interaction-based sequence-to-sequence soft matching model that owes its effectiveness to the transformer's cross-match attention. While soft semantic matching is, undeniably, a valuable signal for relevance that alleviates the vocabulary mismatch problem, a ranking model needs proper handling of exact matching cues as well [83, 165, 140]. Let us take the following query from the MS MARCO collection [9]:

"Causes of left ventricular hypertrophy", as an example. Table 7.1 reports extracts from the top passages ranked by a vanilla monoBERT [173] reranker on top of a BM25 retriever. We can see that all top ranked passages are related to "right ventricular hypertrophy" due to the soft matching between "left" and "right". This example is a reminder of the importance of exact matching for relevance ranking.

In this contribution, we suggest that recent BERT-based ranking models can benefit from explicit exact matches highlight and study the following research questions:

— How to emphasize exact matching signals in PLMs while conserving the same architecture (i.e., benefit directly from pre-trained checkpoints)?

— What is the impact of explicit exact match signals on the effectiveness of PLMs for document ranking?

3 *Highlighting Exact Matches for Pre-trained Contextualized Language Models*

Enhancing pre-trained language models, exemplified by BERT, with the *exact matching* intuition can be achieved by appending a specialized architectural component to the transformer architecture following designs from pre-BERT models. However, BERT has proven to be effective for a wide range of downstream tasks using the same homogeneous (but highly versatile) transformer architecture. BERT is able to "figure out" how to model the target task provided that it is fine-tuned with annotated data that capture nuances of this task. Moreover, efficiency is one of BERT's major limitations. Considering its base configuration, BERT already comprises not less than a $110M$ parameters, adding more parameters specifically for capturing exact match cues would increase the computational cost of the model. Additional parameters might not even be necessary as evidence from the investigation of self-attention patterns conducted by Kovaleva et al. [117] suggests that BERT is over-parameterized. In other words, BERT has enough representation capacity to additionally learn exact match patterns for instance.

Importantly, altering BERT's architecture will cost us the immense "free" benefits of the self-supervised pre-training provided by Google's public BERT checkpoint. More so, subsequent innovations improving BERT cannot be directly integrated for "free" in the altered architecture.

Inspired by Baldini Soares et al. [10] work on relation extraction where new special tokens are introduced to highlight the entities (key tokens), we propose using *marker tokens* to emphasize exact matching terms in the textual input before feeding it to a BERT ranker, hence keeping the same model architecture and number of parameters. We hypothesize that these marker tokens can help BERT focus on the terms that are considered important for

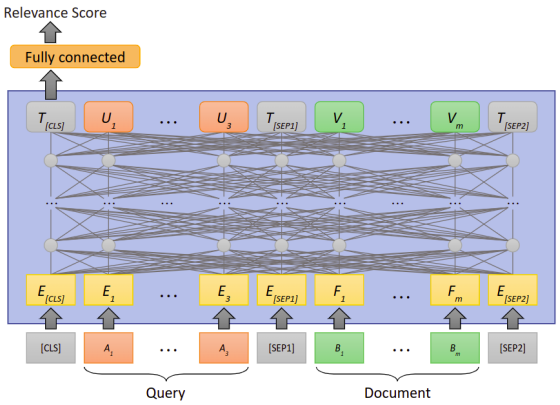


Figure 7.1 – The monoBERT architecture [173]. Copied from Figure 4.2 in Section 4.2

relevance estimation, and learn how to use the exact match hints. A part from conserving BERT's configuration, this approach mitigates the risk of introducing a systematic bias towards exact matching and bringing back the vocabulary mismatch problem.

In the remainder of this section, we first describe the general model architecture we adopt for ranking and then present in detail the marking strategies we propose to explicitly highlight exact term matches. We consider the traditional formulation of exact matching where two terms t_1 and t_2 match exactly if their *stems* are identical. We use the Porter algorithm for stemming and stop words are not considered during marking.

3.1 *Model architecture*

We adopt the monoBERT configuration described by Nogueira and Cho [173] whose architecture is shown in Figure 4.2 in Section 4.2 and repeated here as Figure 7.1.

Given a query q and a candidate document d, the input sequence S to the BERT core model is given by:

$$S = [[CLS], q, [SEP], d, [SEP]] \tag{7.1}$$

where q and d are represented with their tokens obtained after applying the WordPiece tokenizer.

The output representation of the standard [CLS] token generated by BERT is then fed through a single fully-connected layer to estimate the relevance score $R(d, q)$ of the document d w.r.t to the query q.

The whole model including the pre-trained BERT core and the randomly initialized relevance classification layer are optimized with a cross-entropy loss; see Section 4.2 for more details. This target-task fine-tuning is the only thing separating monoBERT from any other application of BERT to

a sentence-pair classification task (e.g., paraphrase detection). There is no further consideration for the particular characteristics of the ranking task in this base model.

3.2 *Exact Match Marking*

We propose different marking strategies which augment the input sequence S defined in Equation 7.1 to highlight the exact matches between the query and the document terms. A marking strategy is defined by two parameters:

1. **Marker-Token type**, we introduce two types of marker tokens, namely: *Simple Markers* and *Precise Markers*.
2. **Marking level**, we investigate two levels for marking: *Document Marking* and *Pair Marking*.

Table 7.2 illustrates all the four marking strategies that can be defined using the two marker-token types at the two different marking levels.

3.2.1 *Marker-Token Type*

We propose two types of marker tokens to investigate whether distinguishing query terms is important or not to the final effectiveness of the model.

SIMPLE MARKERS. A simple unique marker (#) is used to highlight all query terms without distinction. Considering a query $q = \{q_1, \ldots, q_n\}$, whose terms q_r and q_s, with $1 < r < s < n$, occur in the document, i.e., have exact matches, we obtain the new marked query segment $\tilde{q}$ by adding marker tokens before and after the tokens as follows:

$$\tilde{q} = \{q_1, \ldots, \#q_r\#, \ldots, \#q_s\#, \ldots, q_n\} \tag{7.2}$$

PRECISE MARKERS. Newly introduced pairs of opening and closing tokens $[e_k]$ and $[/e_k]$, respectively, where $k \in \{1, ..., n\}$ identifies query terms. This type of markers was first proposed in our previous work [21] for passage ranking. With these precise markers, each unique query-term q_k is associated with a unique pair of marker tokens $[e_k]$ and $[/e_k]$ which identifies it and its occurrences in the document. If a term is repeated in the query, all occurrences of this query term will be marked using the same identifier, i.e, that of the first occurrence. Considering the query q described in the previous paragraph with simple markers, it would now be marked as follows:

$$\tilde{q} = \{q_1, \ldots, [e_r]q_r[/e_r], \ldots, [e_s]q_s[/e_s], \ldots, q_n\} \tag{7.3}$$

Table 7.2 – Example of the proposed marking strategies applied to the query **q**: "causes of left ventricular hypertrophy", and the document **d**: "Left ventricular hypertrophy can occur when some factor ..."

Marker	Level	Strategy	Marked Input Sequence
Simple	Document	Sim-Doc	**q**: causes of left ventricular hypertrophy $\tilde{\mathbf{d}}$: #Left# #ventricular# #hypertrophy# can occur...
	Pair	Sim-Pair	$\tilde{\mathbf{q}}$: causes of #left# #ventricular# #hypertrophy# $\tilde{\mathbf{d}}$: #Left# #ventricular# #hypertrophy# can occur...
Precise	Document	Pre-Doc	**q**: causes of left ventricular hypertrophy $\tilde{\mathbf{d}}$: $[e_2]$Left$[/e_2]$ $[e_3]$ventricular$[/e_3]$ $[e_4]$hypertrophy$[/e_4]$ can occur...
	Pair	Pre-Pair	$\tilde{\mathbf{q}}$: causes of $[e_2]$left$[/e_2]$ $[e_3]$ventricular$[/e_3]$ $[e_4]$hypertrophy$[/e_4]$ $\tilde{\mathbf{d}}$: $[e_2]$Left$[/e_2]$ $[e_3]$ventricular$[/e_3]$ $[e_4]$hypertrophy$[/e_4]$ can occur...

3.2.2 *Marking Level*

In order to better understand whether it is relevant to mark both the query and the document segments or the document segment only in S, we investigate two marking levels: Document and Pair marking. In the former, the occurrences of query terms in the document are marked in the document segment while in the later, the exact matching terms are marked in both the document and query segments as shown in Table 7.2.

DOCUMENT MARKING. It only augments the document segment d with marker tokens indicating the start and the end of each query-term occurrences in the document. Considering a query $q = \{q_1, \ldots, q_n\}$ and a document $d = \{d_1, \ldots, d_m\}$, if $\{d_i, d_j\}$ are occurrences of the query term q_r, and d_l is the only occurrence of q_s in d with $1 < r < s < n$, and $1 < i < j < l < m$, then the augmented query and document sequences $\tilde{q}$ and $\tilde{d}$, respectively, are as follows when using the simple markers:

$$\tilde{q} = \{q_1, \ldots, q_r, \ldots, q_s, \ldots, q_n\}$$
$$\tilde{d} = \{d_1, \ldots, \#d_i\#, \ldots, \#d_j\#, \ldots, \#d_l\#, \ldots, d_m\}$$

and as follows when using the precise markers:

$$\tilde{q} = \{q_1, \ldots, q_r, \ldots, q_s, \ldots, q_n\}$$
$$\tilde{d} = \{d_1, \ldots, [e_r]d_i[/e_r], \ldots, [e_r]d_j[/e_r], \ldots, [e_s]d_l[/e_s], \ldots, d_m\}$$

PAIR MARKING. It augments both the query and document sequences with marker tokens indicating the start and the end of each exact term matches between the query and the document. In our experiments, a query term with no occurrences in the document is not marked. Considering the same example as in the *Document marking* level, the augmented query and document sequences $\tilde{q}$ and $\tilde{d}$, respectively, are as follows when using the simple markers:

$$\tilde{q} = \{q_1, \ldots, \#q_r\#, \ldots, \#q_s\#, \ldots, q_n\}$$
$$\tilde{d} = \{d_1, \ldots, \#d_i\#, \ldots, \#d_j\#, \ldots, \#d_l\#, \ldots, d_m\}$$

and as follows when using the precise markers:

$$\tilde{q} = \{q_1, \ldots, [e_r]q_r[/e_r], \ldots, [e_s]q_s[/e_s], \ldots, q_n\}$$
$$\tilde{d} = \{d_1, \ldots, [e_r]d_i[/e_r], \ldots, [e_r]d_j[/e_r], \ldots, [e_s]d_l[/e_s], \ldots, d_m\}$$

4 *Methodology and Experimental setup*

This section describes the experimental setup used for studying the effectiveness of our approach for ad hoc ranking. Our objectives through this experimental evaluation are as follows:

Table 7.3 – Benchmarks statistics. The MS MARCO document dataset has 43 judged topics in DL 2019 and 45 judged topics in DL 2020

Benchmark	# Judged Topics	# Documents	# Words per Document
Robusto4	249	0.5*M*	0.470*K*
GOV2	149	25*M*	0.835*K*
MS MARCO Document	43/45	3.2*M*	1.123*K*

1. Evaluate the effectiveness of our proposed exact match marking strategies with a BERT core on in-domain data, as well as robustness to out-of-domain data in a zero-shot setting;
2. Study how to improve the domain-transfer capabilities of our approach using score interpolation with a BoW sparse model;
3. Investigate the contribution of additional fine-tuning on limited target-domain data in a multi-phase fine-tuning setting, and how our exact match marking contributes in each phase;
4. Study the impact of exact match marking using the more effective ELECTRA core model;
5. Investigate the contextualized representations of the marker tokens and how they can be leveraged for relevance prediction;
6. Explore the use of marker tokens for implicit query expansion.

4.1 *Experimental Setup*

4.1.1 *Test Collections*

In order to achieve the aforementioned objectives, we use the following standard TREC ad hoc benchmarks involving full-length documents:

— Deep Learning Document Ranking (2019-2020) [45, 44]
— Robusto4 [1]
— GOV2 [2]

These test collections are standard in the IR literature, and are widely used for evaluating BERT reranking models [2, 51, 122], facilitating, hence, comparisons with state-of-the-art reranking models. Table 7.3 resumes statistics of these test collections.

Though we chose to evaluate our approach on traditional full-length document collections – which are common for evaluating BERT-based reranking models – we discuss results on MS MARCO passage collection and TREC DL passage ranking collections in Appendix A.1.

1. https://trec.nist.gov/data/robust/04.guidelines.html
2. http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm

Table 7.4 – Example of Robust04 search topic: Topic 302

Title	Poliomyelitis and Post-Polio
Description	Is the disease of Poliomyelitis (polio) under control in the world?
Narrative	Relevant documents should contain data or outbreaks of the polio disease (large or small scale), medical protection against the disease, reports on what has been labeled as "post-polio" problems. Of interest would be location of the cases, how severe, as well as what is being done in the "post-polio" area.

TREC DEEP LEARNING DOCUMENT RANKING A benchmark for web search issued from the TREC Deep Learning (DL) 2019-2020 tracks [45, 44]. The dataset contains more than $3M$ documents composed of three fields: title, URL and body. Dense NIST judgments are provided for 43 and 45 topics for DL 2019 and 2020, respectively.

ROBUST04 A news wire collection comprising $500K$ documents (TREC Disks 4 and 5) and 249 judged topics. Each topic is composed of three fields: The "title" is a short keyword query, the "description" is a longer well-formed natural language sentence that describes the information need and the "narrative" is a paragraph that provides guidance for relevance assessment. Table 7.4 provides an example of a TREC Robust04 topic.

GOV2 A Web collection crawled from government Websites in early 2004 comprising $25M$ documents and only 149 topics in the same format as Robust04 topics with title, description and narrative. Documents in the GOV2 corpus are on average much longer than those in the Robust04 corpus; see Table 7.3.

4.1.2 *Fine-tuning*

We use the base version (12 layers, 768 hidden size, 12 attention heads, and a total of $110M$ parameters) of BERT due to hardware limitations. We fine-tune both a vanilla monoBERT baseline and our augmented models, with the different marking strategies, on the large publicly released MS MARCO passage dataset [9]. We use a batch-size of 128 query-document pairs, and the maximum sequence length supported by BERT 512 (128 *sequences* $\times$ 512 *tokens* $=$ 65,536 *tokens*/*batch*) for $100k$ iterations (batches) on a free Google Colab TPU [3]. We use Adam optimizer [113] with an initial learning

3. https://colab.research.google.com

rate of $3e-6$ with linear decay, and warmup over the first $1,000$ iterations. The drop out rate is set to 0.1 for all our experiments.

We use the open source implementation of BERT by Hugging Face transformers [247]. It is important to note that fine-tuning an augmented model with a marking strategy does not add a computational cost compared to the vanilla monoBERT baseline, since marking is performed during preprocessing.

4.1.3 *Inference*

We use a *retrieve-than-rerank* pipeline comprised of a BoW retriever followed by our monoBERT rerankers. We use the BM25 implementation from off-the-shelf Anserini open-source IR toolkit [255] to retrieve an initial candidate list of top-1000 documents per query. These candidates are then reranked by our monoBERT models to produce the final document rankings. We additionally consider RM3 query expansion to improve the initial retrieval in scenarios where it has substantial impact –we specify in Section 7.5, the first-stage retriever used for each experimental scenario.

The maximum 512-token limitation of the BERT model prevents from directly applying our models to long documents, as discussed in Section 4.3. Following the strategy proposed by Dai and Callan [51], we split each document into overlapping passages that can be handled individually by BERT. For Robust04 and GOV2, passages are generated using a sliding window of 150 words and a stride of 75 words. As a trade-off between latency and effectiveness, we only consider a maximum of 30 passages per document. The first and last passages are always picked while the remaining 28 are randomly chosen. Inference is conducted over each passage individually using the reranking model, and the best scoring passage is taken as a proxy for the Document-level relevance (i.e., maxP [51]).

For the queries we consider both the topic titles, that are preferred by most pre-BERT models including BM25, and the descriptions that are more similar to MS MARCO's natural language questions used for fine-tuning our monoBERT models.

For TREC DL Document ranking evaluation, we follow the splitting strategy used in Yan et al. [253], and split each document into overlapping passages with the same maximum length of 384 and a stride of 192. If available, the document title is additionally prepended to the beginning of every passage. Finally, we use the best scoring passage as proxy for the whole document relevance.

4.1.4 *Evaluation Metrics*

We compare the proposed approach to the baselines by adopting the TREC protocol. For this purpose, we use the official metrics used in the

context of the TREC DL document ranking tasks, namely: nDCG@10 and MAP@100. And report nDCG@20 and P@20 for Robust04 and GOV2 to allow straightforward comparisons with previously reported results in the literature.

All performance measures are obtained by evaluating the document rankings (retrieval runs) using the official `trec_eval`[4] tool from TREC. It is the tool used by the TREC community to evaluate the performance of track submissions.

We also conduct statistical significance tests, *t-tests* also called *Student t-tests*. Significant improvements are indicated with the symbol † with $p < 0.05$, and the symbol ‡ for $p < 0.01$.

4.2 *Baselines*

We compare our approach against diverse baselines including: Traditional non-neural approaches also known as lexical retrieval methods, BERT-based sparse retrieval approaches, dense retrieval models (bi-encoders), and strong reranking models (cross-encoders).

4.2.1 *Lexical Retrieval baselines*

— BM25, we use the Anserini [255] implementation with default parameters. For description queries, we set $k_1 = 0.9$ for Robust04 and $k_1 = 2.0$ for GOV2 and $b = 0.6$ for both datasets. This unsupervised model serves both as a baseline and as the first stage retriever in all our experiments.

— BM25+RM3, a query expansion model based on RM3 [118] considered as a strong non-neural baseline. We use the Anserini [255] implementation with the default parameters. For description queries, we use 20 expansion terms following [122].

4.2.2 *Sparse Retrieval baselines*

— DeepCT [53], we report the authors results on Robust04 and GOV2 obtained using the BOW+DeepCT-Query model [50], and use their published re-weighted MS MARCO documents[5] produces by the HDCT model [52] in combination with Anserini's BM25 with default parameters for TREC DL 2019 and 2020 evaluations.

— DocT5Query [175], following the original paper setup, we generate 40 expansion queries per document and use Anserini's BM25 with default

4. http://trec.nist.gov/trec_eval/

5. http://boston.lti.cs.cmu.edu/appendices/TheWebConf2020-Zhuyun-Dai/rankings/

parameters. Due to the large size of the GOV2 collection (see Table 7.3) and the high computational cost of DocT5query we do not report results on this collection.

4.2.3 *Dense Retrieval baselines*

— DPR [110], we use DPR as a retriever with the open source implementation from the transformers library [247] and the publicly released DPR checkpoints for Query [6] and Context [7] encoders.

— ANCE [251], we use ANCE as a retriever and use the Sentence Transformers library [199] with the publicly released checkpoint [8].

— ColBERT [112], we use ColBERT as a dense retriever using the authors released code: after encoding the whole collection, we use the top-1000 documents retrieved using ANN with faiss [107] and rerank them using ColBERT late-interaction mechanism. Considering the size of the GOV2 collection (25M documents), and the important space footprint of ColBERT indexes [9], we could not produce results on GOV2.

4.2.4 *Reranking baselines*

We use the following reranking baselines on the Robust04 and GOV2 benchmarks, as previous work report results on these standard IR collections. For comparisons on the TREC DL document ranking tasks, we instead report the best TREC run results from each track.

— Vanilla baseline, the vanilla monoBERT model is our main baseline since it represents the core model we augment with explicit exact match cues in our proposed models. The vanilla baseline as well as our models share the same configuration and evaluation setup making it suitable for evaluating the impact of exact match marking.

— Birch (MS) and Birch (MS-MB) [2], the notation in parentheses indicate the fine-tuning dataset(s): Ms for MS MARCO and MS-MB refers to the model fine-tuned first on MS MARCO and then further fine-tuned on Microblog (MB) data. Since the original paper does not report results on GOV2, we use the results reported by Li et al. [122], in which BM25 is used instead of BM25+RM3 as the first-stage retriever.

6. https://huggingface.co/sentence-transformers/facebook-dpr-question_encoder-multiset-base
7. https://huggingface.co/sentence-transformers/facebook-dpr-ctx_encoder-multiset-base
8. https://huggingface.co/sentence-transformers/msmarco-roberta-base-ance-firstp
9. With less than 4M documents, the size of the MS MARCO Document index was already as big as 200GB.

— BERT-MaxP (MS) [51], we report the results obtained with the re-implementation by Li et al. [122] where the results are improved using a BERT model fine-tuned on MS MARCO rather than Bing search log.

— Parade [122], we report results obtained using both BERT and ELECTRA cores from the paper.

— Parade-v2 [123], we report the results from the latest version of the PARADE paper, where the model training was enhanced and uses only the ELECTRA core. This variant uses BM25+RM3 as the first-stage retriever and the reranking threshold is increased from the previous 100 to 1000, leading to much better performance.

— monoT5 [174], with $3B$ parameters detains the state-of-the-art across many ad hoc benchmarks like Robust04. We report the original results from the paper.

5 *Results and Analysis*

We present, in the following, the results of the experimental evaluation of our exact match marking proposition. First, we investigate the effectiveness of our proposed exact match marking strategies with a BERT core on in-domain data, i.e., on the TREC DL document ranking 2019-2020 benchmarks, and the robustness to out-of-domain collections, i.e., Robust04 and GOV2. Then, we study how to improve the domain-transfer capabilities of our models by first using score interpolation with a BoW model, and second, by additional fine-tuning on limited target-domain data in a multi-phase fine-tuning setting, and isolate the exact match marking contributions in each phase. Finally, we verify the contribution of our exact match marking on the more effective ELECTRA model, and compare our best configurations to the state-of-the-art baselines presented previously.

5.1 *Contribution of exact match marking*

We evaluate, in this section, the contribution of our proposed exact match marking strategies by comparison to the vanilla baseline to answer our first research question:

RQ1. *Is exact match marking beneficial for reranking with BERT?*

We consider results on in-domain data with the MS MARCO Document dataset used in TREC DL document ranking tracks, as well as out-of-distribution data, namely: Robust04 and GOV2 in a zero-shot transfer setting.

Table 7.5 – Reranking effectiveness on the TREC DL 2019 and DL 2020 Document ranking tasks. The best performance of our proposed models is highlighted in **bold**, and baseline's results are underlined when overall best. Significant improvements over the vanilla baseline with $p < 0.05$ are indicated with †. Change rates over the vanilla baseline are reported for each metric (%)

TREC DL Doc	DL 2019				DL 2020			
Model	nDCG@10		MAP@100		nDCG@10		MAP@100	
BM25	0.5176	–	0.2434	–	0.5286	–	0.3793	–
BM25+RM3	0.5169	–	0.2772	–	0.5248	–	0.4006	–
Vanilla $_{\text{BERT}}$	0.6726	–	0.3006	–	0.6340	–	0.4523	–
Sim-Doc $_{\text{BERT}}$	0.6858	▲2.0%	0.3038	▲1.1%	0.6340	▲0.0%	0.4414	▽2.4%
Sim-Pair $_{\text{BERT}}$	0.6798	▲1.1%	0.3057	▲1.7%	0.6495	▲2.4%	0.4505	▽0.4%
Pre-Doc $_{\text{BERT}}$	0.6777	▲0.8%	**0.3061**	▲1.8%	0.6368	▲0.4%	**0.4513**	▽0.2%
Pre-Pair $_{\text{BERT}}$	**0.7025**†	▲4.4%	0.3018	▲1.8%	**0.6498**	▲2.5%	0.4497	▽0.6%

5.1.1 *In-domain effectiveness*

We re-rank the initial list of candidate documents retrieved by BM25 with RM3 query expansion, using all our models and the vanilla baseline. We report the performance on the TREC DL 2019 and 2020 test sets [10] in Table 7.5, in terms of the official task evaluation metrics: nDCG@10 and MAP@100.

Comparison with baselines. Compared to BM25 and the first-stage retriever (BM25+RM3), all BERT-based models perform significantly better.

Adding exact match marking regardless of the marking strategy, leads to better or at least the same performance as the vanilla baseline. The Pre-Pair $_{\text{BERT}}$ model achieves the overall best performance on DL 2019 test topics (+4.4% gain in nDCG@10), and on DL 2020 (+2.5% gain in nDCG@10) along with Sim-Pair $_{\text{BERT}}$ (+2.4% gain in nDCG@10).

Impact of the marker type and marking level on the performance. On TREC DL 2019, using the pair marking strategy brings substantial gains in performance when used in combination with the precise marker type: Pre-Pair $_{\text{BERT}}$ achieves +3.7% relative gain over the Pre-Doc $_{\text{BERT}}$ model in terms of nDCG@10. While it leads to a drop in performance when combined with the simple marker, Sim-Pair $_{\text{BERT}}$ has a relative loss of −0.9% compared to Sim-Doc $_{\text{BERT}}$ in terms of nDCG@10. Interestingly, on TREC DL 2020 using the pair marking level has the same impact regardless of the marker type.

Marking both the query and document segments seems to be more beneficial considering results on both test collections. Using the precise marker type brings further gains in performance on DL 2019.

10. Equivalent results on the passage ranking collections are reported in Appendix A.1

Table 7.6 – Reranking effectiveness in the zero-shot transfer setting of the different models on Robust04 and GOV2 collections. The best performance of our proposed models is highlighted in **bold**, and baseline's results are underlined when overall best. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively. For each measure, the improvement rate over the vanilla baseline is given (%)

Robust04	Title run				Description run			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4058	–	0.3345	–
BM25+RM3	0.4407	–	0.3821	–	0.4255	–	0.3661	–
Vanilla BERT	0.4652	–	0.4046	–	0.4510	–	0.3851	–
Sim-Doc BERT	0.4447	▽4.4%	0.3831*	▽5.3%	0.4166	▽7.6%	0.3510	▽8.9%
Sim-Pair BERT	**0.4773**	▲2.6%	**0.4155**	▲2.7%	**0.4931**‡	▲9.3%	**0.4169**‡	▲8.3%
Pre-Doc BERT	0.4767	▲2.5%	0.4084	▲0.9%	0.4789‡	▲6.2%	0.4026‡	▲4.5%
Pre-Pair BERT	0.4654	▲0.0%	0.4024	▽0.5%	0.4795‡	▲6.3%	0.4034‡	▲4.8%

GOV2	Title run				Description run			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	–	0.5362	–	0.4264	–	0.4705	–
BM25+RM3	0.4851	–	0.5634	–	0.4212	–	0.4966	–
Vanilla BERT	0.4533	–	0.5272	–	0.4696	–	0.5248	–
Sim-Doc BERT	**0.4588**	▲1.2%	**0.5349**	▲1.5%	0.4686	▽0.2%	0.5262	▲0.3%
Sim-Pair BERT	0.4468	▽1.4%	0.5134	▽2.6%	0.4687	▽0.2%	0.5326	▲1.5%
Pre-Doc BERT	0.4485	▽1.1%	0.5121	▽2.9%	**0.4768**	▲1.5%	**0.5315**	▲1.3%
Pre-Pair BERT	0.4515	▽0.4%	0.5238	▽0.6%	0.4752	▲1.2%	0.5285	▲0.7%

5.1.2 *Out-of-domain effectiveness*

In this evaluation, we investigate the generalizability of our approach to out-of-domain collections. We use the fine-tuned models on MS MARCO passages for evaluation on Robust04 and Gov2 test collections. We do not train the models on these test collections, we use all their queries and relevance judgements as a held-out test set. Thus, this evaluation is an instance of a zero-shot transfer setting.

Table 7.6 shows the reranking effectiveness of our different models and baselines on the top-1000 candidate documents retrieved by BM25 from Robust04 and GOV2 collections using both the title and description fields of their TREC topics. We report results using the commonly used nDCG@20 and P@20 metrics to enable direct comparisons with previous work on these collections.

Comparison with baselines. All BERT-based models achieve substantially better performance on both collections compared to the traditional non-neural baselines, at the only exception of GOV2 titles. We observe a

Table 7.7 – Recall of BM25 on Robsuto4 and GOV2 collections on both title and description queries

Collection	Title	Description
Robusto4	0.6989	0.6519
GOV2	0.7106	0.6024

discrepancy in the impact of the exact match marking on GOV2 compared to Robusto4. While all our models, except Sim-Doc BERT, significantly outperform the vanilla baseline on Robusto4 descriptions or at least achieve similar performance on titles, our models have no significant impact on GOV2. Nevertheless, in no case a marking-based model leads to a significant degradation of performance on GOV2. The disparity in the behavior of the models on the two benchmarks is probably due to the nature of the documents involved. While Robusto4 comprises well-written news articles, GOV2 documents are web pages that include navigation bars, advertisements, tables and discontinuous text. The zero-shot domain transfer –from the MS MARCO fine-tuned models to Robusto4 articles– seems to be more attainable than to GOV2 web pages even though MS MARCO passages were extracted from the web. We hypothesise that further fine-tuning on domain-specific data may be required to learn better domain-specific text representations. We investigate this in-domain adaptation in section 5.3.

Impact of the marker type and marking level on the performance. On Robusto4, marking both the query and the document –models based on pair marking– has more impact on the simple marker than the precise marker. Sim-Pair BERT achieves a relative gain of +18%, and +7.3% in terms of nDCG@20 over Sim-Doc BERT on description and title queries, respectively. At the same time, the pair marking level has a lower impact on models using the precise markers (Pre-Doc BERT and Pre-Pair BERT) especially on descriptions. On the other hand, results on the GOV2 collection are quite mitigated.

Marking both the query and the document segments with a simple marker (#) appears to be the best setting, Sim-Pair BERT has the best ranking accuracy among the four tested strategies, with clear margins on the Robusto4 collection especially on descriptions. We, thus, choose to continue our analysis using the Sim-Pair BERT strategy, nevertheless, the full results using all the marking strategies can be found in Appendix 2.2.

Title vs. description queries. Since we are in a reranking configuration, it is important to note that the first stage retriever BM25, as most pre-BERT ranking models, prefers short keyword queries to longer natural language descriptions [51, 174]. Table 7.7 shows the recall at rank 1,000 of BM25 for both title and description queries, where we notice a substantial difference in recall, affecting hence the quality of the candidate documents presented to

Table 7.8 – Reranking effectiveness in the zero-shot transfer setting of the different models on Robust04 and GOV2 collections using the hybrid pipeline. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively. For each measure, the improvement rate over the vanilla baseline is given (%)

	Robust04				GOV2			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4774	–	0.5362	–
BM25+RM3	0.4407	–	0.3821	–	0.4851	–	0.5634	–
Vanilla $_{\text{BERT}}$	0.4845	–	0.4147	–	0.4937	–	0.5611	–
Sim-Pair $_{\text{BERT}}$	**0.5239**‡	▲8.1%	**0.4446**‡	▲7.2%	**0.4991**	▲1.1%	**0.5695**	▲1.5%

the reranking models. However, despite this disadvantageous initial retrieval, the reranking models manage to reduce the gap between title and description runs. The improvement rate over BM25 is much higher for description queries compared to title queries on both collections especially on GOV2 where vanilla $_{\text{BERT}}$ has a change rate of -5% nDCG@20 over BM25, while it achieves over $+10\%$ gain in nDCG@20 on descriptions. This means that BERT is able to take advantage of richer natural language descriptions of the information need as opposed to the BoW retriever. This finding appears to be robust as previous works such as Dai and Callan [51] confirm the higher effectiveness of description queries over title queries as well. Further adding exact match marking in Sim-Pair $_{\text{BERT}}$ does not change this preference, as it improves the search accuracy of the description runs more effectively than the title runs. As a matter of fact, the overall performance reported for our model using descriptions clearly surpasses that obtained using titles by $+4.1\%$ on average, despite the lower recall in the initial retrieval.

Impact of the initial stage retriever. Considering that the first stage retriever, BM25, has higher recall on title queries, and that the BERT-based reranking models prefer description queries, we propose a hybrid reranking pipeline where BM25 uses title queries in the initial retrieval, while reranking models use the richer description queries. This hybrid pipeline allow us to obtain a higher recall in the initial retrieval which means more relevant documents in the candidates pool. At the same time, description queries in the reranking stage allow the BERT-based models to fulfill their potential.

In practice, this pipeline remains realistic as natural language queries can be generated from standard keyword queries [182]. This hybrid approach is also adopted in the recent state-of-the-art reranking model monoT5 [174].

Table 7.8 shows the results obtained using the hybrid reranking pipeline on both test collections. Unsurprisingly, better candidate documents for reranking with descriptions yields overall best ranking effectiveness. The vanilla

BERT model achieves an improvement rate of +14% over BM25 on Robust04 and +3.4% on GOV2 in terms of nDCG@20 (we recall that BM25 results are obtained using titles). Adding exact match marking in the hybrid reranking pipeline outperforms the vanilla baseline on both collections; significantly on Robust04 with a gain of over +8% in terms of nDCG@20.

5.1.3 *In-domain vs. out-of-domain effectiveness.*

Results on both in-domain and out-of-domain benchmarks clearly indicate that exact match marking, aside from the Sim-Doc marking strategy which underperforms the vanilla baseline on Robust04, is more beneficial than using a vanilla baseline. Using Sim-Pair (especially for out-of-domain experiments) or Pre-Pair (especially for in-domain experiments) marking strategies seems to be working best.

In the next two sections, we focus on out-of-domain effectiveness and study common techniques used in the literature to enhance the effectiveness of BERT-based models, and how our models behave in combination with these techniques. Therefore, the MS MARCO document ranking benchmark is not suitable and thus we only report results on Robust04 and GOV2 collections.

5.2 *Contribution of the first-stage retriever scores to the end-to-end effectiveness*

Our experimental design is based on a two-stage reranking architecture where our BERT-based models rerank the documents retrieved by a BoW model. However, the scores from the retriever are never considered in the final ranking of the documents. Inspired by the score interpolation method employed in Birch [2], we consider a simple linear combination of retriever's document-level scores with the passage-level evidence from the reranker, to study our second research question:

RQ2. *Do exact match scores from the first-stage retriever contribute to the end effectiveness and how exact match marking affects this contribution?*

Akkalyoncu Yilmaz et al. [2] defines the final relevance score of a document as the combination of its document-level term-matching score from the BoW retriever, and evidence contributions from the top sentences in the documents as determined by monoBERT, as discussed in Section 3.1. More formally, to determine document relevance s_f, inference is applied over each individual sentence s_i in a candidate document d, and then the top n sentence scores

Table 7.9 – Reranking effectiveness of the different models before and after interpolating BM25 scores on Robust04 and GOV2 collections. Best performance is highlighted in **bold**. For each measure, the improvement rate over the reranking performance without BM25 scores interpolation is given (%)

Robust04	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4058	–	0.3345	–	0.4240	–	0.3631	–
Vanilla BERT	0.4652	–	0.4046	–	0.4510	–	0.3851	–	0.4845	–	0.4147	–
+ BM25	0.4932	▲6.0%	0.4255	▲5.2%	0.4856	▲7.7%	0.4062	▲5.5%	0.5266	▲8.7%	0.4488	▲8.2%
Sim-Pair BERT	0.4773	–	0.4155	–	0.4931	–	0.4169	–	0.5239	–	0.4446	–
+ BM25	**0.4947**	▲3.6%	**0.4265**	▲2.6%	**0.5098**	▲3.4%	**0.4279**	▲2.6%	**0.5497**	▲4.9%	**0.4707**	▲5.9%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	–	0.5362	–	0.4264	–	0.4705	–	0.4774	–	0.5362	–
Vanilla BERT	0.4533	–	0.5272	–	0.4696	–	0.5248	–	0.4937	–	0.5611	–
+ BM25	0.5320	▲17.%	0.5987	▲13.	0.5166	▲10.%	0.5742	▲9.4%	0.5722	▲16.%	0.6383	▲14%
Sim-Pair BERT	0.4468	–	0.5134	–	0.4687	–	0.5326	–	0.4991	–	0.5695	–
+ BM25	**0.5327**	▲19.%	**0.6000**	▲17%	**0.5235**	▲12.%	**0.5893**	▲11.%	**0.5778**	▲16.%	**0.6497**	▲14.%

are combined with the original document score s_d given by the first-stage retriever as follows:

$$s_f \triangleq \alpha \cdot s_d + (1 - \alpha). \sum_{i=1}^{n} w_i \cdot s_i \tag{7.4}$$

where s_i is the i-th top scoring sentence according to monoBERT. The parameters α and w_i's are tuned via cross-validation.

For our evaluation, we apply linear interpolation to the results obtained in the zero-shot transfer setting with the best-scoring passage ($n = 1$). In other words, we use the score combination defined in Equation 7.4 on the document scores obtained by the BM25 retriever at cutoff 1,000 and their corresponding scores estimated with the best-scoring passage method by the reranking models.

Table 7.9 first shows the results of the traditional BM25 retriever alone, then the second and third sections are each dedicated to a reranker: vanilla and Sim-Pair BERT models. For both rerankers, we remind the results of the model alone obtained in the zero-shot transfer setting and then present the end-to-end effectiveness after interpolating BM25 scores (+BM25), and indicate the change rate (%) over the reranker-only effectiveness.

Impact of interpolating BM25 scores. Interpolating BM25 scores (Best Match) that are solely based on surface-level features such as TF and IDF leads to significant gains in performance in both collections with both reranking models. This indicates that BM25 document-level scores provide an additional relevance signal that the BERT-based models alone could not effectively capture. Notably, the improvement rate resulting from interpolating BM25 scores is more substantial on the GOV2 collection (+15% nDCG@20 in average) compared to Robust04 (+5.7% nDCG@20 in average). The fact that

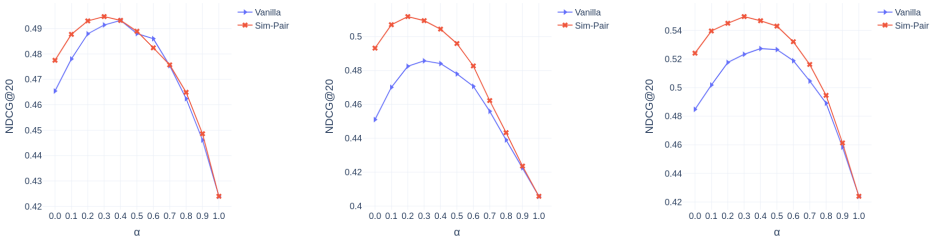
the BERT models outperform BM25 by a large margin on Robust04, while this margin is much smaller on the GOV2 can explain why BM25 scores have more incidence on the end-to-end effectiveness on GOV2 compared to Robust04.

Impact of exact match marking. Combining BM25 scores with the scores produced by both the Vanilla and Sim-Pair BERT models always leads to substantial gains in the end ranking effectiveness. On the Robust04 collections, Sim-Pair $_{\text{BERT}}$ outperforms significantly Vanilla $_{\text{BERT}}$. However, the combination with BM25 scores has more impact on the vanilla model with about about $+7.5\%$ gain in nDCG@20 over the previous Vanilla $_{\text{BERT}}$ reranker-only effectiveness, compared to only $+4\%$ gain with Sim-Pair $_{\text{BERT}}$+BM25 over Sim-Pair $_{\text{BERT}}$. As a result, BM25 scores help more with the vanilla baseline and shrinks the previous large effectiveness gap Vanilla $_{\text{BERT}}$ and Sim-Pair $_{\text{BERT}}$. Nevertheless, Sim-Pair $_{\text{BERT}}$+BM25 still outperforms the vanilla $_{\text{BERT}}$+BM25 variant on the description and hybrid runs. This suggests that Sim-Pair $_{\text{BERT}}$, by virtue of using exact match marking, requires less intervention from the retriever's scores, which are based on exact match signals.

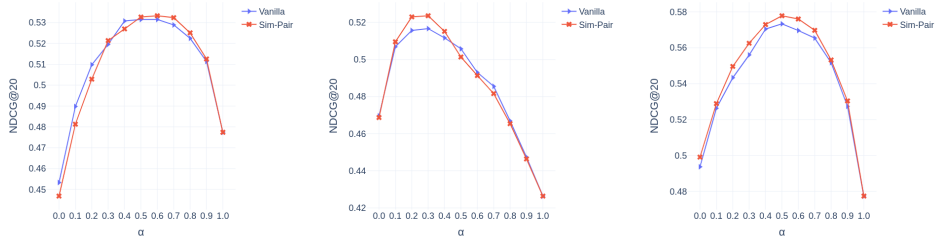
On the other hand, Sim-Pair $_{\text{BERT}}$ and Vanilla $_{\text{BERT}}$ perform similarly on the GOV2 collection. The contribution from BM25 scores is also about the same with both models, and Sim-Pair $_{\text{BERT}}$+BM25 and Vanilla $_{\text{BERT}}$+BM25 have comparable effectiveness.

Contribution of BM25 scores. The contribution of BM25 scores is controlled by the parameter α in Equation 7.4, which we tuned via 5-fold in-collection cross validation. In all scenarios, the weight put on α is non-negligible, in other words, the contribution of BM25 signals remain important, this observation was also reported for the Birch model [129]. However, we notice that the weight of α is less important when combining with the Sim-Pair $_{\text{BERT}}$ model that uses exact match marking. For Robust04 descriptions, the vanilla $_{\text{BERT}}$+BM25 baseline puts a weight of $\alpha \in \{0.3, 0.4\}$ on BM25 scores, when Sim-Pair $_{\text{BERT}}$+BM25 only consider a contribution of $\alpha = 0.2$ from BM25, while achieving substantially better performance. *This indicates that the vanilla model relies more on BM25 to complete its relevance estimation unlike the marking-based model that is able to effectively capture more relevance signals, possibly similar to BM25, and thus needing less contribution from BM25 scores.*

Figure 7.2 visualizes the end ranking accuracy measured by nDCG@20 for $\alpha \in [0, 1]$ on both Robust04 and GOV2 collections. On Robust04, we can clearly see that Sim-Pair $_{\text{BERT}}$+BM25 reaches the most effective combination with smaller contributions from BM25 scores (smaller α), while the vanilla baseline requires more intervention from BM25 and still cannot reach the performance of Sim-Pair $_{\text{BERT}}$+BM25, especially on descriptions. It is only logical that the most performing model, that outperforms BM25 by a large margin, requires less contribution from this later. Nevertheless, if we take the example of the GOV2 descriptions, despite the similar starting performance



(a) Robust04 titles (b) Robust04 descrip. (c) Robust04 hybrid



(d) GOV2 titles (e) GOV2 descrip. (f) GOV2 hybrid

Figure 7.2 – The end ranking accuracy of the vanilla $_{\text{BERT}}$ and Sim-Pair $_{\text{BERT}}$ models with BM25 scores interpolation on Robust04 and GOV2 collections. $\alpha = 0.0$ indicates the reranking model effectiveness only without BM25 scores, and $\alpha = 1.0$ means that only BM25 scores are used

at $\alpha = 0.0$ of vanilla and Sim-Pair BERT models, the gap between their performance starts getting wider at only $\alpha = 0.1$ to reach its peak at $\alpha = 0.2$.

Combining the document scores obtained in the first-stage retriever with passage-level evidence from BERT-based reranking models to determine the final relevance score of a document yields substantial gains in performance. Relevance scores based on traditional IR axioms complete the relevance signals captured by contextual pre-trained LMs such as BERT. Moreover, using our simple, yet effective, marking strategy to highlight the exact matching signals in the query-document pairs enhance BERT's own ability to estimate relevance and thus, requires less contribution from BM25 to achieve the best performance.

5.3 *Multi-Phase Fine-Tuning*

For both Robust04 and GOV2 benchmarks, we only relied on labeled data from the large MS MARCO passage collections to fine-tune our BERT models. This fine-tuning aims at providing the model with general notions

Table 7.10 – Reranking effectiveness in the multi-phase vs. zero-shot transfer setting for the Sim-Pair and vanilla models on Robust04 and GOV2 collections. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively for the same setting. Change rate over the vanilla baseline in the same setting are reported for each metric (%)

Robust04	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4058	–	0.3345	–	0.4240	–	0.3631	–
BM25+RM3	0.4407	–	0.3821	–	0.4255	–	0.3661	–	0.4407	–	0.3821	–
Zero-shot transfer												
Vanilla $_{\text{BERT}}$	0.4764	–	0.4096	–	0.4611	–	0.3867	–	0.4989	–	0.4245	–
Sim-Pair $_{\text{BERT}}$	0.4763	▽0.0%	0.4129	▲0.8%	0.4923‡	▲6.8%	0.4084‡	▲5.6%	0.5273‡	▲5.7%	0.4434‡	▲4.5%
Multi-phase												
Vanilla $_{\text{BERT}}$	0.4995	–	0.4275	–	0.5368	–	0.4492	–	0.5546	–	0.4715	–
Sim-Pair $_{\text{BERT}}$	**0.5058**	▲1.3%	**0.4371**	▲2.2%	**0.5479**†	▲2.1%	**0.4574**†	▲1.8%	**0.5701**‡	▲2.8%	**0.4815**‡	▲2.1%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	–	0.5362	–	0.4264	–	0.4705	–	0.4774	–	0.5362	–
BM25+RM3	0.4851	–	0.5634	–	0.4212	–	0.4966	–	0.4851	–	0.5634	–
Zero-shot transfer												
Vanilla $_{\text{BERT}}$	0.5098	–	0.5916	–	0.4928	–	0.5560	–	0.5510	–	0.6312	–
Sim-Pair $_{\text{BERT}}$	0.5181	▲1.6%	0.5990	▲1.3%	0.4904	▽0.5%	0.5597	▲0.7%	0.5531	▲0.4%	0.6346	▲0.5%
Multi-phase												
Vanilla $_{\text{BERT}}$	0.5476	–	0.6302	–	0.5175	–	0.5772	–	0.5909	–	0.6604	–
Sim-Pair $_{\text{BERT}}$	**0.5743**‡	▲4.9%	**0.6540**‡	▲3.8%	**0.5406**‡	▲4.5%	**0.6084**‡	▲5.4%	**0.5998**	▲1.5%	**0.6758**	▲2.3%

of relevance matching. However, transferring these relevance patterns to the target corpus may, in some cases, be ineffective as we have seen with GOV2 results presented in Section 5.1.2. To overcome this domain-transfer limitation, we use additional fine-tuning on labelled data drawn from the same distribution as the target task. In other words, we use a multi-phase fine-tuning process (Section 2.3.2).

Once the models are fine-tuned on the MS MARCO passage dataset following the training setting described in section 4.1.2, we further fine-tune them on the target task using 5-fold cross validation for both Robust04 and GOV2 collections. We use the folds defined by Yang et al. [256] for Robust04 and the 5-folds configuration adopted by Li et al. [122] for GOV2.

Following prior work by Dai and Callan [51], we consider the top-1000 documents retrieved by BM25 for queries in the training folds for extracting in-domain training instances. Each document is segmented into passages using a sliding window of 150 words with a stride of 75 words similarly to the inference setting described in Section 4.1.3. These passages are further sub-sampled to avoid catastrophic forgetting. Aside from the first passage, passages in a document are randomly preserved with a probability of 0.1. Since passage-level judgements are not available, passages from a relevant document according to the ground-truth (TREC relevance judgements) are considered to be relevant, while passages issued from the other remaining documents are treated as non relevant. We use a cross entropy loss and fine-

tune the models for a single epoch with a batch size of 32 training instances comprising a query-passage pair. We use the Adam optimizer with a learning rate of $1e-5$ with warm up over the first 10% of the total training steps.

For queries in the left-out test fold, we set the rerank threshold to 100 as a trade-off between latency and effectiveness, and otherwise follow the same inference setting presented in Section 4.1.3. We report the average performance across all test folds measured in terms of P@20 and nDCG@20. In this setting, our vanilla baseline corresponds to a BERT-MaxP model [51] initialized from a fine-tuned monoBERT on MS MARCO passages, instead of Google's BERT pre-trained checkpoint without any prior fine-tuning on the text ranking task.

Table 7.10 reports the reranking effectiveness obtained using the multi-phase fine-tuning setting compared to the zero-shot transfer setting for both Robust04 and GOV2 collections. For a fair comparison, we report the results obtained for reranking the top-100 documents retrieved by BM25 for the zero-shot setting, instead of the previously reported results with a rerank threshold of 1000.

Impact of multi-stage fine-tuning. Thanks to the additional in-domain fine-tuning on the target collection, the performance on both collections improves regardless of the topic field. We notice in this setting that Sim-Pair BERT is able to achieve significant gains over the vanilla baseline on the GOV2 collection on title and description queries, confirming our hypothesis that the zero-shot domain transfer from MS MARCO was not sufficient for this collection.

On the Robust04 collection, the impact of exact match marking follows the same tendency as in the zero-shot setting. That is, Sim-Pair BERT achieves substantial gains on both description and hybrid runs, and achieves comparable effectiveness with the vanilla baseline on the title run. However, these gains are less pronounced compared to the zero-shot setting. A possible reason might be that the BERT-based reranker, owing to the additional target-task fine-tuning, has gained more specialized "understanding" of the ranking task on the Robust04 collection, and thus the exact match signals have less impact. This is similar to the in-domain evaluation results on TREC DL document ranking 2019-2020 tasks in Section 5.1.1.

Title vs. Description queries. In the multi-phase fine-tuning setting, the BERT rerankers are able to achieve better performance on Robust04 descriptions compared to the titles with +7.5% and +8.3% gains in nDCG@20 for the vanilla and Sim-Pair models respectively, despite the lower retrieval effectiveness of BM25 on descriptions compared to titles. On the other hand, the gap in BM25 effectiveness between descriptions and titles is more important on the GOV2 collection. Even though the BERT-based rerankers reduce this gap to −5.5% and −5.9% for the vanilla and Sim-Pair models, respectively, it is not enough to reverse the tendency. The end effectiveness

Table 7.11 – Reranking effectiveness with exact match marking ablation at different phases of the multi-phase fine-tuning setting of Sim-Pair BERT on Robust04 and GOV2 collections. MS refers to the MS MARCO fine-tuning phase and ID to the in-domain fine-tuning. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively for the same setting. Change rates over the vanilla baseline are reported for each metric (%)

Robust04	Marking		Title run				Description run				Hybrid run			
Run	MS	ID	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
Vanilla BERT	-	-	0.4995	–	0.4275	–	0.5368	–	0.4492	–	0.5546	–	0.4715	–
Sim-Pair BERT	✓	✓	**0.5058**	▲1.3%	**0.4371**	▲2.2%	0.5479†	▲2.1%	0.4574†	▲1.8%	**0.5701**‡	▲2.8%	**0.4815**‡	▲2.1%
A	✓	-	0.4978	▽0.3%	0.4281	▲0.1%	**0.5521**‡	▲2.9%	**0.4592**†	▲2.2%	0.5678‡	▲2.4%	0.4811†	▲2.0%
B	-	✓	0.4896*	▽2.0%	0.4239	▽0.8%	0.5344	▽0.4%	0.4504	▲0.3%	0.5500	▽0.8%	0.4665	▽1.0%

GOV2	Marking		Title run				Description run				Hybrid run			
Run	MS	ID	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
Vanilla BERT	-	-	0.5476	–	0.6302	–	0.5175	–	0.5772	–	0.5909	–	0.6604	–
Sim-Pair BERT	✓	✓	**0.5743**‡	▲4.9%	**0.6540**‡	▲3.8%	0.5406‡	▲4.5%	0.6084‡	▲5.4%	0.5998	▲1.5%	**0.6758**	▲2.3%
A	✓	-	0.5665†	▲3.5%	0.6430	▲2.0%	**0.5509**‡	▲6.5%	**0.6161**‡	▲6.7%	**0.6027**	▲2.0%	0.6728	▲1.9%
B	-	✓	0.5503	▲0.5%	0.6312	▲0.2%	0.5218	▲0.8%	0.5785	▲0.2%	0.5761	▽2.5%	0.6517	▽1.3%

on this collection is, thus, higher on titles than descriptions as observed in previous state-of-the-art models such as BERT-MaxP [51] or Parade [122] (see results in section 5.5). Still, the hybrid pipeline outperforms both title and description runs on both collections. The reranking accuracy achieved by the hybrid runs are, to our knowledge, the highest reported results using a BERT-based model on both collections, at the time this dissertation was written.

Phase-wise marking. Previous results of the Sim-Pair BERT model presented in Table 7.10 in the multi-phase setting are obtained by applying the Sim-Pair exact match marking strategy through out the two fine-tuning phases. While the first phase fine-tuning focuses on learning general notions of relevance from a large passage collection, the goal of adding in-domain fine-tuning is to learn directly from labelled data with the same distribution as the target task. It is important to determine on which of the two phases, the marking strategy is more beneficial and at which phase it can be omitted. To this aim, we conduct an ablation study on the Sim-Pair BERT model. Table 7.11 shows the results of the marking-strategy ablation on Robust04 and GOV2 collections using the different topic fields. With these results, we can now discuss the following research question:

RQ3. *At which phase the exact match marking is the most beneficial in a multi-phase fine-tuning setting?*

MS marking (labelled run A in Table 7.11), uses exact match marking in the MS MARCO (MS) fine-tuning phase only, and then uses the original data without further marking for the in-domain (ID) fine-tuning phase. We can see from Table 7.11, that using the marking strategy in the first general fine-tuning phase is sufficient to outperform the vanilla baseline or at least

perform similarly for Robust04 titles. In other words, initializing monoBERT with the pre-fine-tuned weights on marked MS MARCO passage training instances is preferable to the model weights optimized on the original non-marked training instances. MS marking only in run A, can even even lead to improved effectiveness on the description runs for both collections, as well as a slight improvement on the GOV2 hybrid run.

ID marking (labelled run B in Table 7.11), uses the marking strategy to augment the inputs during fine-tuning phase on the in-domain data only. This means that monoBERT is pre-fine-tuned (in the first phase) on the original non-marked MS MARCO passage training instances. The results of this exact match marking ablation during the first-phase fine-tuning either has no substantial impact on the model's performance or leads to a degradation in performance. This behavior is predictable, since there is not enough in-domain data for BERT to learn useful representations of the marker tokens, and their contribution to the relevance prediction.

Using a marking strategy during the first general-purpose fine-tuning phase (MS marking) on the MS MARCO passage collection has already enough capacity to outperform the vanilla baseline without requiring additional marking during the in-domain fine-tuning phase. At the end, the fine-tuned model using the Sim-Pair marking strategy on the MS MARCO passage collection is able to use the relevance matching patterns learned using out-of-domain data, with explicit marking, for later phases even without the guidance of the explicit markers. Nevertheless, additional marking in the in-domain fine-tuning phase used in the classical Sim-Pair BERT approach is beneficial for title queries where it brings and additional gain of +1.6% and +1.4%, in terms of nDCG@20, over the MS marking only (run A) on Robust04 and GOV2, respectively.

5.4 *Impact of exact match marking on ELECTRA*

While BERT is the most famous and largely adopted pre-trained language model, additional variants such as RoBERTa [138] or ELECTRA [40] were proposed in order to improve the model from different aspects. Recent state-of-the-art results reported on Robust04 and GOV2 collections were achieved using the ELECTRA model that appears to outperform BERT. ELECTRA [40] replaces the Masked Language Modeling (MLM) with a novel more sample-efficient pre-training task called replaced token detection. In this task, the model learns to distinguish real input tokens from plausible but synthetically generated replacements by a small "generator" model. This approach uses two components: the generator, a small two-layer BERT model that predicts masked tokens and the ELECTRA discriminator model that both require training. However, the new objective allows the model to learn

Table 7.12 – Reranking effectiveness on the TREC DL 2019 and DL 2020 Document ranking tasks for Sim-Pair and vanilla models with both BERT and ELECTRA cores. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ are indicated with †, for the same core. Change rates over the vanilla baseline for the same core type are reported for each metric (%)

TREC DL Doc	DL 19				DL 20			
Model	nDCG@10		MAP		nDCG@10		MAP	
BM25	0.5176	–	0.2434	–	0.5286	–	0.3793	–
BM25+RM3	0.5169	–	0.2772	–	0.5248	–	0.4006	–
Vanilla $_{\text{BERT}}$	0.6726	–	0.3006	–	0.6340	–	0.4523	–
Sim-Pair $_{\text{BERT}}$	0.6798	▲1.1%	0.3057	▲1.7%	**0.6495**	▲2.4%	0.4505	▽0.4%
Vanilla $_{\text{ELECTRA}}$	0.6738	–	0.2976	–	0.6236	–	0.4297	–
Sim-Pair $_{\text{ELECTRA}}$	**0.6816**	▲1.2%	**0.3062**	▲2.9%	0.6331	▲1.5%	**0.4543**†	▲5.7%

from all input positions rather than only 15% of the positions in the MLM task.

In order to be confident in our approach, we investigate if exact match marking is beneficial for a BERT variant pre-trained on a more robust task, and study:

RQ4. *Is exact match marking beneficial for alternative transformer-based models such as ELECTRA?*

For our experiments, we use the base version of the ELECTRA model as the core of our model architecture illustrated in Figure 7.1 in Section 7.3, as a replacement of the BERT model. We use the same fully-connected layer for relevance classification $R(d, q)$ quantifying how relevant the candidate document d is to the query q. We also use the same fine-tuning hyper parameters used with BERT.

5.4.1 *In-domain effectiveness*

Using the same setting used for the BERT-based models, we report the results obtained on TREC DL 2019-2020 document ranking test collections in Table 7.12. For clarity, we only show results with the Sim-Pair marking strategy, full results with all the strategies can be found in Appendix 2.1.

Interestingly, using the ELECTRA core in place of BERT in the vanilla baseline does not lead to increased performance and we even observe a degradation in performance on TREC DL 2020. With exact match marking, Sim-Pair models using both BERT and ELECTRA cores, leads to similar gains over the vanilla baselines. While the gain in average precision is more pronounced with ELECTRA on both DL 2019 and 2020, the effectiveness in terms of nDCG@10 is more interesting with the BERT core on the DL 2020 test collection.

Table 7.13 – Reranking effectiveness in the zero-shot transfer setting for the Sim-Pair and vanilla models on Robusto4 and GOV2 collections using both BERT and ELECTRA cores. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively for the same core. Change rates over the vanilla baseline, for the same core type, are reported for each metric (%)

Robusto4	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4058	–	0.3345	–	0.4240	–	0.3631	–
BM25+RM3	0.4407	–	0.3821	–	0.4255	–	0.3661	–	0.4407	–	0.3821	–
Vanilla $_{\text{BERT}}$	0.4652	–	0.4046	–	0.4510	–	0.3851	–	0.4845	–	0.4147	–
Sim-Pair $_{\text{BERT}}$	**0.4773**	▲2.6%	**0.4155**	▲2.7%	**0.4931**‡	▲9.3%	**0.4169**‡	▲8.3%	**0.5239**‡	▲8.1%	**0.4446**‡	▲7.2%
Vanilla $_{\text{ELECTRA}}$	0.4416	–	0.3833	–	0.4482	–	0.3831	–	0.4782	–	0.4141	–
Sim-Pair $_{\text{ELECTRA}}$	0.4717‡	▲6.8%	0.4124	▲7.6%	0.4597	▲2.6%	0.3886	▲1.4%	0.5043‡	▲5.5%	0.4263	▲2.9%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	–	0.5362	–	0.4264	–	0.4705	–	0.4774	–	0.5362	–
BM25+RM3	0.4851	–	0.5634	–	0.4212	–	0.4966	–	0.4851	–	0.5634	–
Vanilla $_{\text{BERT}}$	0.4533	–	0.5272	–	0.4696	–	0.5248	–	0.4937	–	0.5611	–
Sim-Pair $_{\text{BERT}}$	0.4468	▽1.4%	0.5134	▽2.6%	0.4687	▽0.2%	0.5326	▲1.5%	0.4991	▲1.1%	0.5695	▲1.5%
Vanilla $_{\text{ELECTRA}}$	0.4668	–	0.5332	–	0.4986	–	0.5601	–	0.5147	–	0.5765	–
Sim-Pair $_{\text{ELECTRA}}$	**0.4881**‡	▲4.6%	**0.5577**‡	▲4.6%	**0.5030**	▲0.9%	**0.5634**	▲0.6%	**0.5249**	▲2.0%	**0.5923**	▲2.7%

5.4.2 *Zero-shot transfer setting*

We use the fine-tuned models on exclusively out-of-domain data, i.e MS MARCO passage dataset, and apply inference on the window-passages obtained by splitting each document using the same passage length of 150 words and a 75 words stride used in the BERT experiments. Table 7.13 shows the results obtained at cutoff 1,000 on both Robusto4 and GOV2 collections. We recall the results of the Vanilla and Sim-Pair models with the BERT core for comparison.

Exact Match Marking on ELECTRA. Results indicate clearly that adding exact match marking is also beneficial for the ELECTRA variant. Similarly to its BERT counterpart, Sim-Pair $_{\text{ELECTRA}}$ is more effective on Robusto4 with an average improvement rate of +5% nDCG@20 compared to only half, +2.5%, on GOV2. Interestingly, exact match marking has more notable impact on titles rather than descriptions, even though the vanilla $_{\text{ELECTRA}}$ baseline prefers description queries. Thanks to exact match marking, Sim-Pair $_{\text{ELECTRA}}$ achieves significant gains over the BM25 + RM3 baseline on Robusto4 titles and performs comparably on GOV2 titles.

ELECTRA vs. BERT core. The Sim-Pair$_{\text{ELECTRA}}$ variant achieves better performance than its BERT counterpart regardless of the topic field on the GOV2 collection. In contrast, using the BERT core is more effective on Robusto4 on both titles, descriptions and the hybrid pipeline. The same tendency can be observed for the vanilla baseline with smaller margins.

Table 7.14 – Reranking effectiveness in the multi-phase fine-tuning setting for the Sim-Pair and vanilla models on Robusto4 and GOV2 collections using both BERT and ELECTRA cores. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively for the same core. Significant inferiority with $p < 0.05$ is marked with $*$. Change rate over the vanilla baseline for the same core type are reported for each metric (%)

Robusto4	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	–	0.3631	–	0.4058	–	0.3345	–	0.4240	–	0.3631	–
BM25+RM3	0.4407	–	0.3821	–	0.4255	–	0.3661	–	0.4407	–	0.3821	–
Vanilla $_{\text{BERT}}$	0.4995	–	0.4275	–	0.5368	–	0.4492	–	0.5546	–	0.4715	–
Sim-Pair $_{\text{BERT}}$	0.5058	▲1.3%	0.4371	▲2.2%	0.5479†	▲2.1%	0.4574†	▲1.8%	0.5701‡	▲2.8%	0.4815‡	▲2.1%
Vanilla $_{\text{ELECTRA}}$	0.5375	–	0.4560	–	0.5676	–	0.4663	–	0.5901	–	0.4902	–
Sim-Pair $_{\text{ELECTRA}}$	**0.5380**	▲0.1%	**0.4564**	▲0.1%	**0.5686**	▲0.2%	**0.4705**	▲0.9%	**0.5927**	▲0.4%	**0.4942**	▲0.8%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	–	0.5362	–	0.4264	–	0.4705	–	0.4774	–	0.5362	–
BM25+RM3	0.4851	–	0.5634	–	0.4212	–	0.4966	–	0.4851	–	0.5634	–
Vanilla $_{\text{BERT}}$	0.5476	–	0.6302	–	0.5175	–	0.5772	–	0.5909	–	0.6604	–
Sim-Pair $_{\text{BERT}}$	0.5743‡	▲4.9%	0.6540‡	▲3.8%	0.5406‡	▲4.5%	0.6084‡	▲5.4%	0.5998	▲1.5%	0.6758	▲2.3%
Vanilla $_{\text{ELECTRA}}$	0.5784	–	0.6621	–	**0.5629**	–	**0.6279**	–	**0.6149**	–	0.6862	–
Sim-Pair $_{\text{ELECTRA}}$	**0.5868**	▲1.5%	**0.6661**	▲0.6%	0.5552	▽1.4%	0.6225	▽0.9%	0.6133	▽0.3%	**0.6926**	▲0.9%

5.4.3 *Multi-phase fine-tuning*

Table 7.14 shows the results obtained using the multi-phase fine-tuning on both MS MARCO passage dataset and in-domain labeled data, described in section 5.3 for BERT. Interestingly, the ELECTRA-based models benefit more from the additional in-domain fine-tuning to outperform the BERT-based models on both collections regardless of the topic field. With this increased performance, adding exact match marking has less impact on ELECTRA compared to BERT. This suggests that the ELECTRA variants in this setting have already enough representations capabilities tailored for the target tasks, that explicit exact matching hints are no longer required.

Exact match marking is indeed beneficial for the ELECTRA model in a zero-shot transfer setting where no labelled data is available in the target domain, which is the most common scenario. Sim-Pair $_{\text{ELECTRA}}$ is able to achieve significant gains on titles, where Sim-Pair $_{\text{BERT}}$ is less effective. However, for description and hybrid runs that use descriptions for reranking, exact match marking appears to have more substantial impact when using a BERT core. On TREC DL 2019 and 2020 document ranking benchmarks, both vanilla and Sim-Pair models perform similarly with both BERT and ELECTRA cores. The only advantage of the ELECTRA core is increased average precision with Sim-Pair. Finally, we can say that, in most cases, the

Table 7.15 – Reranking effectiveness of the Sim-Pair $_{\text{BERT}}$ with interpolating BM25 scores vs. Birch (MS) baseline on both Robust04 and GOV2 collections. Results are obtained after reranking the top-100 documents returned by BM25 following the setting used for the Birch(MS) baseline in Li et al. [122]. BM25 results are reported at cutoff 100

Robust04	Title run				Description run			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4407	-	0.3821	-	0.4041	-	0.3468	-
Birch (MS)	0.4227	-	0.3616	-	0.4053	-	0.3341	-
Sim-Pair $_{\text{BERT}}$ + BM25	**0.4839**	+14.5%	**0.4159**	+15.0%	**0.4969**	+22.6%	**0.4098**	+22.7%
GOV2	Title run				Description run			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-
Birch (MS)	0.4722	-	0.5352	-	0.4260	-	0.4701	-
Sim-Pair $_{\text{BERT}}$ + BM25	**0.5432**	+15.0%	**0.6117**	+14.3%	**0.5068**	+19.0%	**0.5651**	+20.2%

ELECTRA-based versions of our models are more effective compared to their BERT counterparts.

5.5 *Comparison with state-of-the-art baselines*

In this section we try to situate our approach with regard to what has already been proposed for document ranking. In a first part, we try to conduct comparative evaluations with models presenting a similar experimental setup for a fair comparison. Then in a second part, we compare our best runs to a wide variety of SOTA approaches with different configurations.

5.5.1 *Comparison in the same experimental design*

In order to fairly compare a novel approach with previously proposed ones, it is important to conduct the evaluation in the same experimental conditions. Here, we try to reproduce as much of the original settings used to produce the results of the Birch and BERT-maxP baselines, respectively. These approaches being the most similar to our ranking configuration as cross-encoders using a monoBERT core.

Birch (MS). This baseline is fine-tuned exclusively on MS MARCO passages, therefore we use our Sim-Pair $_{\text{BERT}}$ + BM25 model equally fine-tuned on MS MARCO passages and augmented with BM25 scores interpolation following the same Equation 7.4 used in Birch [2]. All Robust04 and GOV2 topics and relevance judgements are used as a held-out test set.

Table 7.15 shows the results of our Sim-Pair $_{\text{BERT}}$ + BM25 model compared to the Birch (MS) baseline. Following the setting used by Li et al. [122] for the

Table 7.16 – Reranking effectiveness of the Sim-Pair BERT with multi-phase fine-tuning vs. BERT-MaxP (MS) baseline on both Robusto4 and GOV2 collections. [MS] indicates that the run uses MS marking: exact match marking is only used during fine-tuning on MS MARCO and ablated in the in-domain fine-tuning phase

Robusto4	Title run				Description run			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	-	0.3631	-	0.4058	-	0.3345	-
BERT-MaxP (MS)	0.4931	-	0.4277	-	0.5453	-	0.4522	-
Sim-Pair BERT	**0.5058**	+2.6%	**0.4371**	+2.2%	0.5479	+0.5%	0.4574	+1.1%
Sim-Pair BERT [MS]	0.4978	+1.0%	0.4281	+0.1%	**0.5521**	+1.2%	**0.4592**	+1.5%
GOV2	**Title run**				**Description run**			
Model	nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-
BERT-MaxP (MS)	0.5600	-	0.6352	-	0.5506	-	0.6087	-
Sim-Pair BERT	**0.5743**	+2.6%	**0.6540**	+3.0%	0.5406	−1.8%	0.6084	−0.0%
Sim-Pair BERT [MS]	0.5665	+1.2%	0.6430	+1.2%	**0.5509**	+0.1%	**0.6161**	+1.2%

Birch (MS) baseline, we report results using cutoff 100 in the BM25 first-stage retrieval. The results clearly indicate that our model outperforms Birch (MS).

BERT-MaxP (MS). The configuration of this baseline is the same we used in the multi-phase fine-tuning setting. We compare the results of Sim-Pair BERT fine-tuned first on MS MARCO and then further fine-tuned on the target task obtained with a 5-fold cross validation with BERT-MaxP (MS) in Table 7.16. We report the results when using the exact match marking during fine-tuning on MS MARCO passages only [MS], and the results with the full marking on both MS MARCO and in-domain data, i.e., Sim-Pair BERT. Our approach outperforms clearly the BERT-maxP baseline on titles, and performs slightly better on descriptions. It is important to notice that the BERT-MaxP results reported by Li et al. [122] are better than our vanilla BERT baseline in the multi-phase fine-tuning setting, especially on GOV2. This slight difference can be exaplined by our adoption of the traditional pointwise loss function (monoBERT [173]) in contrast to the authors use of a pairwise loss function.

5.5.2 *Comparison with different experimental designs*

Each approach has the optimal experimental conditions that lead to the best ranking accuracy possible, and these optimal conditions are hardly the same for the different models we want to compare. Independently of the experimental framework employed to obtain the results, or the nature of the approach, Table 7.17 compares our best runs with both BERT and ELECTRA cores obtained in the multi-phase fine-tuning setting, with the best baseline

Table 7.17 – Reranking effectiveness on Robust04 and GOV2 of our best runs vs. the best baseline runs. The change rate (%) of our best run, Sim-Pair ELECTRA, over each baseline is indicated for both metrics if available. We use the multi-phase fine-tuning for our runs, the same multi-phase fine-tuning is adapted in Parade and BERT-maxP baselines. For a fair comparison with sparse and dense retrieval models we add Sim-Pair runs in the zero-shot setting on descriptions. Our best results are indicated in **bold**, and overall best results among baselines are underlined

Runs	Robust04					GOV2				
	nDCG@20		P@20		Field	nDCG@20		P@20		Field
Lexical Retrieval										
[01] BM25	0.4240	+40.%	0.3631	+36.%	Title	0.4774	+28.%	0.5362	+29.%	Title
[02] BM25+RM3	0.4407	+34.%	0.3821	+29.%	Title	0.4851	+26.%	0.5634	+23.%	Title
Sparse Retrieval										
[03] BOW+DeepCT-Query	0.4450	+33.%	-	-	Desc	0.4300	+43.%	-	-	Desc
[04] BM25+DocT5Query	0.4076	+45.%	0.3361	+47.%	Title	-	-	-	-	-
Dense Retrieval										
[05] DPR	0.1832	+223%	0.1508	+228%	Title	0.1618	+279%	0.1644	+321%	Desc
[06] ANCE	0.3517	+69.%	0.2767	+79.%	Desc	0.3604	+70.%	0.3738	+85.%	Desc
[07] ColBERT	0.3919	+51.%	0.3275	+51.%	Title	-	-	-	-	-
Zero-shot Setting										
[08] Sim-Pair BERT	0.4931	+20.%	0.4169	+19.%	Desc	0.4687	+31.%	0.5326	+30.%	Desc
[09] Sim-Pair ELECTRA	0.4597	+29.%	0.3886	+27.%	Desc	0.5030	+22.%	0.5634	+23.%	Desc
Reranking										
[10] Birch (MS-MB)	0.5137	+15.%	0.4404	+12.%	Title	0.5608	+9.4%	0.6409	+8.1%	Title
[11] BERT-MaxP (MS)	0.5453	+8.7%	0.4522	+9.3%	Desc	0.5600	+9.5%	0.6356	+9.0%	Title
[12] Parade	0.5605	+5.7%	0.4661	+6.0%	Desc	0.5750	+6.7%	0.6530	+6.1%	Title
[13] Parade ELECTRA	0.5713	+3.7%	0.4717	+4.8%	Desc	0.5851	+4.8%	0.6678	+3.7%	Title
[14] Parade-v2-Transformer	0.6127	−3.3%	0.5255	−6.0%	Desc	0.6093	+0.7%	0.6651	+4.0%	Title
[15] monoT5-base	0.5298	+12.%	-	-	Hybrid	-	-	-	-	-
[16] monoT5-large	0.5345	+11.%	-	-	Hybrid	-	-	-	-	-
[17] monoT5-3B	0.6091	−2.7%	-	-	Hybrid	-	-	-	-	-
Our Runs										
[18] Sim-Pair BERT	0.5701	-	0.4815	-	Hybrid	0.5998	-	0.6758	-	Hybrid
[19] Sim-Pair ELECTRA	**0.5927**	-	0.4942	-	Hybrid	**0.6133**	-	0.6926	-	Hybrid

runs. While Table 7.18 compares our best in-domain runs to both TREC best runs from the TREC DL 2019 and 2020 tracks and the SOTA baselines.

Robust04 and GOV2 collections. Unsurprisingly, the reranking models achieve the best results and largely outperform all other baselines. For a fair comparison with the sparse and dense retrieval methods (runs [03-07]) which do not use target-domain fine-tuning, we add our runs in the zero-shot setting on descriptions (runs [08-09]). Nevertheless, our rerankers still outperform the dense retrievers.

Results obtained using the best Sim-Pair BERT, run [18] in Table 7.17, outperform all the BERT-based models that represent the state of the art and achieves better performance than monoT5 for both base and large versions on robust04. The Sim-Pair ELECTRA variant (run [19]) achieves comparable performance with the monoT5-3B model while using only 3.6% of its parameters.

Table 7.18 – Reranking effectiveness on TREC DL 2019 and 2020 Document ranking tasks of our Sim-Pair models with both BERT and ELECTRA cores vs. the best TREC runs and baselines. The change rate (%) of our best run, over each baseline is indicated for both metrics if available. DPR* and ANCE* results were copied from the ANCE paper [251]. Our best results are indicated in **bold**, and overall best results among baselines are underlined

Runs	DL 2019				DL 2020			
	nDCG@10		MAP@100		nDCG@10		MAP@100	
Lexical Retrieval								
[01] BM25	0.5176	+31.%	0.2434	+26.%	0.5286	+23.%	0.3793	+19.%
[02] BM25+RM3	0.5170	+32.%	0.2774	+10.%	0.5225	+24.%	0.4014	+12.%
Sparse Retrieval								
[03] BM25+HDCT	0.4523	+50.%	0.2067	+13.%	0.4506	+44.%	0.3022	+49.%
[04] BM25+DocT5Query	0.5968	+14.%	0.2700	+13.%	0.5885	+10.%	0.4230	+6.5%
Dense Retrieval								
[05] DPR*	0.5570	+22.%	-	-	-	-	-	-
[06] ANCE*	0.6150	+10.%	-	-	-	-	-	-
[07] ColBERT	0.5756	+18.%	0.1914	+60.%	0.5481	+18.%	0.2963	+52.%
Reranking-TREC								
[08] ucas_runid1	0.6437	+5.7%	0.2642	+16.%	-	-	-	-
[09] idst_bert_r1	0.7189	−5.4%	0.2915	+5.0%	-	-	-	-
[10] Parade-v2-Tranformer	0.6500	+4.6%	0.2740	+12.%	0.6010	+8.1%	0.4030	+12.%
[11] Parade-v2-Max	0.6790	%	0.2870	%	0.6130	%	0.4200	%
[12] d_d2q_duo	-	-	-	-	0.6934	−6.3%	0.5422	−17.%
[13] ICIP_run1	-	-	-	-	0.6623	−1.9%	0.4333	+4.0%
Our Runs								
[14] Sim-Pair $_{\text{BERT}}$	0.6798	-	0.3057	-	**0.6495**	-	0.4505	-
[15] Sim-Pair $_{\text{ELECTRA}}$	**0.6801**	-	**0.3061**	-	0.6331	-	**0.4543**	-

The monoT5 baseline is by far the strongest baseline, it is important to note that it uses a zero-shot transfer setting without the need for in-domain fine-tuning as opposed to BERT-MaxP, Parade and our best runs [18-19], however, its large size make it unpractical compared to a BERT$_{\text{Base}}$ or ELECTRA $_{\text{Base}}$. For the Parade baselines, our best runs outperform the Parade, and Parade $_{\text{ELECTRA}}$ model on both Robust04 and GOV2 collections by a varying margin from +3% to more than +4%. However, the Parade-v2-Transformer with its improved training and reranking threshold of 1000 outperforms our models, but also all other baselines including monoT5-3B.

TREC DL Document Ranking task. Similarly to the Robust04 and GOV2 results, the best TREC runs which are cross-encoding rerankers outperform all other baselines. For TREC DL 2019, we include the best `idst_bert_r1` run [253] which uses StructBERT [243], a BERT model which better models sentence relationships thanks to an improved Next Sentence Prediction task, and `ucas_runid1` [37] which uses BERT-MaxP [51]. We also include Parade-v2

results [123]. Interestingly, our runs outperform Parade-v2-Transformer by large margins in the in-domain scenario. Our best runs also outperform Parade-v2-Max on TREC DL 2020, and perform similarly on DL 2019.

Our best runs further outperform `ucas_runid1` but cannot outperform `idst_bert_r1` –StructBERT core– in terms of nDCG@10. In TREC DL 2020, the best run `d_d2q_duo` [192] is a large multi-stage ranking model including a BM25 retriever, DocT5Query document expansion and two cascading T5-3B rerankers, making hard to outperform. The `ICIP_run1` [36], uses a BERT-Large model at its core with a refined fine-tuning process including passage filtering and better negative sampling which explains its higher performance. Nevertheless, our runs are still competitive and outperform Parade-v2 which has the same model size as our models. Interestingly, the performance on TREC DL 2020 are lower in terms on nDCG@10 compared to TREC DL 2019 for the same model as observed for both our runs and the Parade-v2 run.

5.6 *Investigating the Contextualized Representations of Marker Tokens*

We have proposed marking strategies to emphasize exact match signals by introducing marker tokens in the input of a PLM encoder. However, the output representations produced by the encoder for these spacial marker tokens are not explicitly put to use for relevance scoring. Instead, the contextualized representation of the standard [CLS] token is taken as an aggregate representation of the entire input sequence, including the marker tokens. This representation is then fed through the classification layer for relevance prediction following the standard monoBERT architecture.

In this section, we are interested in the contextualized representations produced by the BERT encoder for the marker tokens, and how they can contribute to relevance prediction. We propose alternatives to the standard [CLS] token representation, and explicitly make use of the contextualized representations of the marker tokens to extract a fixed-size representation of the query-document input sequence for relevance classification, and answer:

RQ5. *Can the contextualized representations of the marker tokens be effectively and explicitly leveraged for relevance prediction?*

5.6.1 *Query-Document Sequence Representation*

In addition to the standard [CLS] representation pooling, we introduce new variations for representing the query-document input sequence which rely on pooling the last hidden representations of the transformer model, which we define as $T = [T_{[CLS]}, U_1, ...U_n, T_{[SEP]}, V_1, ..., V_m, T_{[SEP]}]$, where $U_{1:n}$ and $V_{1:m}$ are the last hidden representations, i.e., the contextualized representations, of the query tokens $q_{1:n}$ and document tokens $d_{1:m}$, respectively. The output

representation obtained for input sequence is denoted $h_o \in \mathbb{R}^H$, where H is the size of the representation vector.

[CLS] POOLING The contextualized representation of the [CLS] token $T_{[CLS]}$ is commonly used as a fixed-size representation of the entire input sequence $h_o = T_{[CLS]}$ as recommended by Devlin et al. [58], and adopted in monoBERT [173]. We used this standard representation in all our previous experiments.

MARKER TOKEN POOLING The contextualized representations of the marker tokens (i.e., the simple marker "#" used in Sim-Pair) are pooled from the output of the BERT encoder to produce a fixed-size representation of the input sequence based on exact match signals.

More formally, given the contextualized representations produced by the BERT encoder for the input sequence, that is:

$$T = [T_{[CLS]}, U_1, ...U_n, T_{[SEP]}, V_1, ..., V_m, T_{[SEP]}] \tag{7.5}$$

we first pool the marker token representations from the query segment $U_{1:n}$ to produce a query representation h_q, and extract similarly a document representation h_d from the marker tokens in the document segment $V_{1:m}$. Then, we concatenate the query and document representations to get a single input representation:

$$h_o = h_q \oplus h_d \tag{7.6}$$

where $h_o \in \mathbb{R}^H$, and $H = D + D$ with D being the hidden size of the BERT encoder.

We define three methods for extracting a segment representation (i.e., h_q and h_d) from its marker token representations:

1. **Avg.** The segment representation is obtained by a mean average pooling over the representations of its marker tokens;
2. **Max.** The segment representation is obtained by a max pooling over the representations of its marker tokens. This pooling only keeps the strongest signals from the marker token representations;
3. **First.** The segment is represented by the contextualized representation of its first occurring marker token. The intuition behind this pooling is that a user can judge the relevance of a document at the first exact matching term;

These new pooling methods can require more operations to build the output representations h_o compared to standard [CLS] pooling.

5.6.2 *Relevance Classification*

Figure 7.3 illustrates the full architecture of the novel representation variant introduced above. In all variants, the output representation of the query-

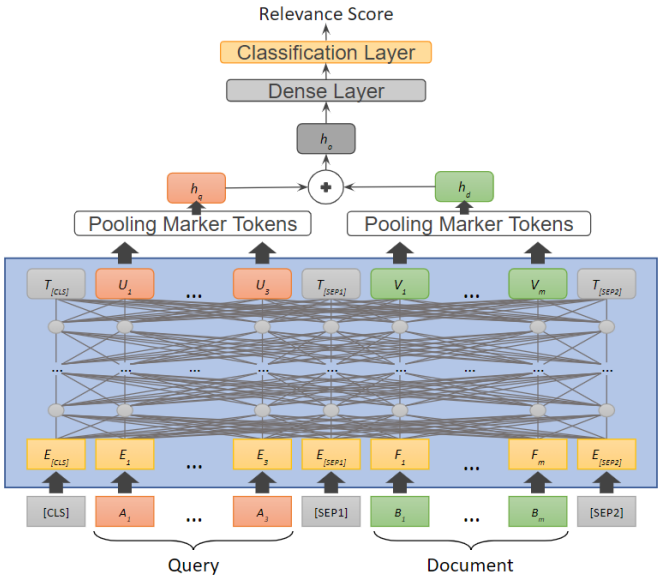


Figure 7.3 – The architecture of the novel representation variant relying on marker token contextualized representations for relevance extraction.

document sequence h_o from the BERT model is fed into a dense layer with a linear activation [11], and obtain h'_o. Then, for the binary relevance classification task, we use single classification layer $W \in \mathbb{R}^{2\times H}$. Since the marker token pooling methods produce an output vector h_o with double the number of dimensions ($H = D + D$) compared to the [CLS] pooling method ($H = D$), the new model variants require more parameters in the relevance classifier; Nonetheless, these additional parameters are negligible, considering the large number of BERT parameters.

During fine-tuning, the classification loss is the standard cross-entropy of the *softmax* of $h_o \cdot W$ with respect to the true relevance label as in the original monoBERT model.

5.6.3 *Experimental setup*

We focus on the Sim-Pair $_{\text{BERT}}$ model and modify its output representation to the variants defined previously. For a fair comparison, we follow the same experimental design described in Section 4.1 for fine-tuning, and for evaluating the new representation variants on both the in-domain and out-of-domain collections. For out-of-domain generalizability, we only report results in the zero-shot setting as it is the most common setting in real-world

11. It is important to note that BERT integrates a pre-trained pooler for the [CLS] token representation that uses a dense layer with the *tanh* activation function. We follow exactly the same process with the pooled representations of the query and document segments h_q and h_d, respectively.

Table 7.19 – Reranking effectiveness of the Sim-Pair BERT with representation variants on TREC DL 2019-2020 document ranking collections. Best results are indicated in **bold**

		DL 2019		DL 2020	
Model	Output Rep.	nDCG@10	MAP@100	nDCG@10	MAP@100
BM25 + RM3	-	0.5169	0.2772	0.5248	0.4006
Vanilla BERT	CLS pooling	0.6726	0.3006	0.6340	**0.4523**
Sim-Pair BERT	CLS pooling	0.6798	**0.3057**	**0.6495**	0.4505
Sim-Pair BERT	Avg pooling	0.6529	0.2732	0.6274	0.4244
	Max pooling	0.6684	0.2876	0.6441	0.4327
	First pooling	**0.6895**	0.2849	0.6321	0.4372

applications. Moreover, we only use the hybrid runs as they give the best retrieve-then-rerank setting.

5.6.4 *Results and Discussion*

Table 7.19 reports the results of the different representation variants of the Sim-Pair BERT model on the TREC DL 2019 and 2020 document ranking collections. Interestingly, restricting the relevance classification decision to the information encoded in the marker token representations leads, on average, to on par reranking performance with using the standard [CLS] pooling in terms of nDCG@10. In other words, the exact matching signals from the marker tokens have enough representation capacity to model relevance and achieve a reranking performance close to the vanilla baseline on both collections.

Extracting the strongest signals from the marker token representations in a segment with the max pooling method outperforms the simple average pooling method. On the TREC DL 2020 collection, the first pooling method leads to a reranking performance on par with the vanilla baseline, but slightly worse than the max pooling method. On the TREC DL 2019 collection, however, using only the representation of the first marker token in a segment is enough to outperform the performance of the standard [CLS] representation pooling used in both the Vanilla and Sim-Pair models. The information encoded in the marker token representations produced by the BERT encoder show promising potential, and other pooling methods can be build around them for relevance classification.

The results of the zero-shot transfer to the out-of-domain collections, namely: Robust04 and GOV2, with the representation variants are shown in Table 7.20. Similarly to the in-domain results, relying exclusively on the representations of the marker tokens for relevance decisions is able to achieve

Table 7.20 – Reranking effectiveness of the Sim-Pair $_{\text{BERT}}$ with representation variants on both Robust04 and GOV2 collections. We report results using the hybrid runs at cutoff 1000. Best results are indicated in **bold**

		Robust04		GOV2	
Model	Output Representation	nDCG@20	P@20	nDCG@20	P@20
BM25	-	0.4240	0.3631	0.4774	0.5362
Vanilla $_{\text{BERT}}$	CLS pooling	0.4845	0.4147	0.4937	0.5611
Sim-Pair $_{\text{BERT}}$	CLS pooling	**0.5239**	**0.4446**	**0.4991**	**0.5695**
Sim-Pair $_{\text{BERT}}$	Avg pooling	0.4805	0.4012	0.4908	0.5497
	Max pooling	0.4860	0.4133	**0.4993**	0.5644
	First pooling	0.4832	0.4082	0.4983	0.5574

on par performance with the vanilla baseline. Notably, using the max pooling method on the GOV2 collection achieves on par results with the [CLS] pooling method with the Sim-Pair model. However, none of the newly introduced pooling methods is able to match the performance of the Sim-Pair model with [CLS] pooling. At the end, using the exact match signals encoded in the contextualized representations of the marker tokens is powerful enough to match the performance of the vanilla baseline which relies on [CLS] pooling (i.e., all tokens). Nevertheless, it does not seem to be able to outperform the standard [CLS] pooling with Sim-Pair $_{\text{BERT}}$ on Robust04, in which the aggregate representation combines the signals from all tokens in the sequence including the marker tokens to achieve significantly better performance than both the vanilla baseline and all remaining pooling methods.

Experiments with pooling representations of marker tokens show the potential of such special tokens for encoding important relevance signals, which can replace the standard [CLS] representation in a Vanilla model (i.e., without exact match marking). Nevertheless, using [CLS] pooling over our Sim-Pair $_{\text{BERT}}$ model to build a more complete aggregate representation combining interactions between existing tokens and introduced marker tokens achieves more important gains in performance notably on Robust04.

5.7 *Marker Tokens for Query Expansion*

We have seen that the contextualized representations of marker tokens produced by the BERT encoder carry valuable information for relevance classification. These marker tokens are strategically placed around query-document overlapping terms to bring focus on them as they are considered to be important in traditional IR. A natural extension of this idea, is to explore the use of marker tokens for other purposes such as query expansion.

We propose padding the query segment with marker tokens whose responsibility is not to highlight existing query terms but rather allow BERT to encode new query embeddings to expand the existing query-term embeddings, and study the following research question:

RQ6. *Can marker tokens be used for implicit and differentiable query expansion?*

Given a query q and a candidate document d, we first tokenize them into their WordPiece tokens $q_{1:n}$ and $d_{1:m}$, respectively. Then, we pad the query tokens with a fixed number N of marker tokens "#" within the maximum query length range. Finally, we concatenate the two sequences of tokens separated with the special [SEP] token, we also close the entire sequence with a [SEP] token and prepend the token [CLS] to the resulting sequence as follows:

$$S = [[CLS], q1, ...q_n, \#, ...\#, [SEP], d_1, ...d_m, [SEP]] \tag{7.7}$$

This query expansion is intended to serve as a differentiable mechanism for learning to expand queries with new token representations or re-weigh existing tokens based on their importance for the relevance matching. This approach was used by ColBERT [112] to expand the query representations in a bi-encoder architecture using the [MASK] token.

5.7.1 *Experimental Setup*

We apply the marker-token-based query expansion technique introduced above on both the Vanilla BERT baseline and the Sim-Pair BERT model with the standard [CLS] pooling representation. In the vanilla baseline, the marker token "#" is exclusively used for query expansion, while it has has a double role in Sim-Pair: highlight the exact term matches, and expand the query.

We keep the same fine-tuning and evaluation settings described in Section 4.1, and fix the number N of expansion tokens to 8 for all collections. We report results on in-domain TREC DL collections, and zero-shot generalizability to out-of-domain collections on Robust04 and GOV2. We use the title queries for the initial retrieval for better recall and then investigate the effect of the implicit query expansion technique on both titles and descriptions in the rerankers. That is, we use the title and hybrid runs which share the same initial candidates (retrieved with BM25 using titles) and then use titles and descriptions, respectively for reranking with the BERT-based models.

5.7.2 *Results and Discussion*

Table 7.21 reports the results of our query expansion approach applied on the Vanilla and Sim-Pair models with the BERT core on TREC DL document ranking collections. The results show that padding the TREC DL queries with marker tokens does not bring any gains in ranking performance. The marker-token-based query expansion leads to slight losses in effectiveness on the

Table 7.21 – Reranking effectiveness with marker-token-based query expansion on TREC DL 2019-2020 document ranking collections. Best results are indicated in **bold**

		DL 2019		DL 2020	
Model	Query Expansion	nDCG@10	MAP@100	nDCG@10	MAP@100
BM25 + RM3	RM3	0.5169	0.2772	0.5248	0.4006
Vanilla $_{\text{BERT}}$	-	0.6726	0.3006	0.6340	0.4523
	Marker Tokens	0.6665	0.2977	0.6206	0.4442
	[MASK] Tokens	0.6588	0.2962	0.6254	0.4414
Sim-Pair $_{\text{BERT}}$	-	0.6798	**0.3057**	0.6495	0.4505
	Marker Tokens	0.6568	0.2913	0.5991	0.4275
	[MASK] Tokens	**0.6859**	0.3022	0.**6578**	**0.4542**

Table 7.22 – Reranking effectiveness with marker-token-based query expansion on both Robust04 and GOV2 collections. We report results using the title and hybrid runs at cutoff 1000. QE indicates the query expansion technique used. Best results are indicated in **bold**. Significant improvements of the query expansion models are indicated with † and ‡ for $p < 0.05$ and $p < 0.01$, respectively

		Robust04				GOV2			
		Title runs		Hybrid runs		Title runs		Hybrid runs	
Model	QE	nDCG@20	P@20	nDCG@20	P@20	nDCG@20	P@20	nDCG@20	P@20
BM25	-	0.4240	0.3631	0.4240	0.3631	**0.4774**	0.5362	0.4774	0.5362
Vanilla $_{\text{BERT}}$	-	0.4652	0.4046	0.4845	0.4147	0.4533	0.5272	0.4937	0.5611
	Marker	**0.4820**†	**0.4183**†	0.5104‡	0.4321‡	0.4638	**0.5376**	**0.5091**†	**0.5701**
	[MASK]	0.4647	0.4032	0.4939	0.4213	0.4528	0.5238	0.4942	0.5614
Sim-Pair $_{\text{BERT}}$	-	0.4773	0.4155	**0.5239**	**0.4446**	0.4468	0.5134	0.4991	0.5695
	Marker	0.4539	0.3938	0.4524	0.3837	0.4634	0.5319	0.4823	0.5463
	[MASK]	0.4747	0.4153	0.5176	0.4386	0.4594	0.5332	0.5054	0.5685

vanilla baseline, meanwhile the losses are more prominent with the Sim-Pair model, where the marker tokens endorse two different roles simultaneously. This double role of exact match marking and query expansion assumed by the same marker token appears to be confusing for the BERT encoder to figure out. Therefore, we propose using a different token for query expansion for comparison. We chose the [MASK] token, which was used for the purpose of query expansion in ColBERT. In contrast to the marker token "#", the [MASK] token was pre-trained to generate the representation of the masked token form its context via the Masked Language Modeling objective in BERT. Thus, this token is more fitted to the query expansion task.

Using the [MASK] token instead of the marker token does not make the query expansion more effective with Vanilla $_{\text{BERT}}$. However, it leads to better results with Sim-Pair $_{\text{BERT}}$ and even achieves substantial gains in performance.

The results of the zero-shot transfer to Robust04 and GOV2 are shown in Table Table 7.22. Similarly to in-domain results, using the marker token for query expansion with the vanilla baseline leads to improvements, while using it with Sim-Pair leads to mitigated results. On Robust04, exact match marking provides substantial improvements in performance, but additionally using marker tokens for query expansion leads to degradation in performance. On GOV2, however, exact match marking brings slight gains in performance on description queries while it leads to degradation on title queries. Consequently, using the marker token for query expansion brings gains on titles, while degrading performance on descriptions. Interestingly, the results reveal larger gains on descriptions (i.e., hybrid runs) compared to titles (i.e., title runs). This suggest that richer query contexts inform better implicit query expansion.

Similarly to in-domain, replacing the marker token with the [MASK] token for query expansion does not achieve better results on the vanilla baseline. Meanwhile, it is a better fit with the Sim-Pair model on Robust04 and GOV2 descriptions (i.e., where exact match marking outperforms the vanilla baseline). Nevertheless, using marker tokens for emphasizing exact match signals without query expansion brings more substantial gains on Robust04.

Implicit query expansion with marker tokens yields interesting gains in zero-shot reranking performance on out-of-distribution test data with no impact on the model's efficiency (i.e., the expansion tokens are simply appended to the query after tokenization). Notably, richer query contexts provided by long natural descriptions inform the BERT encoder to build better contextualized expansion representations in place of the marker tokens. The results of this investigation suggest that marker tokens can be introduced in the input of PLMs for various purposes including: exact match marking and implicit query expansion. Nonetheless, assigning a single role to the marker tokens appears to be crucial for the encoder.

6 *Discussion and Conclusion*

We have shown throughout this chapter the empirical validation of the IR approach that we have proposed to harness the exact matching signals from the query-document pairs to enhance document ranking with pre-trained language models. To do so, we have defined an experimental protocol which allowed us to answer the our research questions as follows:

We have shown that using marking strategies to explicitly emphasize exact matching signals can enhance the performance of pre-trained language models, in the context of reranking, on the ad hoc document ranking task. The

experimental results on in-domain and out-of-domain benchmarks showed that combining a simple marker soft marker with a pair marking strategy (Sim-Pair) is the most simple yet effective marking strategy. Notably, highlighting exact term matches in the input sequence improves, significantly, the zero-shot generalizability of the monoBERT model to the out-of-distribution Robust04 data. Moreover, experiments confirm the ability of BERT to leverage the rich long natural language descriptions of the Robust04 and GOV2 collections. And since we follow a retrieve-then-rerank architecture where the retriever is a BoW model that prefers short keyword queries, we propose a hybrid pipeline where short keyword queries are used during the initial retrieval, and then replaced by descriptions in the reranking stage which leads to substantial gains in performance.

We investigated the contribution of the lexical scores from the first-stage retriever to the end effectiveness of the entire reranking architecture. We found that exact term matching scores from the traditional BoW model, BM25, are still beneficial for BERT-based document reranking for out-of-domain collections. More importantly, using exact match marking does not appear to require as much contribution from BM25's exact matching scores to achieve better performance compared to the vanilla model. This suggests that our exact match marking strategy does, indeed, capture exact match signals.

In a multi-phase fine-tuning setting, we showed that adding in-domain fine-tuning on top of the first general-purpose fine-tuning phase on out-of-domain data leads to better ranking performance. We further demonstrated through an ablation study, that using exact match marking in the general-purpose fine-tuning phase on large out-of-domain data is enough to achieve substantial leaps in performance especially on descriptions.

We would argue that our exact match marking induces focus on exact match signals leading to better performance than a vanilla model (in 24 comparisons, with 9 being significant), or at least to comparable performance (only in 4 comparisons, with no significant loss). Importantly, our extensive experiments did not show a single case where Sim-Pair performs significantly worse, thus it offers a more interesting alternative to the vanilla monoBERT. On the efficiency side, our approach inherits the efficiency issues of the cross-encoder architecture. However, we do not add more complexity to the model, making our approach a better substitute for a vanilla cross-encoder with the exact same architecture and number of parameters ($110M$). More so, marking is not needed for in-domain fine-tuning once the general-purpose fine-tuning phase is completed.

Overall, our exact match emphasizing approach based on marking strategies showed better performance over the vanilla baseline. It also produce competitive effectiveness compared to state-of-the-art models. Additionally, further investigation of the contextualized representations produced by BERT for our introduced marker tokens show promising potential, as the information

encoded in these special tokens has enough representation capacity to inform relevance classification decisions.

Finally, the proposed marking strategies are not limited to exact match emphasizing only. Marker tokens can be further applied for other purposes such as implicit query expansion which yields substantial gains in ranking performance in the zero-shot transfer setting to standard ad hoc retrieval collections exemplified by Robust04.

8

Investigating contextualized representations for ad hoc ranking

1 *Introduction*

Contextualized representations from pre-trained language models (PLMs), such as BERT [58], have become the default representations in neural information retrieval (IR), achieving state-of-the-art in text ranking with large performance leaps [129].

A category of neural IR models aims at learning contextualized representations suitable for ranking as part of the emerging research area on dense retrieval covered in Chapter 6. Many of these dense retrievers are *single-vector systems*, in which a PLM encodes each query and each document into a single vector, and relevance is given by a simple measure of similarity between the two vectors. Alternatively, ColBERT [112] introduces a new *late interaction* mechanism, where queries and documents are encoded into their token representations, and relevance is computed based on all-to-all soft matching between query and document token vectors. From the success of these models, which rely on simple soft matching of dense vectors to predict relevance, it is clear that PLMs effectively encode useful information for ranking in the output contextualized representations (i.e., the dense vectors). However, little is understood about what is precisely in these contextualized representations that largely surpasses the capabilities of static embedding methods [189, 162]. This lack of clarity raises the question of whether the full flexibility of PLMs' contextualization is required to achieve high-ranking performance.

The contextualization process in a transformer-based PLM starts with a static token embedding (augmented with a position embedding) that is gradually modified through multi-headed self-attention mechanisms to create a contextualized representation (see Section 3.3). This process could integrate various information such as syntax and semantics [230], how meanings vary across linguistic contexts (e.g., polysemy) [246], and the topic of a context. While it is still unclear what types of information play a role in state-of-the-art ranking models, prior work has provided some hints. The COIL model [75] is highly effective even while constraining soft matching to only the overlapping tokens between the query and the document (i.e., only soft matches between identical tokens are considered). The success of COIL suggests that the contextualized representations of the same token can further encode fine-

grained topic or sense information necessary for relevance estimation (i.e., distinguish between relevant and non-relevant occurrences of the same token). On the other hand, ColBERTv2 [217] qualifies the semantic space resulting from ColBERT as "lightweight", finding that it can be summarized, with high precision, by a set of static cluster centroids produced by k-means clustering. Given the contextualized representation of a token, it is first mapped to a cluster, where it is approximated by the embedding of the cluster centroid. Minor refinements are then added at the dimension level to better approximate the original contextualized representation (see Section 5.3). The effectiveness of ColBERTv2's compression technique implies that ColBERT contextualized representations can be distilled into static embeddings (i.e., cluster centroids) along with minor refinements.

Building on these insights, we investigate how well simplified approaches can approximate a PLM's contextualization process. This investigation aims to gain insight into the contextualization process of PLMs in the context of ranking. The results of our study can motivate more efficient architectures specifically designed for ranking by considering only what is necessary for the task.

Given a contextualized representation of a token produced by an "oracle" PLM, we devise aggregation methods to approximate this representation by combining a finite set of K learned static embeddings for this token, referred to as *sub-embeddings*. More specifically, we consider two types of aggregations: (1) Intrinsic aggregation (Section 3.1.1), which combines the sub-embeddings of the target token, and (2) extrinsic aggregation (Section 3.1.2), which combines the intrinsic representations of tokens in the local-context of the target token. Intuitively, intrinsic aggregation can model signals closely tied to a token like its senses or general topics commonly co-occurring with it (e.g., "bank" in finance), whereas extrinsic aggregation is better suited for incorporating fine-grained information like sub-topics (e.g., "bank" in the context of mortgage rates) from neighboring tokens.

The remainder of this chapter is organized as follows. Section 8.2 presents the motivation of our proposition and introduces the research questions we study. In Section 8.3, we present the aggregation methods we propose and their implementation as modules, followed by the detailed life cycle of these modules. Section 8.4 describes the experimental setup and Section 8.5 reports the evaluation results. We end this chapter with a conclusion.

2 *Motivation and Research Questions*

Transformer-based PLMs produce *contextualized representations* as a function of the entire context in which the term appears. The success of these contextualized models on a wide range of downstream tasks suggests that

contextualized representations capture highly transferable properties of language [135]. In an attempt to analyze these representations, several studies propose distilling them into static embeddings. Ethayarajh [63] show that BERT produces more context-specific representations in the upper layers in the same way upper layers of LSTMs produce more task-specific representations [135]. The authors then propose a method for distilling static embeddings by simply taking the first principal component of the contextualized representations of a token. Similarly, Bommasani et al. [20] introduce a simple and fully general method for distilling contextualized representations to static lookup-table embeddings outperforming standard static embeddings (e.g., GloVe [189]), by aggregation over different contexts. Alternatively, Wang et al. [244] uses BERT representations to train static embeddings following the Skip-Gram framework, enhancing performance on lexical semantic tasks.

These approaches distill the rich contextualized representations of all term occurrences into a *single* static embedding. However, this means that polysemous occurrences of a token will share the same embedding leading to the *semantic mismatch problem* [75] that the same term can refer to different concepts, e.g., bank of river vs. bank in finance. Thus, we ask the question of how to automatically decompose the PLM-produced semantic space of a token into its "senses"? Or, more generally, its aspects.

Contemporaneously, ColBERTv2 [217] studies the semantic space of ColBERT and finds that token representations produced by ColBERT localize in a small number of regions corresponding to the contextual "senses" of a token. Hence, this semantic space can be summarized, with high precision, by a set of static embeddings (i.e., cluster centroids) along with minor refinements at the dimension level to better approximate the token's contextualized representation. Despite the success of this clustering-based technique, it is not end-to-end differentiable.

In this contribution, we are interested in studying how well simpler approaches can approximate the contextualization process of transformer-based PLMs for ranking, by answering the following research questions:

— How to automatically learn static token sub-embeddings which decompose the semantic space of the token?

— How to aggregate these token sub-embeddings to approximate the oracle contextualized representation?

3 *Distilling the Oracle Contextualization Process*

Our proposed approach examines the possibility of constraining the contextualization process in PLMs, while maintaining their high ranking-effectiveness. Our goal is to better understand the role of contextualization for the ranking task, rather than to propose a new efficient approach for ranking. We view

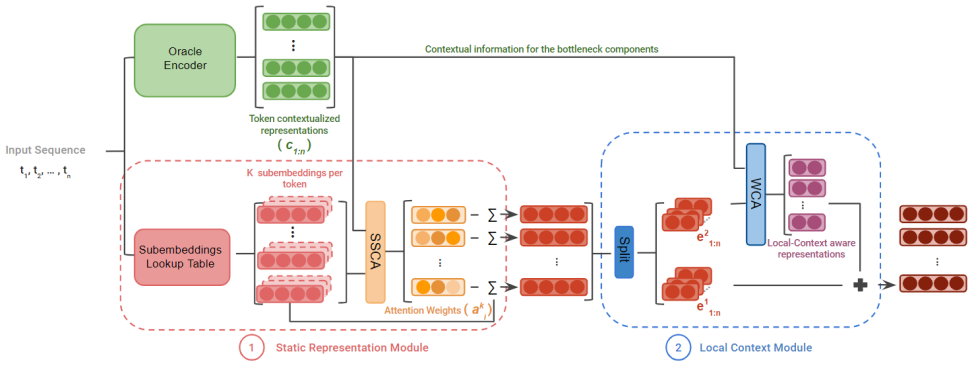


Figure 8.1 – ① SRM combines $K = 3$ token sub-embeddings using attention (SSCA) weights to produce token representations. ② LCM uses windowed attention (WCA) to integrate local context into the SRM representations.

this analogously to work probing PLMs [209]. While probes are classifiers that typically can perform some classification task, the point of the probe is to gain insight into the PLM rather than to propose a new way to perform the classification task. This a primary study which, we hope can motivate more efficient designs of contextualized ranking models.

In order to study the complex contextualization process in PLMs, we use distillation into simpler modules based on intrinsic and extrinsic aggregation methods. This simpler modules are designed to generate contextualized representations approximating those obtained by the studied PLM, referred to as the oracle, using *bottlenecked contextual information* from the oracle as guidance.

More formally, given an input sequence $T = t_{1:n}$ of n tokens, we use the oracle PLM to produce contextualized representations $c_{1:n}$ for each token $t_i \in T$. These oracle representations will serve as guidance for our modules. We present the same input sequence T to our module, or the first module in a combination of modules to produce contextualized representations for each token $t_i \in T$. As illustrated in Figure 8.1, each module integrates an information bottleneck to access the oracle representations $c_{1:n}$ to help decide the aggregation weights of the token's sub-embeddings in module ① (intrinsic aggregation), or the local context token representations in module ② (extrinsic aggregation). We detail the implementation of each of these modules in the following. Then we present the full life cycle of the resulting models including pre-training, fine-tuning and inference.

3.1 *Aggregation Methods*

We first design the *static representation module* to investigate the performance of intrinsic aggregation combining a small number of K static token sub-

embeddings to represent different occurrences of the same token. This intrinsic approach to token representation can capture different signals closely tied to a token, such as its senses or commonly co-occurring topics. We capture these signals and automatically learn static sub-embeddings during pretraining (Section 3.2.2) through distillation from the oracle.

Intrinsic aggregation of token sub-embeddings builds on the observation that a BERT encoder produces a lightweight semantic space, in which vectors corresponding to different aspects or contextual "senses" of a token gather in the same region [217]. It summarizes the entire semantic space of a token using combinations of a small number K of static sub-embeddings representing high-level contextual senses of the token, e.g., left as the past form of the verb to leave vs. left as in left/right directions vs. political left. While $K = 1$ is equivalent to learning classic static token embeddings (i.e., one per token), the approach can become as flexible as a PLM's contextualization process with a large enough value of K where each sub-embedding can correspond to one contextualized representation of the token (though it would be difficult to train and expensive to store).

On the other hand, we design the *local context module* to investigate the impact of extrinsic refinements of token representations produced by the static representation module when considering a small context window. Considering the summarization realized during the intrinsic aggregation, fine-grained topic information (e.g., "bank robberies" vs. "deposit money in the bank") tied to the same general topic of a token (interacting with a bank in the financial sense) have low chances of being encoded in the token's intrinsic representation. Hence, we investigate if missing fine-grained variation of a token representation can be captured using a reduced context window (e.g., one token to the left/right).

3.1.1 *Static Representation Module (SRM)*

The Static Representation Module uses a lightweight architecture which models a token representation as a combination of its sub-embeddings as illustrated in Figure 8.1 as module ①.

The SRM learns a static lookup-table containing sub-embeddings for each token in the oracle's vocabularly V, reducing its contextual space into a set of K static sub-embeddings per token (i.e., $K \times V$ sub-embeddings). To generate a token representation, the token's sub-embeddings are combined into a single representation with a weighted average, where weights are computed based on the bottlenecked contextual information from the oracle.

Submbeddings Lookup Table (SLT). SRM builds a set of K static sub-embeddings representing the different "*contextual senses*" of a term. Considering an input sequence $T = t_{1:n}$ of n tokens, we lookup the K sub-embeddings

$s_i^{1:K} \in \mathbb{R}^{K \times D}$ (D being the dimension of the sub-emebddings) in SLT for each token $t_i \in T$.

Single Score Cross Attention (SSCA). Constitutes a controlled information bottleneck allowing contextual information of tokens from the oracle to select more indicative sub-embeddings automatically. Formally, we use the scaled dot-product attention weights by using the token contextualized representation c_i from the oracle for the query vector and each token sub-embedding s_i^k for the key vectors as follows:

$$(a_h)_i^k = \underbrace{softmax\left(\frac{(c_i.W_h^{Query})(s_i^k.W_h^{Key})}{\sqrt{D_{Key}}}\right)}_{ATT_h(c_i, s_i^k)} \tag{8.1}$$

where W_h^{Query} and W_h^{Key} represent the query and key layers of the attention head h, respectively, and D_{Key} is the dimension of the key vector.

We max pool over the heads to output a single attention weight per token sub-embedding:

$$a_i^k = \max_{h \in H} (a_h)_i^k, k \in \{1..K\} \tag{8.2}$$

Finally, the token sub-embeddings are combined using the corresponding attention weights to produce the output representation:

$$SRM(t_i) = \sum_{k=1}^{K} a_i^k \cdot s_i^k \tag{8.3}$$

3.1.2 *Local Context Module (LCM)*

To complement the SRM's intrinsic sub-embedding aggregation, the Local Context Module performs extrinsic aggregation by using nearby tokens for refining the SRM's representations through a bottlenecked Windowed Cross-Attention (WCA).

Formally, given a token $t_i \in T$, and its SRM output embedding e_i, WCA applies cross-attention on the context window of size ws around t_i. The contextualized representation c_i of the the center token t_i is used for the query while the embeddings $e_{i-ws:i+ws}$ of the context tokens are used for the key and value.

$$(z_h)_i = ATT_h(c_i, e_j) \cdot (e_j \cdot W_h^{Value}) \tag{8.4}$$

with $j \in \{i - ws : i + ws\}$, and W_h^{Value} being the value layer of the attention head h.

The center token's refined representation is obtained by concatenating the results from all attention heads and projecting through the output layer:

$$WCA(e_i) = concat_{h=1}^{H}(z_h)_i \cdot W^{Output} \tag{8.5}$$

Finally, we formulate the output of LCM as a gating function, and find it effective to split the input token embedding into two independent parts $(e_i^1, e_i^2) \in \mathbb{R}^{D/2}$ along the channel dimension D:

$$LCM(e_i) = (e_i^1 \oplus WCA(e_i^2)) \cdot W^{Gate} \tag{8.6}$$

where $\oplus$ denotes concatenation, and $W^{Gate} \in \mathbb{R}^{D \times D}$ is a feed-forward layer. The LCM's architecture is illustrated in module ② in Figure 8.1.

At the end, the LCM resembles a transformer with a reduced local context instead of the full context. Nevertheless, our module is different from a normal transformer architecture, LCM implementation was inspired by Liu et al. [134] work exploring representation-independent gated MLPs as an alternative for the attention (representation-dependent). g-MLP focuses only on position when computing token-to-token importance, however we find this alternative less effective in pilot studies.

3.2 *Life Cycle*

In this section we describe our oracle model followed by a description of how our modules are pre-trained and then fine-tuned for ranking.

3.2.1 *Oracle Model*

ColBERT's [112] state-of-the-art relevance scoring is performed by soft matching all query and document token's contextualized representations. Comparing its effectiveness with that of methods using static embeddings, it is clear that the contextualization process is incorporating signals that are highly effective for this task. Additionally, using token-to-token vector similarities offers a simple and interpretable scoring method, and an effective way for evaluating the quality of token-level vector representations for relevance matching.

As we have previously seen in Section 5.3, given a query q with n tokens and a document d comprising m tokens, ColBERT produces first contextualized representations $\Phi(q_{1:n})$ and $\Phi(d_{1:m})$ of the query and document tokens, respectively, using a BERT encoder. The relevance (similarity) score between the query and document is then computed using the *MaxSim* operation, which identifies the closest-matching document token for each query token:

$$R(d, q) \triangleq \sum_{i=1}^{n} \max_{j=1}^{m} \Phi(q_i) \cdot \Phi(d_j) \tag{8.7}$$

The original ColBERT model uses query expansion by appending [MASK] tokens to the end of the query, however because it is non-trivial to explain reliably what each [MASK] token stands for, we chose not to include this query expansion in our investigations.

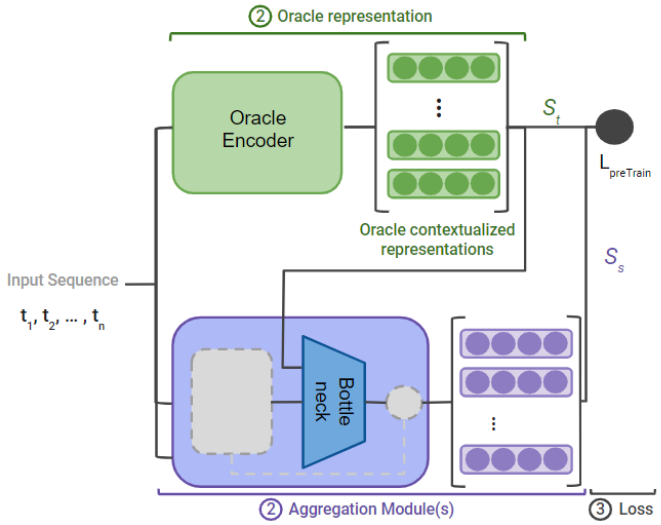


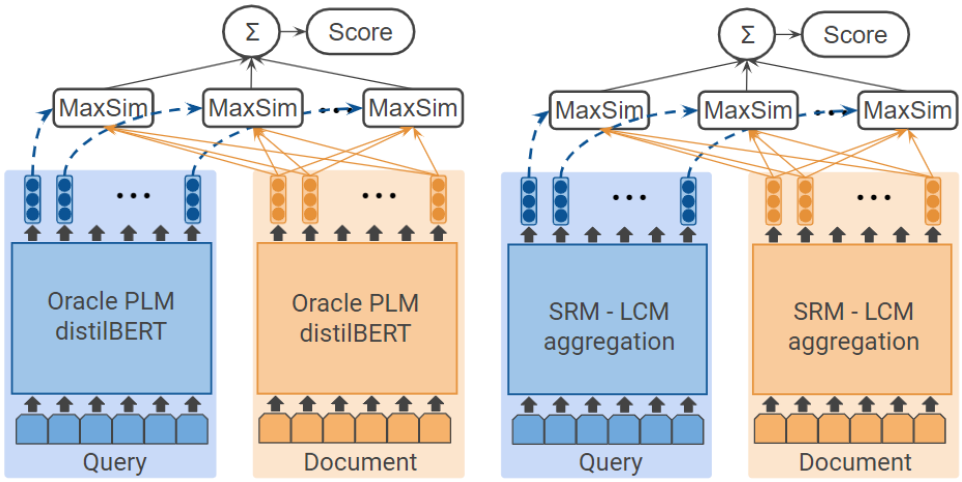
Figure 8.2 – The pre-training procedure of a module or combination of modules through distillation from the oracle PLM encoder using the MSE loss in Eq.8.8. The weights of the module(s) are randomly initialized and the oracle encoder weights are not updated.

For our oracle PLM, we use the trained encoder in a ColBERT variant in which the encoder is a distilBERT model [216]. This variant matches the original ColBERT's effectiveness while reducing its size (i.e., distilBERT with only 6 transformer layers compared to BERT with 12 layers) thanks to distillation from a teacher ensemble coupled with the *MarginMSE* loss [88]; see Section 6.2.

In our approach, the oracle plays two roles. First, the oracle serves as the teacher model for distillation into our modules in the pre-training phase. Second, the contextualized token representations produced by the oracle are used as guidance for the aggregation mechanisms at the level of the bottleneck components in both SRM and LCM (i.e., in the attention mechanisms). This can be viewed as a preprocessing step where the contextualized embeddings are replaced with the attention weights from the SRM and LCM. For example, the SRM's output can be computed using only each token ID and its K attention weights. The LCM's output can be computed similarly.

3.2.2 *Pre-training*

We pre-train our SRM and LCM modules from scratch using distillation from the oracle contextualized representations. To do so, we use single text sequences as input, and randomly sample N tokens from each sequence to use for pre-training. All parameters in the SRM and LCM modules are randomly initialized at the beginning of the training. We optimize these module parameters to minimize the Mean Squared Error (MSE) loss between



(a) Oracle based ranking model (b) SRM-LCM based ranking model

Figure 8.3 – Comparison between (a) the ColBERT ranking model which relies on the oracle PLM distilBERT for contextualization, and (b) our aggregation-based ranking model which relies on our specifically designed aggregation modules (SRM and LCM) for contextualization. Both models rely on ColBERT's late interaction mechanism which is based on the MaxSim operator

their output token representations S_s (i.e., student model output) and the equivalent oracle contextualized representations S_t produced by the oracle encoder (i.e., the teacher model) as shown in Figure 8.2:

$$L_{preTrain} = MSE(S_s, S_t) \tag{8.8}$$

3.2.3 *Fine-tuning and Ranking*

During pre-training, our models are only trained on single sequences to distill the contextualized representations of ColBERT's encoder. The purpose of this general-purpose pre-training is to build a generalizable static lookup sub-embeddings table that can be easily transferred to the downstream task. Therefore, we freeze the sub-embedding lookup table (SLT) after pre-training, and fine-tune the rest of the modules' parameters for the ranking task.

In order to deploy our modules for text ranking, we use ColBERT's *MaxSim* operator to compute relevance scores using the output representations produced by SRM or LCM. Using this similarity mechanism, we fine-tune our modules, analogously to how the oracle was fine-tuned for fair comparisons, with triples containing a query, a relevant passage, and a less relevant passage $\langle q, d^+, d^- \rangle$. We also use distillation from a teacher ensemble M_t using

the *MarginMSE* loss [88], which attempts to mimic the teacher's pairwise differences in passage scores:

$$L_{rank} = MSE(M_s(q, d^+) - M_s(q, d^-), M_t(q, d^+) - M_t(q, d^-)) \tag{8.9}$$

where M_t provides the teacher relevance score for both passages w.r.t query for our student model M_s.

After this fine-tuning, we have our aggregation-based ranking model which follows a ColBERT architecture in which the BERT encoder is replaced by our modules SRM and LCM as shown in Figure 8.3.

4 *Methodology and Experimental Setup*

We describe in this section the experimental setup we use for validating our simplified contextualization approach. In order to investigate how well simplified intrinsic and extrinsic aggregations perform, we proceed with an incremental validation of the modules presented above (see Section 3.1). Our objectives through this evaluation are as follows:

1. Investigate how well intrinsic aggregation of static token sub-embeddings can model the oracle contextualized representations;
2. Study the contribution of extrinsic information from the local context to the token representation;
3. Explore the complementarity of intrinsic and extrinsic aggregation approaches;
4. Study the zero-shot generalizability to out-of-domain collections.

4.1 *Experimental Setup*

4.1.1 *Test Collections*

We consider the following standard passage ranking benchmarks to achieve the above experimental objectives:

— MS MARCO passage ranking development collection [9];

— TREC Deep Learning passage ranking tracks from 2019 and 2020 [45, 44];

— DL-HARD passage benchmark [151].

These test collections are standard in the dense retrieval literature [110, 112, 69]. In addition to these common test collections based on the MS MARCO passage corpus which we use for fine-tuning, we further evaluate our modules in a zero-shot transfer setting on the following out-of-distribution data collections:

— TripClick [200];
— TREC Robust04 [1].

MS MARCO This collection consists of about 500k training queries and 6,980 development queries, which are both sparsely judged. The evaluation set is private, so we follow common practice and report results on the development queries (MS MARCO Dev). The performance of models is measured using the Mean Reciprocal Rank, MRR@10, metric. We use Anserini's [255] implementation of BM25 with default parameters to retrieve the top-1000 candidate passages for reranking.

TREC DEEP LEARNING PASSAGE RANKING We consider the densely-judged query sets of 43 and 54 queries from the TREC Deep Learning (DL) passage reranking tracks of 2019 (DL'19) [45] and 2020 (DL'20) [44]. Unlike MS MARCO Dev, there are more passages annotated per query, and the relevance judgements are graded (instead of binary judgements), allowing to use the more informative nDCG@10 metric. We rerank the official organizers BM25 runs.

DL-HARD We include experiments on the DL-Hard passage benchmark, focusing on 50 challenging and complex queries partially from DL'19 and '20, by reranking the authors' BM25 run baseline.

TRIPCLICK A collection of click log data from the health search engine *Trip Database*. It contains 1.5M passages, and 3,525 test queries distributed into three query sets with 1,175 queries each, namely Head, Torso, and Tail queries, grouped by their frequency.

We rerank the top 200 candidate passages retrieved by the BM25 implementation in Anserini [255] with default parameters.

ROBUST04 A newswire collection comprising 500K long documents (TREC Disks 4 and 5) and 249 judged topics. We use the standard "title" keyword queries.

Since the documents are longer than the length limit of ColBERT, they are split into passages using a sliding window of 250 tokens, and only consider the up to the first 8 passages per document. The maximum passage sequence length is set to 256, and the maximum query sequence length is set to 50. A document relevance score is given by its best scoring passage.

We use BM25 with RM3 query expansion [256] as first-stage retriever as implemented in Anserini with the default parameters.

1. https://trec.nist.gov/data/robust/04.guidelines.html

4.1.2 *Pre-training*

We use more than $20M$ passages extracted from the TREC-CAR collection [59] to pre-train our models following the procedure described in Section 3.2.2. This collection is drawn from Wikipedia pages offering a diversity of contents ideal for learning generalizable static sub-embeddings. In addition to quantity and diversity, Wikipedia was one of the corpora used for pre-training BERT.

We use the ADAM optimizer [113] with a learning rate of $2e-3$. We train for 10 epochs with a batch size of $4,096$ passages of maximum 128 tokens, and considering $N = 50$ randomly sampled tokens per passage. We utilize the full 768 dimensions of the oracle encoder for a fair comparison.

4.1.3 *Fine-tuning*

For ranking, we fine-tune the models with a learning rate of $2e-5$ on the MS MARCO train set for 30 epochs with a batch size of 32 triples comprising a query, a relevant passage, and a less relevant passage. We use the Margin-MSE loss as described in Section 3.2.

4.1.4 *Inference*

Given a query and a list of passage candidates, our task is to rerank the passages according to their relevance to the query using the late-interaction mechanism. We consider reranking as a fair setting where the candidate documents are the same for all models. While we could directly use ANN (see Section 6.3) with our modules, similarly to ColBERT, or devise an optimization that takes advantages of our modules (i.e., with the SRM, each representation is simply a weighted sum of K static sub-embeddings), we focus on this reranking setting in order to simplify experimentation by avoiding the costly step of creating ANN indexes.

4.1.5 *Evaluation Metrics*

We evaluate our modular approach by adopting the TREC protocol. For this purpose, we use the official metric used in the context of the in-domain collections, that is MRR@10 for MS MARCO Dev and nDCG@10 for TREC DL passage ranking and DL-HARD collections. For the out-of-domain evaluation, we report nDCG@20 and P@20 for Robust04, and MRR@10 and nDCG@10 for TripClick to allow straightforward comparisons with previously reported results in the literature.

All performance measures are obtained by evaluating the document rankings (retrieval runs) using the official `trec_eval`[2] tool from TREC. It is the

2. http://trec.nist.gov/trec_eval/

tool used by the TREC community to evaluate the performance of track submissions.

4.2 *Baseline and Evaluation Scenarios*

We consider a ColBERT architecture based model as our oracle for distillation into the intrinsic and extrinsic aggregation modules. We use the ColBERT checkpoint made publicly available by Hofstätter et al. [88] on Hugginface hub [3]. This ColBERT model represents the baseline for our investigation of the contextualized process for soft matching in the context of text ranking.

To investigate the impact of each factor introduced in our modules, we distill the contextualization process in ColBERT's encoder into different variants of our SRM and LCM modules. We detail these variants in the following.

SRM-Avg This variant of SRM replaces the attention mechanism SSCA with a simple average pooling of the token sub-embeddings. This variant creates a single embedding per token.

SRM-Top1 Considering that each sub-embedding captures a distinct aspect (i.e., sense, topic, etc) of the token, we represent a token with its closest aspect embedding instead of a mixture of its different aspect embeddings. This variant of SRM represents a token by its most indicative aspect, i.e., the sub-embedding with the highest attention weight is used to represent the token. To do so, SRM-Top1 is trained from scratch using *Gumbel Softmax* [152, 101] instead of Softmax in SCCA (Eq.8.1), to approximate argmax gradients in the backward pass. For training stability, we find it more effective to initialize the token sub-embeddings from the centroids produced by K-means on a sample of contextualized representations.

SRM-LCM This represents the model resulting from the combination of the SRM and LCM modules, where the output of the intrinsic aggregation from SRM is fed through LCM for local context refinement. We also experiment with the SRM-Top1 variant which gives the SRM-Top1-LCM combination.

5 *Results and Analysis*

We incrementally evaluate our modules beginning with in-domain evaluations to investigate the intrinsic aggregation approach implemented in SRM; followed by the investigation of the impact of the extrinsic refinement considering local context information implemented in LCM; and then study their complementarity; finally we explore the zero-shot generalizability to out-of-domain collections.

3. https://huggingface.co/sebastian-hofstaetter/colbert-distilbert-margin_mse-T2-msmarco

Table 8.1 – SRM reranking effectiveness on the MS MARCO Dev set with variable number of token sub-embeddings K. D is the embedding size (768). Our module's best results are in **bold**, and oracle results are underlined when overall best.

	K	# SLT parameters ($\times 10^6$)	MRR@10
BM25	-	-	0.184
ColBERT	-	-	0.342
SRM	1	$0.03 * D$	0.218
	5	$0.15 * D$	0.317
	10	$0.30 * D$	0.330
	15	$0.46 * D$	**0.332**

5.1 *Intrinsic Aggregation*

First, we investigate to what extent the contextual (semantic) space of ColBERT can be distilled into a finite set of static token sub-embeddings. We examine the reranking effectiveness of SRM which, given a token, determines automatically its most indicative sub-embeddings, and combines them to produce the output representation. To better understand the source of SRM effectiveness, we examine details in its encoding process to study:

RQ1. *How well can contextualized representations be modeled as an intrinsic aggregation of static token sub-embeddings?*

Impact of the number of token sub-embeddings. We first ask: what is the optimal number of sub-embeddings required for an accurate representation of a token's semantic space? To address this question, we distill ColBERT's encoding process into multiple copies of SRM with a variable number K of sub-embeddings per token, ranging from using a single static embedding per token, up to $K = 15$ sub-embeddings. We report the ranking performance on the MS MARCO Dev collection and the resulting number of parameters in the SLT for each K in Table 8.1.

As the results show, using token intrinsic aggregation of static sub-embeddings, SRM can retain up to 97% of the oracle's performance. Compared to the initial bag-of-words BM25 retriever, even using a single static embedding ($K = 1$) to represent all token occurrences is more effective. Using increasing numbers of sub-embeddings leads to better performance up to $K = 10$, where the performance of SRM stabilizes. SRM with $K = 15$ brings no significant gains over $K = 10$ while requiring 50% more parameters, hence we use $K = 10$ for the rest of our experiments.

Sub-embeddings Aggregation Mechanism. Subsequently, we ask if our SSCA mechanism is better than other simpler alternatives. For this investiga-

Table 8.2 – Ranking effectiveness of SRM with different aggregation mechanisms on all datasets.

	Dev	DL'19	DL'20	DL-Hard
	MRR@10	nDCG@10	nDCG@10	nDCG@10
[1] BM25	0.184	0.506	0.480	0.304
[2] ColBERT	0.342	0.713	0.699	0.394
[3] SRM	**0.330**	**0.707**	**0.682**	**0.382**
[4] SRM-Avg	0.192	0.446	0.446	0.238
[5] SRM-Top1	0.280	0.605	0.574	0.302
[6] SRM-K1	0.218	0.500	0.517	0.270

tion, we report ranking performance using $K = 10$ on both the MS MARCO Dev set and the DL query sets in Table 8.2, which shows our main SRM configuration with SSCA in run [3].

We evaluate the SRM-Avg variant with simple average pooling of the token sub-embeddings. As the results show in run [4], assigning the same weight to all sub-embeddings achieves low performance and confirms the importance of SSCA, which assigns non-uniform weights to sub-embeddings.

For the SRM-Top1 variant, the results in run [5] suggest that representing a token with its most-indicative static sub-embedding can achieve better ranking performance than using a single static embedding for all token occurrences, as shown in run [6]. Nevertheless, considering a small number of only $K = 10$ sub-embeddings cannot possibly model all the fine-grained aspects of a token accurately, and further refining is required to match the oracle's performance.

5.2 *Extrinsic Aggregation*

While SRM focuses on intrinsic token representations, LCM extends token representations to include information from nearby tokens to study:

RQ2. *How can a token representation be refined using extrinsic information from its local context?*

Table 8.3 reports the results of refining SRM representations with LCM using different context window lengths. Interestingly, by considering only the direct neighbours of a token ($ws = 1$) in LCM, the refined representations already match ColBERT's effectiveness on MS MARCO Dev. Considering that we use the same WordPiece tokenizer as in ColBERT, we can hypothesize that with $ws = 1$ the LCM is possibly aggregating full-word representations from its sub-tokens (i.e., word pieces). Further enlarging the context window leads to comparable performance with the oracle.

Table 8.3 – Ranking effectiveness of SRM-LCM with different context window lengths(ws) on MS MARCO Dev and DL query sets. TREC-Best reports the best DL submitted runs. Our module's best results are in **bold**, and oracle results are underlined when overall best.

	ws	Dev	DL'19	DL'20	DL-Hard
		MRR@10	nDCG@10	nDCG@10	nDCG@10
BM25	-	0.184	0.506	0.480	0.304
TREC-Best (no ensembles)	all	-	0.731	0.746	0.408
ColBERT	all	0.342	0.713	0.699	0.394
SRM	0	0.330	0.707	0.682	0.382
SRM-LCM	1	**0.343**	0.721	0.721	0.369
	2	0.341	0.723	0.717	0.406
	3	0.341	0.715	0.713	0.407
	4	0.342	0.719	0.723	0.387
	5	0.342	**0.728**	**0.727**	**0.409**
	all	0.337	0.717	0.707	0.382

On TREC DL'19 and DL'20, we notice consistent improvements over the oracle effectiveness with slight variations due to the window size. When we focus on the challenging queries in DL-Hard, SRM (with $ws = 0$) achieves better ranking performance than SRM-LCM with $ws = 1$. Nevertheless, increasing the local context window is beneficial for these complex queries, and outperforms the oracle. With a local context window of only $ws = 3$, SRM-LCM performs as well as the TREC-Best run. Nonetheless, using a global context window encompassing all sequence tokens [4] hurts the performance, suggesting that local context is more informative.

LCM is a simpler implementation of a transformer using a very restrained local context in a single attention layer. Nevertheless, it performs well when combined with the SRM. This is interesting as it shows that SRM combined with minimal contextualization is sufficient to match the oracle.

5.3 *Intrinsic-extrinsic complementarity*

The results from Table 8.3 indicate that the combination of intrinsic and extrinsic approaches in SRM-LCM reaches the oracle ranking performance despite its simplified contextualization process. To better understand the sources of this effectiveness, we examine the importance of the intrinsic vs. extrinsic aggregation to answer:

4. Special tokens are masked, i.e., [CLS] and [SEP].

Table 8.4 – Ranking effectiveness of LCM extrinsic refinement applied to SRM variants, on MS MARCO Dev and DL sets. Our module's best results are in **bold**, and oracle results are underlined when overall best.

	Dev	DL'19	DL'20	DL-Hard
	MRR@10	nDCG@10	nDCG@10	nDCG@10
[1] BM25	0.184	0.506	0.480	0.304
[2] ColBERT	0.342	0.713	0.699	0.394
[3a] SRM	0.330	0.707	0.682	0.382
[3b] SRM-LCM	**0.343**	**0.721**	**0.721**	0.369
[4a] SRM-K1	0.218	0.500	0.517	0.270
[4b] SRM-K1-LCM	0.336	0.705	0.710	**0.383**
[5a] SRM-Top1	0.280	0.605	0.574	0.302
[5b] SRM-Top1-LCM	0.324	0.698	0.669	0.377

RQ3. *Are the intrinsic and extrinsic approaches (SRM and LCM, respectively) complementary?*

For this study, we report the ranking performance on both Dev and DL query sets in Table 8.4, which shows the main SRM-LCM model using $ws = 1$, run [3b], with MRR@10 of 34.3%.

To begin with, we ask if the intrinsic sub-embeddings aggregation can be replaced with LCM extrinsic aggregation. Model [4b] tackles this question by using a single static embedding per token ($K = 1$) with a window size of $ws = 1$. As the results show, extrinsic refinement provides large gains compared to SRM-K1 in run [4a], and performs as well as SRM in [3a] with $K = 10$. This suggests that sub-embedding combination alone is comparable to contextualizing a static embedding with a reduced attention window. Interestingly, these models [3a] and [4b] achieve better performance on DL-HARD queries compared to the full SRM-LCM [3b].

Next, we examine if LCM applied to SRM-Top1 can provide better results considering that SRM-Top1 [5a] performs better than SRM-K1 [4a]. Surprisingly, LCM refinement provides more important gains when applied on SRM-K1 outputs compared to SRM-Top1 outputs as shown in lines ([4b] vs. [5b]). This is probably due to the instability of the Gumbel Softmax approximation in SRM-Top1-LCM.

5.4 *Case Study*

Table 8.5 shows how our intrinsic and extrinsic approaches see token similarity across different contexts compared to the oracle. We report results

Table 8.5 – Sample query-passage token matches from the MS MARCO passage collection.

Query	Module(s)	Top matching sampled tokens				
pain in **right** arm	ColBERT	right (14.8)	left (11.0)	west (8.5)	upper (8.3)	straight (8.3)
	SRM	right (10.7)	left (7.4)	rights (6.6)	north (6.1)	west (5.6)
	SRM-LCM	right (12.8)	left (9.4)	west (7.4)	straight (7.2)	wrong (7.2)
right to own arms	ColBERT	right (14.1)	rights (11.7)	freedom (8.7)	power (8.5)	free (8.5)
	SRM	right (9.8)	rights (8.6)	free (5.5)	liberty (5.5)	freedom (5.5)
	SRM-LCM	right (11.4)	rights (10.2)	freedom (7.9)	liberty (7.6)	freedoms (7.5)
operating **system**	ColBERT	system (15.2)	systems (13.5)	pc (10.5)	computer (10.4)	server (10.1)
	SRM	system (10.7)	systems (9.4)	computer (6.9)	software (6.7)	unix (6.7)
	SRM-LCM	system (12.4)	systems (12.1)	unix (9.2)	linux (9.0)	software (9.0)
nervous **system**	ColBERT	system (15.2)	systems (13.4)	nervous (9.4)	brain (9.3)	tract (9.2)
	SRM	system (9.6)	systems (8.7)	computer (6.4)	unix (6.2)	linux (6.1)
	SRM-LCM	system (13.5)	systems (11.5)	nervous (9.0)	peripheral (9.0)	central (8.7)

Table 8.6 – Ranking effectiveness of SRM-LCM ($K = 10$ and $ws = 1$) on TripClick and Robust04 test collections. Best results are indicated in **bold**.

	TripClick Head		TripClick Torso		TripClick Tail		Robust04	
	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	P@20	nDCG@20
Initial retriever	0.301	0.149	0.305	0.224	0.263	0.285	0.397	0.451
ColBERT	0.480	0.164	0.395	0.233	0.326	0.271	**0.397**	**0.458**
SRM-LCM	**0.510**	**0.169**	**0.400**	**0.240**	**0.329**	**0.283**	0.379	0.441

using SRM with $K = 10$ and SRM-LCM with $ws = 1$. Term definitions are taken from the Cambridge English Dictionary. Representations are sampled.

The first query searches for "right" in the sense of direction [5]. The SRM is able to soft match the tokens related to direction like "left" or "north", but it also matches "rights" that is another sense of the same token. Adding local context information (LCM) increases the similarity to tokens like "right" or "west" and removes the strong matching to "rights". The modules share 4/5 tokens with the oracle with the same ranking. The second query uses "right" in the sense of legal [6]. Both SRM and the combination of SRM with LCM are able to distinguish the correct sense of the term and stay close to the oracle. On the other hand, the queries related to "systems" show the importance of context to determine the right topical meaning induced by the surrounding context of non-polysemous terms. SRM matches both "operating system" and "nervous system" to computer related systems; this bias could be induced by the training data containing a significant number of occurrences of "system" in computer-oriented contexts. Adding the LCM makes a drastic improvement by matching words related to nerves (medical topic), demonstrating the utility of the local context.

5. Right (Direction): on or towards the side of your body that is to the east when you are facing north.

6. Right (Legal): a moral or legal entitlement to have or do something.

5.5 *Zero-shot generalizability to out-of-domain collections*

We report in Table 8.6 the zero-shot performance of the SRM-LCM combination, with $K = 10$ and $ws = 1$, compared to the oracle and the initial first-stage retriever to study:

RQ4. *How well can the SRM-LCM module combination generalize to out-of-domain collections in a zero-shot setting?*

First, we examine the generalizability to the medical collection on the three TripClick query sets. The results clearly show that the simple module combination exhibit same, and even better, zero-shot performance as the oracle across the three different query sets. Notably, SRM-LCM outperforms substantially the oracle on head queries in terms of MRR@10, and outperforms the oracle on torso and tail queries in terms of nDCG@10. This suggests that simpler contextualization processes can generalize as well as a PLM full-flexible contextualization.

Second, we study the performance on Robust04 newswire collection. Interestingly, the zero-shot performance of the oracle is on-par with the initial BM25 with RM3 expansion retriever, indicating the low zero-shot transfer capabilities of the ColBERT oracle. Our SRM-LCM combination also performs poorly in this configuration. The nature of the Robust04 short key-word queries could be challenging for models trained with MS MARCO question-like queries. On the other hand, as opposed to the MS MARCO and TripClick corpora, Robust04 documents are long and the best scoring passage is taken as proxy for the document-level relevance score.

At the end, our simpler contextualization process cannot only yield comparable or slightly better performance on in-domain MS MARCO passage ranking benchmarks, but can also exhibit on-par zero-shot performance with ColBERT's contextualization. Nevertheless, our proposed aggregations show lower zero-shot performance on Robust04 long document ranking benchmark.

6 *Discussion and Conclusion*

Contextualized representations have been widely adopted for their soft matching effectiveness in the context of ranking, especially in the emerging dense retrieval area in which models are representation-focused. However, prior work suggests that they can be replaced, to some extent, by static embeddings.

In this chapter, we investigated how well the complex contextualization process in PLMs can be distilled to simplified aggregation methods based on static embeddings, in the context of ranking. We propose an intrinsic

aggregation module combining a small number of static sub-embeddings which capture different aspects closely tied to a token such as its senses or commonly recurrent topics. Empirical results indicate that distilling the entire oracle-produced semantic space of a token into a finite set of static token sub-embeddings with the intrinsic aggregation can retain up to 97% of the oracle ranking performance on the MS MARCO Dev set.

To better model the oracle semantic space, we further propose refining the intrinsic aggregation module with additional information from local context. The extrinsic aggregation module can capture the fine-grained topic information from surrounding tokens which have low chances of being modeled by the token sub-embeddings. Experiments show that the intrinsic and extrinsic aggregations are complementary and can perform as well as the oracle, and even outperform it on Trec Deep Learning densely-judged queries.

Empirical evaluations on out-of-domain collections demonstrate that the combination of intrinsic and extrinsic aggregations can additionally exhibit interesting zero-shot generalization abilities notably to the TripClick medical test collection.

Overall, our results suggest that text ranking does not necessarily require the complexity of a PLM contextualization process. Though we do not propose a more efficient alternative for transformer models, this study is a preliminary investigation of the contextualization process in PLMs for ranking. Our observations can motivate custom transformer designs for the ranking task, which can leverage static embeddings or local window attention mechanisms. Finally, distillation from PLMs can help pre-train these custom ranking models for representation learning instead of directly training them from scratch for ranking (e.g., the TK model discussed in Section 5.2 rethinks transformers for ranking but loses the benefits of pre-training).

Conclusion

Conclusion and Future Work

Contributions Overview

In this dissertation, we focused on the core task of information retrieval (IR), ad hoc ranking. We were particularly interested in neural IR approaches, which rely on deep neural networks to perform ad hoc search effectively.

The work covered in this dissertation came during a paradigm shift in ad hoc search. Specifically, the advent of contextualized language models, of which BERT [58] is the best-known example, has shown promising potential for building much more effective neural ranking models while also introducing new challenges. Work in this dissertation provided explorations in various directions for adapting these powerful models, stemming from the closely related NLP field, to the ad hoc ranking task.

The big leap in ranking effectiveness achieved by the simple application of BERT was unprecedented and led to immediate excitement in the IR community. Besides, the pre-train then fine-tune recipe, and the availability of large-scale collections facilitated the rapid widespread of BERT, which has soon come to dominate the research landscape. The state-of-the-art overview part of this dissertation is hence focused on this family of neural approaches. We presented an overview of BERT, which is the most adopted instance of transformer models. We describe its architecture and present the different innovations it integrated. Three chapters were devoted to covering the different methods proposed in the literature these past four years to apply BERT and its subsequent variants to ranking:

1. BERT in multi-stage reranking, Chapter 4 focuses on the early and most straightforward applications of BERT for reranking in multi-stage architectures;
2. BERT for sparse retrieval, Chapter 5 covers applications of BERT for query and document expansion to mitigate the vocabulary mismatch problem in sparse retrieval while benefiting from the efficiency of inverted indexes;
3. BERT for dense retrieval, Chapter 6 presents models which directly learn dense representations through BERT for retrieval, which allows soft semantic matching to address the vocabulary mismatch problem.

Our contributions in the context of neural IR examined different leads for adapting contextualized pre-trained language models (PLMs) to the specific task of ad hoc ranking. Considering the unique characteristics of the

ranking task and the domain knowledge acquired throughout the decades of progress in IR, we believe that this knowledge can interact with PLMs to build better ranking models. But what do we mean by better ranking models? Are we seeking more effectiveness without special considerations regarding efficiency? Or alternatively, are we trying to balance effectiveness and efficiency for more practical models?

The two directions are promising, and we provide, in this dissertation, leads forward in both directions: In our early work, we focused on adapting PLMs by integrating an important traditional IR cue, exact matches, to improve effectiveness. Instead of rethinking the design of BERT for this purpose, we instead use the flexibility of its transformer architecture to our advantage. We propose manipulating the input template (i.e., the organization of the query document text) to the model and benefit from the special tokens convention to introduce marker tokens to convey explicit exact match signals. BERT's flexibility makes it possible to learn how to exploit this explicit signal through fine-tuning. Moreover, this approach is easily transferable to the subsequent variants of BERT, such as ELECTRA [40] and takes advantage of their improved pre-training. In later work, we were instead interested in the possibility of restraining the flexibility of transformer models to potentially achieve better effectiveness/efficiency trade-offs. We proposed using distillation and static embeddings in combination with information bottlenecks and carefully-designed modules to analyze the role of the contextualization process for the ranking task. This is analogous to work probing PLMs [209], where the point of the probe is to gain insight into the studied PLM. The finality of our study is to provide insights about contextualization in the context of ranking, which could motivate more efficient models tailored specifically for ranking by stripping away unnecessary signals.

The context of our works and their contributions can be summarized as follows:

1. In the months following the introduction of BERT, a plethora of work reported successful applications to text ranking [149, 2, 257, 51], as well as a wide variety of other tasks by simply fine-tuning the same homogeneous transformer architecture on the target task. Typically, the ad hoc ranking task is formalized as a matching problem between two texts in most proposed BERT-based models and treated as equivalent to many NLP tasks such as paraphrase identification, sharing even the same architecture. However, the ranking task is mainly about *relevance matching* while most NLP matching tasks concern *semantic matching*, and there are some fundamental differences between these two matching tasks [83]. Notably, pre-BERT studies [83, 165] argue that successful relevance matching requires proper handling of both exact and semantic matching signals. Building on this insight and seeing how BERT (and its variants) excels at semantic matching [194], we hypothesize that

incorporating traditional exact matching cues with BERT's inherent semantic matching can be favorable. We, thus, investigate the following question: *What is the impact of emphasizing exact match signals in PLMs for document ranking?*

The proposed approach aims to adapt the flexible BERT model to the task of document ranking by integrating the exact match intuition used in traditional IR models. Our approach relies on marking strategies that strategically introduce special marker tokens in the textual input of a PLM model to emphasize the overlapping terms between the query and document [21, 22]. We propose different marking strategies using different choices of marker tokens (i.e., reusing existing tokens in the vocabulary vs. introducing novel tokens) and marking levels (i.e., marking the document text only vs. marking both the query and document texts). Because this approach was developed in the context of reranking applications of BERT (Chapter 4), we adopt the cross-encoder architecture where the query and document are fed together through BERT. We follow the widely adopted monoBERT configuration [173] at the time and use the output of the [CLS] token for relevance classification. In other words, we rely on the flexibility of BERT's transformer architecture to learn how to use the exact match hints conveyed by the marker tokens for relevance matching, as it does for other special markers such as [CLS] or [SEP]. This method follows BERT's convention of using special tokens. Moreover, it avoids any superfluous parameters [117] added by new architectural components or modifications to the transformer, which would cost the benefits from Google's pre-training.

This contribution was thoroughly evaluated across different experimental scenarios, including in-domain and out-of-domain test collections. The results showed the effectiveness of our approach and demonstrated the importance of exact matching for relevance matching. On the standard TREC Robust04 and GOV2 ad hoc benchmarks, the impact of exact match marking is more prominent for long natural descriptions compared to short keyword queries as opposed to traditional retrieval models. This disparity inspired a hybrid pipeline, in which each stage is presented with its preferred query formulation, thus maximizing the end effectiveness of the multi-stage model. On the other hand, the gains were more substantial when the model was confronted with out-of-distribution data (i.e., not seen or different from the training data) compared to in-domain data. This suggests that when BERT is adequately fine-tuned for the target domain, its semantic space is representative enough to capture strong relevance matching signals. Nonetheless, when in-domain data is unavailable (i.e., zero-shot setting) or in fewer quantities (i.e., multi-phase fine-tuning), exact matching

cues can complement the semantic matching signals and enable better ranking effectiveness.

Aside from exact matching, we have also investigated another use case for marker tokens, query expansion. The experimental results confirmed the flexibility of our marker tokens since they can be used for implicitly learning query expansion representations.

2. Later developments moved away from the effective but expensive cross-encoder design towards a more efficient design named bi-encoders [98]. While both query and document are fed together to cross-encoders in a template input, the query and the document are encoded separately in bi-encoders. This independent encoding allows offline computation of document representations. The success of these representation-focused models (covered in Chapter 6) suggests that contextualized representations produced by BERT can encode signals important for relevance matching. Nevertheless, the transformer-based contextualization process in BERT is complex and opaque. Studies have shown that it integrates diverse information [209] and can learn highly transferable properties of language [135], making it a highly versatile encoder. Though the contextualization process in BERT is highly effective for ranking, seeing how it is flexible and versatile, we cannot but wonder: *Do we need the flexibility of the complex contextualization process of PLMs for document ranking or can we simplify it?*

   In order to answer this question, we propose distilling the contextualization process used in a ColBERT [112] model trained for ranking, which we refer to as the oracle, into simpler contextualization methods. This approach aims to investigate whether representations generated by a simpler contextualization process can perform as well as the oracle representations on the ranking task. In our study, we consider two assumptions about the semantic space of each token: (1) building on Santhanam et al. [217] observation that representations corresponding to different "contextual senses" of a token cluster closely in the semantic space produced by ColBERT, we consider that aspects closely tied to the token such as its polysemous senses can be captured in a finite set of static sub-embeddings; (2) summarizing the semantic space of a token by a small number of static sub-embeddings is unlikely to capture fine-grained variations in the meaning of the token which are due to context. To handle these two aspects, we proceed as follows:

   a) We proposed an intrinsic aggregation approach that models token representations as a combination of a finite number K of static sub-embeddings capturing the coarse-grained aspects of the token. For instance, bank in finance vs. river bank.

      We implement this aggregation in an independent static representation module (SRM). For a given token, SRM produces its

representation using a weighted sum of its static sub-embeddings. Each sub-embedding is weighted according to its significance to the context, which is determined by an attention mechanism using the token contextualized representation –produced by the oracle– accessed via an information bottleneck.

b) We proposed an extrinsic aggregation approach that refines a token representation using information from its local context to capture fine-grained variations due to context in the token's representation—for instance, robbing a bank vs. depositing money at a bank.

   Following a modular architecture that allows us to analyze the impact of each module in isolation, this aggregation is also implemented in a separate module: the local context module (LCM). Given the initial token representation produced by SRM, the LCM refines it by integrating the representation of surrounding tokens using attention. The importance of each token in the local context is determined by the token's contextualized representation –produced by the oracle– accessed through an information bottleneck.

Using distillation from the oracle to these modules in combination with information bottlenecks, we are enforcing more structure on the oracle's contextualization process (i.e., constraining its flexibility) by modeling it as two steps: a combination of static sub-embeddings (SRM), and contextualization using a small local window (LCM). All in order to gain insight into the role of token-context sub-embeddings and local context.

The experimental results confirmed our initial assumptions stated above. Experimental results show that dense representations produced by our module combination (SRM-LCM) with only $K = 10$ sub-embeddings and a local window of $ws = 1$, can perform on par with BERT's representations when ranking with ColBERT's late interaction mechanism. Indeed, we have shown through concrete examples the importance of LCM for modeling fine-grained variations, due to context, which SRM cannot capture.

This evaluation confirms our hypothesis that the flexibility of BERT's contextualization process is not necessary for the ranking task. In other words, specialized ranking architectures can be designed by stripping away unnecessary elements in BERT's contextualization process (e.g., attending to global context), thereby possibly achieving better effectiveness/efficiency trade-offs.

Perspectives and Future Work

While many works have proposed exploiting PLMs for neural IR, and the field has seen incredible progress in a short period of time, we are far from seeing the end of this line of research. We believe there are still many challenges to be addressed and exciting directions to explore. Our contribution, as well as the various experiments conducted, open many perspectives.

In the short term, our perspectives address the following aspects:

— Our work on domain knowledge integration through marking strategies (Chapter 7) offers many promising leads for future work. First, our current approach assumes all query terms to be of equal importance when, in reality, they hardly have the same importance, especially in long descriptions. It would be interesting to integrate query term importance during marking. However, the challenge is defining the importance of a term. One possible estimator could be the traditional inverse document frequency of each term. Different marker tokens can then be used to distinguish different levels of term importance. Alternatively, the representations produced by BERT for the marker tokens can be weighted by query term importance before proceeding with relevance estimation.

— In our current work, we use marker tokens to convey exact matching signals. However, other techniques can be explored for explicitly emphasizing exact matches for PLMs. An interesting idea to explore is adding a term matching embedding to the actual input embedding. For instance, BERT's input embedding includes the token embeddings, a positional embedding to inform BERT of token positions and sequencing, and a segment embedding to indicate whether the token is in the first or second segment (see Section 3.5). Similarly, we can define an additional term matching embedding to add to the input embeddings of the tokens that match exactly between the query and document to emphasize them.

— Our current contribution focuses on reranking with cross-encoder architectures. Nevertheless, the current research landscape focuses on efficient bi-encoder designs for learning dense representations. A major reflection element in this perspective is adapting our exact match marking strategy for dense retrieval with bi-encoders. Because marking needs to be performed online upon the reception of the query, the document representations cannot be precomputed. In order to obviate the need for online marking, an idea to consider is to use knowledge distillation from our current (expensive) cross-encoder models (i.e., Sim-Pair $_{\text{BERT}}$ and Sim-Pair $_{\text{ELECTRA}}$) for improving the supervision of

bi-encoder models (see Section 6.2). On the other hand, our ablation study shows that exact match marking can be restricted to the general-phase fine-tuning, and still bring gains after in-domain fine-tuning without marking (see Section 5.3). These ideas can both help address the poor generalization capabilities of bi-encoders [232].

— In our second work (Chapter 8) investigating contextualization in transformers for ranking, we focused on the ColBERT [112] architecture as it is more interpretable (i.e., uses token-level representations and simple MaxSim operations). Nonetheless, this architecture can use any PLM for encoding, and we only focus on the distilbert [216] encoder for its reduced size (only 6 layers of transformers). Thus, we could investigate different PLMs as oracles, such as BERT [58], ELECTRA [40]. Furthermore, we can study the impact of different training regimes on the quality of the distilled static sub-embeddings. For instance, the original ColBERT model [112] with a BERT encoder was fine-tuned from MS MARCO labelled data, while the ColBERT model used in our experiments was fine-tuned using distillation from an ensemble of teachers [88].

— In the same contribution, our results on the out-of-distribution dataset Robusto4 indicate poor zero-shot generalizability of our simple aggregation-based representation method. Further analysis on other out-of-domain full-length test collections is required to determine the reasons behind such results.

In the long term, we want to investigate the following aspects:

— For our work on exact match signals integration, we proposed using special marker tokens to emphasize exact matching terms. Furthermore, we present a study where these same tokens endorse a different role: query expansion. The results of this investigation demonstrate the potential of marker tokens for assuming other roles. Thus, an interesting research path would study marker tokens for conveying other signals such as synonymy, concepts, or topics from knowledge bases. Highlighting such important signals, especially topics and concepts that are independent of the query, can be integrated into more efficient bi-encoder designs to improve their retrieval quality.

— In the context of our first contribution, we have empirically shown that exact match marking is effective on standard ad hoc benchmarks. Nevertheless, in terms of explainability, additional analysis is required to understand how the marker tokens convey the exact match signals to BERT and how they are integrated into the relevance prediction process. We already presented a primary exploration of the contribution of the contextualized representations of marker tokens for relevance estimation in Section 5.6. Nevertheless, further exploration can provide

insight into how they intervene in the contextualization process and how to exploit them for relevance matching properly. Understanding the contribution of marker tokens in general or in the specific context of exact matching can open exciting paths forward to building effective and practical ranking models. Combined with the insights from our second contribution, we can carefully design a new efficient ranking architecture based on static sub-embeddings, contextualization through reduced local windows, and exact match integration. We could rethink the design of transformers toward more interpretable and efficient ranking architectures, and expand previous work such as TK [92], TKL [91], or CK [166] covered in Section 5.2. Notably, the importance of local context for characterizing a term encourages local attention schemes [91, 14], and hierarchical models based on transformers [171]. Moreover, distillation can be leveraged to transfer the "free" general language understanding of PLMs, which was acquired through self-supervision on large amounts of data, into the clean-slate redesigns of transformers for ranking.

Appendices

Appendix A

Additional results using exact match marking strategies

We present in this appendix additional experimental results to complement the evaluation results discussed in Chapter 7 in the context of our first contribution involving exact match marking.

First, we discuss the results of our approach on test collections based on the MS MARCO passage corpus [9] in Section A.1. Then in Section A.2, we report additional results on the full-length document collections to show the behaviour of all the marking strategies we proposed (see Section 7.3) on in-domain collections (i.e., TREC DL Document ranking 2019-2020) and out-of-domain collections (i.e., Robust04 and GOV2).

1 *Additional results on passage reranking collections*

Early work with BERT focused on the reranking task in retrieve-then-rerank architectures. We have seen in Chapter 4 that many works focused on full-length document collections such as TREC Robust04 [2, 51, 122]. Combined with the fact that our approach integrates a traditional IR cue that was used with these standard collections over the years, we chose to evaluate our approach on full-length document collections.

Nevertheless, our monoBERT-based models were fine-tuned using the training set of the MS MARCO passage collection, it is only natural to evaluate their performance on test collections involving the passage corpus from this same collection. We report the results of marking models compared to the vanilla baseline on the following test collections:

— **MS MARCO Dev.** This collection consists of 6,980 development queries, which are sparsely judged. The performance of the models is measured using the Mean Reciprocal Rank, MRR@10, metric.

— **TREC Deep Learning passage ranking.** We consider the densely-judged query sets of 43 and 54 queries from the TREC Deep Learning (DL) passage reranking tracks of 2019 (DL'19) [45] and 2020 (DL'20) [44]. Unlike MS MARCO Dev, there are more passages annotated per query, and the relevance judgements are graded (instead of binary judgements), allowing to use the more informative nDCG@10 metric.

For both collections, we use Anserini's [255] implementation of BM25 with default parameters to retrieve the top-1000 candidate passages for reranking.

Table A.1 – Reranking effectiveness on MS MARCO Dev, and TREC DL 2019 and DL 2020 Passage ranking tasks. Best performance is highlighted in **bold**. Change rate over the vanilla baseline are reported for each collection (%).

	Dev		DL'19		DL'20	
Model	MRR@10		nDCG@10		nDCG@10	
BM25	0.1840	-	0.5058	-	0.4796	-
Vanilla $_{\text{BERT}}$	**0.3634**	-	0.7009	-	0.7017	-
Sim-Doc $_{\text{BERT}}$	0.3548	−2.4%	0.7131	+1.7%	0.6985	−0.5%
Sim-Pair $_{\text{BERT}}$	0.3606	−0.8%	0.7035	+0.4%	0.7082	+0.9%
Pre-Doc $_{\text{BERT}}$	0.3262	−10.%	0.7104	+1.4%	0.7020	+0.0%
Pre-Pair $_{\text{BERT}}$	0.3154	−13.%	**0.7169**	+2.3%	0.7095	+1.1%

Table A.1 shows the evaluation results on the three passage-level collections [1]. Adding exact match highlights on the MS MARCO Dev queries degrades the performance of the vanilla model especially when using the precise marker type (i.e., Pre-Doc and Pre-Pair). Interestingly, the behavior of our models is constant across the TREC DL 2019 and 2020 ranking collections on both passages and long documents (refer to Table A.2 for results on TREC DL document collections). The exact match marking on these densely judged queries performs on par with the vanilla baseline. It further leads to a slight improvement in ranking effectiveness when using the Pre-Pair marking strategy as opposed to MS MARCO Dev.

This disparity in the impact of the exact match marking on the MS MARCO Dev and TREC DL queries can be due to the fact that Dev queries are sparsely judged with generally a single relevant passage per query. This is not a common scenario in ad hoc ranking for which the exact match cue was intended. In contrast, TREC DL queries are densely judged with different relevance degrees and more than one relevant passage per query. The TREC DL test collections are, hence, more close to the ad hoc ranking task.

Additionally, our models were fine-tuned on the MS MARCO training set which uses the same passage corpus as the test collections. That is, our models had enough training on the target task and domain, i.e., the same data distribution, especially on the MS MARCO Dev collection. Therefore, the models are capable of modeling the passage ranking task well enough that adding explicit exact match does not contribute more relevance signals that were not already captured by monoBERT [2]. Our results support previous evidence from experiments conducted by Lin et al. [129], which indicated that

1. The Pre-Pair $_{\text{BERT}}$ model has the same architecture with our previously proposed MarkedBERT [21]. However, their fine-tuning setups are different, and thus their results differ.

2. In contrast, MarkedBERT was trained under a low data regime. As a result, exact match marking with the pre-pair strategy was beneficial and lead to gains in performance.

exact match signals do not appear to provide additional value to monoBERT on the MS MARCO passage ranking task; at least when using a simple linear combination of BM25 and monoBERT scores [129], or marking strategies in our case [3].

2 *Additional results on full-length document reranking collections*

In this section, we report all our marking strategies' results on both in-domain and out-of-domain document collections. We follow the same experimental setup described in Section 7.4. We first present in-domain evaluation results on TREC DL 2019 and 2020 document ranking collections using both a BERT and ELECTRA cores in Section 2.1. Then we move to the out-of-domain evaluation in Section 2.2 in both the zero-shot transfer setting and the multi-phase fine-tuning setting.

2.1 *In-domain evaluations*

Table A.2 reports the results of all our models with both BERT and ELECTRA cores on the TREC DL 2019 and 2020 Document ranking tasks.

2.2 *Out-of-domain evaluations*

In this section we report the experimental results on the Robust04 and GOV2 collections. For readability, we present the results obtained with a BERT core in Section 2.2.1, and with an ELECTRA core in Section 2.2.2.

2.2.1 *Results using the BERT core*

ZERO-SHOT TRANSFER SETTING Table A.3 shows the full results obtained using all the proposed strategies on Robust04 and GOV2 collections at cutoff 100 and 1,000. We report results using the title, description and hybrid runs. The results at cutoff 1,000 complement the reported results in Tables 7.6 and 7.8. For the 100-cutoff results, they complement the results of Table 7.10 for the zero-shot transfer section. We report the results at cutoff 100 for direct comparison with the multi-phase fine-tuning setting where we only rerank the top-100 documents retrieved by BM25 as a trade-off between effectiveness and efficiency.

3. Nonetheless, exact match integration through marking will prove effective when applied on out-of-distribution data. Moreover, existing work proved that BM25 retrieval scores are helpful in boosting end-to-end effectiveness on out-of-domain collections [2, 149]

Table A.2 – Reranking effectiveness on the TREC DL 2019 and DL 2020 Document ranking tasks. Best performance is highlighted in **bold**. Significant improvements over the vanilla baseline with $p < 0.05$ are indicated with †, for the same core. Change rate over the vanilla baseline for the same core type are reported for each metric (%).

TREC DL Doc	DL 19				DL 20			
Model	nDCG@10		MAP		nDCG@10			
BM25	0.5176	-	0.2434	-	0.5286	-	0.3793	-
BM25+RM3	0.5169	-	0.2772	-	0.5248	-	0.4006	-
Vanilla $_{\text{BERT}}$	0.6726	-	0.3006	-	0.6340	-	**0.4523**	-
Sim-Doc $_{\text{BERT}}$	0.6858	+2.0%	0.3038	+1.1%	0.6340	+0.0%	0.4414	−2.4%
Sim-Pair $_{\text{BERT}}$	0.6798	+1.1%	0.3057	+1.7%	0.6495	+2.4%	0.4505	−0.4%
Pre-Doc $_{\text{BERT}}$	0.6777	+0.8%	**0.3061**	+1.8%	0.6368	+0.4%	0.4513	−0.2%
Pre-Pair $_{\text{BERT}}$	**0.7025**†	+4.4%	0.3018	+1.8%	**0.6498**	+2.5%	0.4497	−0.6%

TREC DL Doc	DL 19				DL 20			
Model	nDCG@10		MAP		nDCG@10			
BM25	0.5176	-	0.2434	-	0.5286	-	0.3793	-
BM25+RM3	0.5169	-	0.2772	-	0.5248	-	0.4006	-
Vanilla $_{\text{ELECTRA}}$	0.6738	-	0.2976	-	0.6236	-	0.4297	-
Sim-Doc $_{\text{ELECTRA}}$	**0.6889**	+2.2%	**0.3082**	+3.6%	0.6369	+2.1%	0.4482†	+4.3%
Sim-Pair $_{\text{ELECTRA}}$	0.6816	+1.2%	0.3062	+2.9%	0.6331	+1.5%	0.4543†	+5.7%
Pre-Doc $_{\text{ELECTRA}}$	0.6801	+0.9%	0.3061	+2.9%	**0.6453**	+3.5%	**0.4582**†	+6.6%
Pre-Pair $_{\text{ELECTRA}}$	0.6763	+0.4%	0.2886	−3.0%	0.6234	−0.0%	0.4306	+0.2%

Table A.3 – Reranking effectiveness in the zero-shot transfer setting of all our models on Robust04 and GOV2 collections. Best results, for each cutoff, are highlighted in **bold**. Significant improvements over the Vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively, for the same cutoff. For each measure, the improvement rate over the Vanilla baseline is given (%).

Robust04	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	-	0.3631	-	0.4058	-	0.3345	-	0.4240	-	0.3631	-
BM25+RM3	0.4407	-	0.3821	-	0.4255	-	0.3661	-	0.4407	-	0.3821	-
Top-100												
Vanilla $_{\text{BERT}}$	0.4764	-	0.4096	-	0.4611	-	0.3867	-	0.4989	-	0.4245	-
Sim-Doc $_{\text{BERT}}$	0.4678	−1.8%	0.4042	−1.3%	0.4616	+0.1%	0.3865	−0.1%	0.4912	−1.5%	0.4129	−2.7%
Sim-Pair $_{\text{BERT}}$	0.4763	−0.0%	**0.4129**	+0.8%	**0.4923**‡	+6.8%	**0.4084**‡	+5.6%	**0.5273**‡	+5.7%	**0.4434**‡	+4.5%
Pre-Doc $_{\text{BERT}}$	**0.4781**	+0.4%	0.4078	−0.4%	0.4867‡	+5.6%	0.4016‡	+3.9%	0.5205‡	+4.3%	0.4294‡	+1.2%
Pre-Pair $_{\text{BERT}}$	0.4700	−1.3%	0.4064	−0.8%	0.4812‡	+4.4%	0.3974‡	+2.8%	0.5132‡	+2.9%	0.4410‡	+3.9%
Top-1000												
Vanilla $_{\text{BERT}}$	0.4652	-	0.4046	-	0.4510	-	0.3851	-	0.4845	-	0.4147	-
Sim-Doc $_{\text{BERT}}$	0.4447	−4.4%	0.3831	−5.3%	0.4166	−7.6%	0.3510	−8.9%	0.4476	−7.6%	0.3817	−7.9%
Sim-Pair $_{\text{BERT}}$	**0.4773**	+2.6%	**0.4155**	+2.7%	**0.4931**‡	+9.3%	**0.4169**‡	+8.3%	**0.5239**‡	+8.1%	**0.4446**‡	+7.2%
Pre-Doc $_{\text{BERT}}$	**0.4767**	+2.5%	**0.4084**	+0.9%	0.4789‡	+6.2%	0.4026‡	+4.5%	0.5035‡	+3.9%	0.4235	+2.1%
Pre-Pair $_{\text{BERT}}$	0.4654	+0.0%	0.4024	−0.5%	0.4795‡	+6.3%	0.4034‡	+4.8%	0.5086‡	+5.0%	0.4319‡	+4.1%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-	0.4774	-	0.5362	-
BM25+RM3	0.4851	-	0.5634	-	0.4212	-	0.4966	-	0.4851	-	0.5634	-
Top-100												
Vanilla $_{\text{BERT}}$	0.5098	-	0.5916	-	0.4928	-	0.556	-	0.5510	-	0.6312	-
Sim-Doc $_{\text{BERT}}$	0.5146	+0.9%	0.5936	+0.3%	0.4884	−0.9%	0.5557	−0.1%	0.5497	−0.2%	**0.6359**	+0.7%
Sim-Pair $_{\text{BERT}}$	**0.5181**	+1.6%	**0.5990**	+1.3%	0.4904	−0.5%	0.5597	+0.7%	0.5531	+0.4%	0.6346	+0.5%
Pre-Doc $_{\text{BERT}}$	0.5100	−0.0%	0.5903	−0.2%	**0.4952**	+0.5%	**0.5601**	+0.7%	**0.5568**	+1.1%	0.6322	+0.2%
Pre-Pair $_{\text{BERT}}$	0.5168	+1.4%	0.5936	+0.3%	0.4920	−0.2%	0.5584	+0.4%	0.5559	+0.9%	0.6332	+0.3%
Top-1000												
Vanilla $_{\text{BERT}}$	0.4533	-	0.5272	-	0.4696	-	0.5248	-	0.4937	-	0.5611	-
Sim-Doc $_{\text{BERT}}$	0.4588	+1.2%	0.5349	+1.5%	0.4686	−0.2%	0.5262	+0.3%	0.4943	+0.1%	0.5607	−0.1%
Sim-Pair $_{\text{BERT}}$	0.4468	−1.4%	0.5134	−2.6%	0.4687	−0.2%	0.5326	+1.5%	0.4991	+1.1%	**0.5695**	+1.5%
Pre-Doc $_{\text{BERT}}$	0.4485	−1.1%	0.5121	−2.9%	**0.4768**	+1.5%	**0.5315**	+1.3%	**0.5013**	+1.5%	0.5668	+1.0%
Pre-Pair $_{\text{BERT}}$	0.4515	−0.4%	0.5238	−0.6%	0.4752	+1.2%	0.5285	+0.7%	0.4979	+0.9%	0.5594	−0.3%

Table A.4 – Reranking effectiveness in the multi-phase fine-tuning setting of the different models on Robusto4 and GOV2 collections. Best results are highlighted in **bold**. Significant improvements over the Vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively. For each measure, the improvement rate over the Vanilla baseline is given (%).

Robusto4	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	-	0.3631	-	0.4058	-	0.3345	-	0.4240	-	0.3631	-
BM25+RM3	0.4407	-	0.3821	-	0.4255	-	0.3661	-	0.4407	-	0.3821	-
Vanilla $_{\text{BERT}}$	0.4995	-	0.4275	-	0.5368	-	0.4492	-	0.5546	-	0.4715	-
Sim-Doc $_{\text{BERT}}$	0.4976	−0.4%	0.4273	−0.0%	0.5378	+0.2%	0.4470	−0.5%	0.5632	+1.6%	0.4783	+1.4%
Sim-Pair $_{\text{BERT}}$	**0.5058**	+1.3%	**0.4371**	+2.2%	0.5479†	+2.1%	0.4574†	+1.8%	**0.5701**‡	+2.8%	**0.4815**‡	+2.1%
Pre-Doc $_{\text{BERT}}$	0.5039	+0.9%	0.4331	+1.3%	0.5462	+1.8%	0.4568	+1.7%	0.5607	+1.1%	0.4757	+0.9%
Pre-Pair $_{\text{BERT}}$	0.5021	+0.5%	0.4333	+1.4%	**0.5532**‡	+3.1%	**0.4631**‡	+3.1%	**0.5699**‡	+2.8%	**0.4821**‡	+2.2%
GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-	0.4774	-	0.5362	-
BM25+RM3	0.4851	-	0.5634	-	0.4212	-	0.4966	-	0.4851	-	0.5634	-
Vanilla $_{\text{BERT}}$	0.5476	-	0.6302	-	0.5175	-	0.5772	-	0.5909	-	0.6604	-
Sim-Doc $_{\text{BERT}}$	0.5413	−1.2%	0.6248	−0.9%	0.5151	−0.5%	0.5799	+0.5%	0.5754	−2.6%	0.6513	−1.4%
Sim-Pair $_{\text{BERT}}$	**0.5743**‡	+4.9%	**0.6540**‡	+3.8%	0.5406‡	+4.5%	**0.6084**‡	+5.4%	**0.5998**	+1.5%	**0.6758**	+2.3%
Pre-Doc $_{\text{BERT}}$	0.5635†	+2.9%	0.6470†	+2.7%	**0.5432**‡	+5.0%	**0.6074**‡	+5.2%	**0.6002**	+1.6%	0.6715	+1.7%
Pre-Pair $_{\text{BERT}}$	0.5705‡	+4.2%	0.6513‡	+3.3%	0.5387‡	+4.1%	0.6034‡	+4.5%	0.5966	+1.0%	0.6708	+1.6%

MULTI-PHASE FINE-TUNING SETTING Table A.4 shows the results obtained using the multi-phase fine-tuning setting described in section 5.3 for all our models using all proposed marking strategies. These results expand Table 7.10.

2.2.2 *Results using the ELECTRA core*

ZERO-SHOT TRANSFER SETTING Table A.5 resumes the results of applying all proposed marking strategies on the ELECTRA core model for Robusto4 and GOV2 collections. This table complements the results presented in Table 7.13. We add the results at the reranking cutoff 100 in order to give an idea about the zero-shot setting results without in-domain fine-tuning directly comparable with the multi-phase fine-tuning setting that uses the same reranking threshold of 100 in Table A.6.

MULTI-PHASE FINE-TUNING SETTING Table A.6 results complements the results presented in Table 7.14 obtained in the multi-phase fine-tuning setting using all exact match marking strategies proposed in this paper.

Table A.5 – Reranking effectiveness in the zero-shot transfer setting of the different models on Robust04 and GOV2 collections. Best results, for each cutoff, are highlighted in **bold**. Significant improvements over the Vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively, for the same cutoff. For each measure, the improvement rate over the Vanilla baseline is given (%).

Robust04	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	-	0.3631	-	0.4058	-	0.3345	-	0.4240	-	0.3631	-
BM25+RM3	0.4407	-	0.3821	-	0.4255	-	0.3661	-	0.4407	-	0.3821	-
Top-100												
Vanilla ELECTRA	0.4712	-	0.4108	-	0.4721	-	0.3988	-	0.5103	-	0.4323	-
Sim-Doc ELECTRA	0.4680	−0.7%	0.4054	−1.3%	**0.4804**†	+1.8%	**0.4040**	+1.3%	0.5231‡	+2.5%	**0.4422**†	+2.3%
Sim-Pair ELECTRA	**0.4820**†	+2.3%	**0.4181**	+1.8%	0.4749	+0.6%	0.3964	−0.6%	**0.5235**†	+2.6%	0.4418†	+2.2%
Pre-Doc ELECTRA	0.4663	−1.0%	0.4080	−0.7%	0.4789	+1.4%	0.4016	+0.7%	0.5182	+1.5%	0.4378	+1.3%
Pre-Pair ELECTRA	0.4668	−0.9%	0.4064	−1.1%	0.4740	+0.4%	0.4002	+0.4%	0.5169	+1.3%	0.4416	+2.2%
Top-1000												
Vanilla ELECTRA	0.4416	-	0.3833	-	0.4482	-	0.3831	-	0.4782	-	0.4141	-
Sim-Doc ELECTRA	0.4479	+1.4%	0.3878	+1.2%	0.4640†	+3.5%	**0.3948**†	+3.1%	0.4970‡	+3.9%	0.4247	+2.6%
Sim-Pair ELECTRA	**0.4717**‡	+6.8%	**0.4124**‡	+7.6%	0.4597	+2.6%	0.3886	+1.4%	**0.5043**‡	+5.5%	**0.4263**	+2.9%
Pre-Doc ELECTRA	0.4500	+1.9%	0.3912	+2.1%	**0.4662**†	+4.0%	**0.3948**†	+3.1%	0.4996‡	+4.5%	0.4251	+2.7%
Pre-Pair ELECTRA	0.4511	+2.2%	0.3934	+2.6%	0.4537	+1.2%	0.3878	+1.2%	0.4936†	+3.2%	0.4245	+2.5%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-	0.4774	-	0.5362	-
BM25+RM3	0.4851	-	0.5634	-	0.4212	-	0.4966	-	0.4851	-	0.5634	-
Top-100												
Vanilla ELECTRA	0.5278	-	0.6094	-	0.5153	-	0.5785	-	**0.5803**	-	**0.6617**	-
Sim-Doc ELECTRA	0.5342	+1.2%	**0.6188**	+1.5%	0.5120	−0.6%	0.5795	+0.2%	0.5761	−0.7%	0.6527	−1.4%
Sim-Pair ELECTRA	**0.5387**	+2.1%	0.6171	+1.3%	**0.5207**	+1.0%	**0.5859**	+1.3%	0.5801	−0.0%	0.6587	−0.5%
Pre-Doc ELECTRA	0.5350	+1.4%	0.6148	+0.9%	0.5086	−1.3%	0.5711	−1.3%	0.5779	−0.4%	0.6557	−0.9%
Pre-Pair ELECTRA	0.5306	+0.5%	0.6131	+0.6%	0.5108	−0.9%	0.5775	−0.2%	0.5760	−0.7%	0.6557	−0.9%
Top-1000												
Vanilla ELECTRA	0.4668	-	0.5332	-	0.4986	-	0.5601	-	0.5147	-	0.5765	-
Sim-Doc ELECTRA	0.4796	+2.7%	0.5530†	+3.7%	0.4958	−0.6%	0.5544	−1.0%	0.5198	+1.0%	**0.5930**	+2.9%
Sim-Pair ELECTRA	**0.4881**†	+4.6%	**0.5577**‡	+4.6%	**0.5030**	+0.9%	**0.5634**	+0.6%	**0.5249**	+2.0%	0.5923	+2.7%
Pre-Doc ELECTRA	0.4845†	+3.8%	0.5530‡	+3.7%	0.4981	−0.1%	0.5560	−0.7%	0.5212	+1.3%	0.5883	+2.0%
Pre-Pair ELECTRA	0.4820	+3.3%	0.5513†	+3.4%	0.4828	−3.2%	0.5419	−3.2%	0.5075	−1.4%	0.5711	−0.9%

Table A.6 – Reranking effectiveness in the multi-phase fine-tuning setting of the different models on Robust04 and GOV2 collections. Best results are highlighted in **bold**. Significant improvements over the Vanilla baseline with $p < 0.05$ and $p < 0.01$ are indicated with † and ‡ respectively. For each measure, the improvement rate over the Vanilla baseline is given (%).

Robust04	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4240	-	0.3631	-	0.4058	-	0.3345	-	0.4240	-	0.3631	-
BM25+RM3	0.4407	-	0.3821	-	0.4255	-	0.3661	-	0.4407	-	0.3821	-
Vanilla ELECTRA	0.5375	-	0.4560	-	0.5676	-	0.4663	-	0.5901	-	0.4902	-
Sim-Doc ELECTRA	0.5367	−0.1%	0.4560	+0.0%	0.5662	−0.2%	0.4683	+0.4%	0.5893	−0.1%	0.4912	+0.2%
Sim-Pair ELECTRA	0.5380	+0.1%	0.4564	+0.1%	0.5686	+0.2%	**0.4705**	+0.9%	**0.5927**	+0.4%	0.4942	+0.8%
Pre-Doc ELECTRA	0.5338	−0.7%	**0.4590**	+0.7%	**0.5705**	+0.5%	0.4697	+0.7%	0.5889	−0.2%	0.4926	+0.5%
Pre-Pair ELECTRA	**0.5390**	+0.3%	0.4566	+0.1%	0.5677	+0.0%	0.4699	+0.8%	**0.5930**	+0.5%	**0.4970**	+1.4%

GOV2	Title run				Description run				Hybrid run			
Model	nDCG@20		P@20		nDCG@20		P@20		nDCG@20		P@20	
BM25	0.4774	-	0.5362	-	0.4264	-	0.4705	-	0.4774	-	0.5362	-
BM25+RM3	0.4851	-	0.5634	-	0.4212	-	0.4966	-	0.4851	-	0.5634	-
Vanilla ELECTRA	0.5784	-	0.6621	-	**0.5629**	-	**0.6279**	-	0.6149	-	0.6862	-
Sim-Doc ELECTRA	0.5891	+1.8%	0.6685	+1.0%	0.5044	−10.4%	0.5758	−8.3%	0.6120	−0.5%	0.6926	+0.9%
Sim-Pair ELECTRA	0.5868	+1.5%	0.6661	+0.6%	0.5552	−1.4%	0.6225	−0.9%	0.6133	−0.3%	**0.6926**	+0.9%
Pre-Doc ELECTRA	0.5841	+1.0%	0.6634	+0.2%	0.5524	−1.9%	0.6188	−1.4%	0.6130	−0.3%	0.6852	−0.1%
Pre-Pair ELECTRA	**0.5920**†	+2.4%	**0.6718**	+1.5%	0.5486	−2.5%	0.6134	−2.3%	**0.6207**	+0.9%	**0.6956**	+1.4%

Appendix B

Reproducibility

Reproducibility or replicability of experimental results are primary concerns in all fields of research [24]. Information retrieval (IR) is notably a field deeply rooted in empirical experiments and has always strived for reproducibility by relying on standard evaluation protocols based on the Cranfield paradigm [42]. To this end, the IR community has developed standard experimental benchamrks and large-scale evaluation initiatives across the decades such as the TREC campaigns (see chapter 1).

In order to ensure the repeatability of the experimental results presented in our contributions, we chose commonly recognized benchmarks for evaluation. More specifically, we report results on the most commonly used benchmarks in the literature for the task at the time of the publication. That is, standard long document collections such as Robusto4 and TREC DL document ranking collections for reranking with cross-encoders. On the other hand, the focus on test collections based on the MS MARCO passage corpus with bi-encoders. Additionally, we share details about our experimental environment including complete model architectures, training parameters, and inference process.

Considering the importance of reproducibility and its complexity especially in the context of neural models, we present in this appendix further details on how to reproduce our results.

1 *Reproducing the results of the exact match marking contribution*

We discuss our first contribution in Chapter 7, in which we examine the impact of explicit exact match hints on pre-trained language models. We describe our model configuration and marking strategies in Section 7.3. We further provide the detailed experimental setup used for producing our results and baseline results in Section 7.4.

We conduct our experiments with the base configuration of both BERT [1] and ELECTRA [2] models publicly available on Hugginface hub. We fine-tune these models for ranking using the monoBERT architecture for relevance classification on the MS MARCO training set. We provide the fine-tuning setting used in Section 4.1.

We conduct all our experiments on a single free Colab TPU. Our main code dependencies are Huggingface Transformers [247] and Anserini [255]. Our

1. https://huggingface.co/bert-base-uncased
2. https://huggingface.co/google/electra-base-discriminator

code and reproduction scripts are publicly available [3], including the Colba notebooks for training and inference.

Our fine-tuned models can be directly used via the Transformers library and can be found on the Hugginface model hub [4].

2 *Reproducing the results of the simpler contextualization process contribution*

In Chapter 8, we present our second contribution which investigates the contextualization process in pre-trained language models for soft matching in the context of ranking, and whether it can be replaced by a simpler process.

We detail our proposition and its implementation in Section 3.1, followed by its life cycle including pre-training, fine-tuning and inference in Section 3.2. For reproducibility purposes, we also provide the exact experimental settings used at each stage of the approach life cycle in Section 8.4.

In our experiments presented in Section 8.5, we rely on a ColBERT-based model fine-tuned on MS MARCO train triples with distillation from an ensemble of teachers. This model uses a distilbert [216] encoder with $66M$ parameters, and was made publicly available by Hofstätter et al. [88] on Hugginface hub [5].

We pre-train, fine-tune and evaluate our proposed constrained contextualization approach on publicly available datasets, we provide detailed descriptions of each dataset in Section 8.4. We report the official metrics recommended for each collection test and use `trec_eval` [6] for computing the metrics.

During pre-training, we randomly sample $N = 50$ tokens per passage uniformly. This hyper-parameter is set empirically in pilot experiments where we evaluated different sample sizes among 25, 50 and using all sequence tokens. We found that $N = 50$ gives the best ranking performance on MS MARCO Dev in terms of MRR@10 (official metric). We also experimented with a frequency-aware sampling strategy during pilot experiments, however this strategy was less effective then random sampling from a uniform distribution.

All our experiments were conducted using a NVIDIA Quadro RTX 8000 GPU, AMD EPYC 7502P CPU, and 256 GB RAM. The training and inference settings are described in Section 8.4.

3. https://github.com/BOUALILILila/ExactMatchMarking
4. https://huggingface.co/LilaBoualili/
5. https://huggingface.co/sebastian-hofstaetter/colbert-distilbert-margin_mse-T2-msmarco
6. https://github.com/usnistgov/trec_eval

Our main training and inference dependencies are PyTorch [188], HuggingFace Transformers [247], and Capreolus [259]. We make our code, model checkpoints and reproduction scripts publicly available[7].

7. https://github.com/BOUALILILila

Bibliography

[1] Mohammad Reza Abbasifard, Bijan Ghahremani, and Hassan Naderi. 2014. Article: A Survey on Nearest Neighbor Search Methods. *International Journal of Computer Applications* 95, 25 (June 2014), 39–52. Full text available.

[2] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3490–3496.

[3] Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Trans. Inf. Syst.* 20, 4 (oct 2002), 357–389.

[4] Vo Ngoc Anh, Owen de Kretser, and Alistair Moffat. 2001. Vector-Space Ranking with Effective Early Termination. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, Louisiana, USA) *(SIGIR '01)*. Association for Computing Machinery, New York, NY, USA, 35–42.

[5] Nima Asadi and Jimmy Lin. 2013. Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '13)*. Association for Computing Machinery, New York, NY, USA, 997–1000.

[6] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. *Inf. Syst.* 87, C (jan 2020), 13.

[7] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

[8] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768* (2020).

[9] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268v3* (2018).

[10] Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2895–2905.

[11] Mustapha Baziz, Mohand Boughanem, and Nathalie Aussenac-Gilles. 2005. Conceptual Indexing Based on Document Content Representation. In *Context: Nature, Impact, and Role*, Fabio Crestani and Ian Ruthven (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 171–186.

[12] Jeffrey S Beis and David G Lowe. 1997. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1000–1006.

[13] Nicholas J Belkin. 1980. Anomalous states of knowledge as a basis for information retrieval. *Canadian journal of information science* 5, 1 (1980), 133–143.

[14] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[15] Michael Bendersky and W. Bruce Croft. 2012. Modeling Higher-Order Term Dependencies in Information Retrieval Using Query Hypergraphs. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 941–950.

[16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[17] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 41–48.

[18] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (sep 1975), 509–517.

[19] Adam Berger and John Lafferty. 1999. Information Retrieval as Statistical Translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, California, USA) *(SIGIR '99)*. Association for Computing Machinery, New York, NY, USA, 222–229.

[20] Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 4758–4781.

[21] Lila Boualili, Jose G. Moreno, and Mohand Boughanem. 2020. MarkedBERT: Integrating Traditional IR Cues in Pre-Trained Language Models for Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1977–1980.

[22] Lila Boualili, Jose G Moreno, and Mohand Boughanem. 2022. Highlighting exact matching via marking strategies for ad hoc document ranking with pretrained contextualized language models. *Information Retrieval Journal* (2022), 1–47.

[23] Mohand Boughanem, Claude Chrisment, and Chantal Soulé-Dupuy. 1999. Query modification based on relevance back-propagation in an ad hoc environment. *Information processing & management* 35, 2 (1999), 121–139.

[24] Timo Breuer, Nicola Ferro, Norbert Fuhr, Maria Maistro, Tetsuya Sakai, Philipp Schaer, and Ian Soboroff. 2020. How to Measure the Reproducibility of System-Oriented IR Experiments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 349–358.

[25] Chris Buckley and Ellen M. Voorhees. 2004. Retrieval Evaluation with Incomplete Information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Sheffield, United Kingdom) *(SIGIR '04)*. Association for Computing Machinery, New York, NY, USA, 25–32.

[26] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine*

Learning (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 89–96.

[27] Christopher JC Burges. 2010. *From ranknet to lambdarank to lambdamart: An overview*. Technical Report 23-581. 81 pages.

[28] Arthur Câmara and Claudia Hauff. 2020. Diagnosing BERT with Retrieval Heuristics. In *Advances in Information Retrieval*, Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins (Eds.). Springer International Publishing, Cham, 605–618.

[29] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136.

[30] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2016. Quality versus Efficiency in Document Scoring with Learning-to-Rank Models. *Inf. Process. Manage.* 52, 6 (nov 2016), 1161–1177.

[31] Claudio Carpineto and Giovanni Romano. 2012. A Survey of Automatic Query Expansion in Information Retrieval. *ACM Comput. Surv.* 44, 1, Article 1 (jan 2012), 50 pages.

[32] Jia Chen, Yiqun Liu, Yan Fang, Jiaxin Mao, Hui Fang, Shenghao Yang, Xiaohui Xie, Min Zhang, and Shaoping Ma. 2022. Axiomatically Regularized Pre-Training for Ad Hoc Search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1524–1534.

[33] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J. Shane Culpepper. 2017. Efficient Cost-Aware Cascade Ranking in Multi-Stage Retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 445–454.

[34] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems* 22 (2009).

[35] Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2021. Simplified tinybert: Knowledge distillation for document retrieval. In *European Conference on Information Retrieval*. Springer, 241–248.

[36] Xuanang Chen, Ben He, Le Sun, and Yingfei Sun. 2020. ICIP at TREC-2020 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication)*, Ellen M. Voorhees and Angela Ellis (Eds.), Vol. 1266. National Institute of Standards and Technology (NIST).

[37] Xuanang Chen, Canjia Li, Ben He, and Yingfei Sun. 2019. UCAS at TREC-2019 Deep Learning Track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019 (NIST Special Publication)*, Ellen M. Voorhees and Angela Ellis (Eds.), Vol. 1250. National Institute of Standards and Technology (NIST).

[38] Francois Chollet. 2021. *Deep learning with Python*. Simon and Schuster.

[39] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Florence, Italy, 276–286.

[40] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[41] Charles LA Clarke, J Shane Culpepper, and Alistair Moffat. 2016. Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. *Information Retrieval Journal* 19, 4 (2016), 351–377.

[42] Cyril Cleverdon. 1970. Evaluation tests of information retrieval systems. *Journal of Documentation* (1970).

[43] Daniel Cohen, Bhaskar Mitra, Katja Hofmann, and W. Bruce Croft. 2018. Cross Domain Regularization for Neural Ranking Models Using Adversarial Learning. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1025–1028.

[44] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *arXiv:2102.07662* (2021).

[45] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv:2003.07820* (2020).

[46] Fabio Crestani, Mounia Lalmas, Cornelis J Van Rijsbergen, and Iain Campbell. 1998. "Is this document relevant?... probably" a survey of probabilistic models in information retrieval. *ACM Computing Surveys (CSUR)* 30, 4 (1998), 528–552.

[47] W Bruce Croft and David J Harper. 1979. Using probabilistic models of document retrieval without relevance information. *Journal of documentation* (1979).

[48] Sally Jo Cunningham, James Littin, and Ian H Witten. 1997. Applications of machine learning in information retrieval. (1997).

[49] Andrew M. Dai and Quoc V. Le. 2015. Semi-Supervised Sequence Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 3079–3087.

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

[51] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 985–988.

[52] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *Proceedings of The Web Conference 2020*. Association for Computing Machinery, New York, NY, USA, 1897–1907.

[53] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1533–1536.

[54] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-Hoc Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 126–134.

[55] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry* (Brooklyn, New York, USA) *(SCG '04)*. Association for Computing Machinery, New York, NY, USA, 253–262.

[56] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391–407.

[57] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 65–74.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[59] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC Complex Answer Retrieval Overview. In *TREC*.

[60] Kevin Duh and Katrin Kirchhoff. 2008. Learning to Rank with Partially-Labeled Data. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Singapore, Singapore) *(SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 251–258.

[61] Miles Efron, Peter Organisciak, and Katrina Fenlon. 2012. Improving Retrieval of Short Texts through Document Expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 911–920.

[62] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. 2009. The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, David van Dyk and Max Welling (Eds.), Vol. 5. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 153–160.

[63] Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 55–65.

[64] Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics* 8 (2020), 34–48.

[65] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling Diverse Relevance Patterns in Ad-Hoc Retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 375–384.

[66] Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A Formal Study of Information Retrieval Heuristics. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Sheffield, United Kingdom) *(SIGIR '04)*. Association for Computing Machinery, New York, NY, USA, 49–56.

[67] Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. Diagnostic Evaluation of Information Retrieval Models. *ACM Trans. Inf. Syst.* 29, 2, Article 7 (apr 2011), 42 pages.

[68] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[69] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292.

[70] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.

[71] Luyu Gao and Jamie Callan. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 981–993.

[72] Luyu Gao and Jamie Callan. 2022. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 2843–2853.

[73] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Modularized Transfomer-based Ranking Framework. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4180–4190.

[74] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding BERT Rankers Under Distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval* (Virtual Event, Norway) *(ICTIR '20)*. Association for Computing Machinery, New York, NY, USA, 149–152.

[75] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3030–3042.

[76] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement Lexical Retrieval Model with Semantic Residual Embeddings. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I (Lecture Notes in Computer Science)*, Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani (Eds.), Vol. 12656. Springer, 146–160.

[77] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2013), 744–755.

[78] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. 2007. Feature Selection for Ranking. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Amsterdam, The Netherlands) *(SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 407–414.

[79] Goran Glavaš and Ivan Vulić. 2020. Is supervised syntactic parsing beneficial for language understanding? an empirical investigation. *arXiv preprint arXiv:2008.06788* (2020).

[80] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[81] Robert Gray. 1984. Vector quantization. *IEEE Assp Magazine* 1, 2 (1984), 4–29.

[82] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic Models for the First-Stage Retrieval: A Comprehensive Review. *ACM Trans. Inf. Syst.* 40, 4, Article 66 (mar 2022), 42 pages.

[83] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 55–64.

[84] Donna Harman. 2011. Information retrieval evaluation. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 3, 2 (2011), 1–119.

[85] Donna Harman. 2019. Information retrieval: the early years. *Foundations and Trends in Information Retrieval* 13, 5 (2019), 425–577.

[86] James Henderson. 2020. The Unstoppable Rise of Computational Linguistics in Deep Learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 6294–6306.

[87] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[88] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666* (2020).

[89] Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019 (CEUR Workshop Proceedings)*, Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu (Eds.), Vol. 2409. CEUR-WS.org, 12–16.

[90] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 113–122.

[91] Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. 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* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 2021–2024.

[92] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. 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) (Frontiers in Artificial Intelligence and Applications)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.), Vol. 325. IOS Press, 513–520.

[93] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 328–339.

[94] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[95] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 2333–2338.

[96] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A Position-Aware Neural IR Model for Relevance Matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural*

Language Processing. Association for Computational Linguistics, Copenhagen, Denmark, 1049–1058.

[97] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-Hoc Retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 279–287.

[98] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969* (2019).

[99] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Dallas, Texas, USA) *(STOC '98)*. Association for Computing Machinery, New York, NY, USA, 604–613.

[100] Peter Ingwersen. 1992. *Information retrieval interaction*. Vol. 246. Taylor Graham London.

[101] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[102] Kyoung-Rok Jang, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Heecheol Seo. 2021. Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 1016–1029.

[103] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[104] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[105] Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. 2020. Long Document Ranking with Query-Directed Sparse Transformer. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 4594–4605.

[106] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 4163–4174.

[107] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv:1702.08734* (2017).

[108] Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. In *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II (Lecture Notes in Computer Science)*, Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins (Eds.), Vol. 12036. Springer, 28–34.

[109] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[110] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781.

[111] Diane Kelly. 2009. *Methods for evaluating interactive information retrieval systems with users*. Now Publishers Inc.

[112] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 39–48.

[113] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[114] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

[115] Jon M Kleinberg. 2000. Navigation in a small world. *Nature* 406, 6798 (2000), 845–845.

[116] Ron Kohavi, Randal M. Henne, and Dan Sommerfield. 2007. Practical Guide to Controlled Experiments on the Web: Listen to Your Customers Not to the Hippo. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Jose, California, USA) *(KDD '07)*. Association for Computing Machinery, New York, NY, USA, 959–967.

[117] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 4365–4374.

[118] Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, Louisiana, USA) *(SIGIR '01)*. Association for Computing Machinery, New York, NY, USA, 120–127.

[119] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[120] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880.

[121] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. 2018. NPRF: A Neural Pseudo Relevance Feedback Framework for Ad-hoc Information Retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 4482–4491.

[122] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *arXiv:2008.09093* (2020).

[123] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2021. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093v2* (2021).

[124] Hang Li. 2014. Learning to rank for information retrieval and natural language processing. *Synthesis lectures on human language technologies* 7, 3 (2014), 1–121.

[125] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[126] Jimmy Lin. 2019. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (jan 2019), 40–51.

[127] Jimmy Lin. 2021. The Neural Hype, Justified! A Recantation. *SIGIR Forum* 53, 2 (mar 2021), 88–93.

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

[129] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies* 14, 4 (2021), 1–325.

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

[131] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Association for Computational Linguistics, Online, 163–173.

[132] Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open Sesame: Getting inside BERT's Linguistic Knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Florence, Italy, 241–253.

[133] Binsheng Liu, Hamed Zamani, Xiaolu Lu, and J. Shane Culpepper. 2021. Generalizing Discriminative Retrieval Models Using Generative Tasks. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 3745–3756.

[134] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems* 34 (2021), 9204–9215.

[135] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 1073–1094.

[136] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade Ranking for Operational E-Commerce Search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. Association for Computing Machinery, New York, NY, USA, 1557–1565.

[137] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (mar 2009), 225–331.

[138] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692* (2019).

[139] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-Duet Neural Ranking: Understanding the Role of Knowledge Graph Semantics in Neural Information Retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2395–2405.

[140] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, Dense, and Attentional Representations for Text Retrieval. *arXiv:2005.00181* (2020).

[141] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics* 9 (2021), 329–345.

[142] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. 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* (Virtual Event, Israel) *(WSDM '21)*. Association for Computing Machinery, New York, NY, USA, 283–291.

[143] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: Bootstrapped Pre-Training with Representative Words Prediction for Ad-Hoc Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1513–1522.

[144] Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv preprint arXiv:2104.05740* (2021).

[145] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. 2021. *Pre-Training for Ad-Hoc Retrieval: Hyperlink is Also You Need*. Association for Computing Machinery, New York, NY, USA, 1212–1221.

[146] Sean MacAvaney, Sergey Feldman, Nazli Goharian, Doug Downey, and Arman Cohan. 2020. ABNIRML: Analyzing the Behavior of Neural IR Models.

[147] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 49–58.

[148] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. *Training Curricula for Open Domain Answer Re-Ranking*. Association for Computing Machinery, New York, NY, USA, 529–538.

[149] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1101–1104.

[150] Joel Mackenzie, Zhuyun Dai, Luke Gallagher, and Jamie Callan. 2020. *Efficiency Implications of Term Weighting for Passage Retrieval*. Association for Computing Machinery, New York, NY, USA, 1821–1824.

[151] Iain Mackie, Jeffrey Dalton, and Andrew Yates. 2021. How Deep is Your Learning: The DL-HARD Annotated Deep Learning Dataset. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2335–2341.

[152] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* (2016).

[153] Alireza Makhzani and Brendan J Frey. 2015. Winner-Take-All Autoencoders. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc.

[154] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[155] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1723–1727.

[156] Ryan T. McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep Relevance Ranking using Enhanced Document-Query Interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 1849–1860.

[157] Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *International Conference on Machine Learning*. PMLR, 15376–15400.

[158] Donald Metzler and W. Bruce Croft. 2005. A Markov Random Field Model for Term Dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 472–479.

[159] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking Search: Making Domain Experts out of Dilettantes. *SIGIR Forum* 55, 1, Article 13 (jul 2021), 27 pages.

[160] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking Search: Making Domain Experts out of Dilettantes. *SIGIR Forum* 55, 1, Article 13 (jul 2021), 27 pages.

[161] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[162] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. (2013), 3111–3119.

[163] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (nov 1995), 39–41.

[164] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends in Information Retrieval* 13, 1 (dec 2018), 1–126.

[165] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1291–1299.

[166] Bhaskar Mitra, Sebastian Hofstatter, Hamed Zamani, and Nick Craswell. 2020. Conformer-kernel with query term independence for document retrieval. *arXiv preprint arXiv:2007.10434* (2020).

[167] Calvin N Mooers. 1948. *Application of random codes to the gathering of statistical information*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[168] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web* (Montréal, Québec, Canada) *(WWW '16 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 83–84.

[169] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. 2021. CEQE: Contextualized Embeddings for Query Expansion. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part I*. Springer-Verlag, Berlin, Heidelberg, 467–482.

[170] Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Comput. Surv.* 41, 2, Article 10 (feb 2009), 69 pages.

[171] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. 2022. Hierarchical Transformers Are More Efficient Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics, Seattle, United States, 1559–1571.

[172] Yifan Nie, Yanling Li, and Jian-Yun Nie. 2018. 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* (Tianjin, China) *(ICTIR '18)*. Association for Computing Machinery, New York, NY, USA, 59–66.

[173] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085*, (2019).

[174] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 708–718.

[175] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* 6 (2019).

[176] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[177] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[178] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375* (2019).

[179] Cicero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 1722–1727.

[180] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten Mcnamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler Mcdonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten Rijke, and Matthew Lease. 2018. Neural

Information Retrieval: At the End of the Early Years. *Information Retrieval Journal* 21, 2–3 (June 2018), 111–182.

[181] Daniel W Otter, Julian R Medina, and Jugal K Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems* 32, 2 (2020), 604–624.

[182] Ramith Padaki, Zhuyun Dai, and Jamie Callan. 2020. Rethinking Query Expansion for BERT Reranking. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II* (Lisbon, Portugal). Springer-Verlag, Berlin, Heidelberg, 297–304.

[183] Harshith Padigela, Hamed Zamani, and W. Bruce Croft. 2019. Investigating the Successes and Failures of BERT for Passage Re-Ranking. *arxiv:1905.01758* (2019).

[184] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[185] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. *arXiv:1606.04648* (2016).

[186] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona) *(AAAI'16)*. AAAI Press, 2793–2799.

[187] Biswajit Paria, Chih-Kuan Yeh, Ian EH Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing flops to learn efficient sparse representations. *arXiv preprint arXiv:2004.05665* (2020).

[188] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.

[189] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*

(EMNLP). Association for Computational Linguistics, Doha, Qatar, 1532–1543.

[190] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237.

[191] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia) *(SIGIR '98)*. Association for Computing Machinery, New York, NY, USA, 275–281.

[192] Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Nogueira, and Jimmy Lin. 2020. H2oloo at TREC 2020: When All You Got Is a Hammer... Deep Learning, Health Misinformation, and Precision Medicine. In *Text Retrieval Conference (TREC)*.

[193] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *arXiv:2101.05667* (2021).

[194] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv:1904.07531* (2019).

[195] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 5835–5847.

[196] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI* (2018).

[197] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[198] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[199] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[200] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. 2021. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2507–2513.

[201] Daniël Rennings, Felipe Moraes, and Claudia Hauff. 2019. An axiomatic approach to diagnosing neural IR models. In *European Conference on Information Retrieval*. Springer, 489–503.

[202] C. J. Van Rijsbergen. 1979. *Information Retrieval* (2nd ed.). Butterworth-Heinemann, USA.

[203] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[204] Stephen E Robertson. 1977. The probability ranking principle in IR. *Journal of documentation* (1977).

[205] Stephen E Robertson and K Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information science* 27, 3 (1976), 129–146.

[206] S. E. Robertson and S. Walker. 1994. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '94)*. Springer-Verlag, Berlin, Heidelberg, 232–241.

[207] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994 (NIST Special Publication)*, Donna K. Harman (Ed.), Vol. 500-225. National Institute of Standards and Technology (NIST), 109–126.

[208] Joseph Rocchio. 1971. *Relevance Feedback in Information Retrieval*. Prentice Hall, Englewood, Cliffs, New Jersey, 313–323.

[209] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics* 8 (2020), 842–866.

[210] Corby Rosset, Damien Jose, Gargi Ghosh, Bhaskar Mitra, and Saurabh Tiwary. 2018. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1193–1196.

[211] Gerard. Salton. 1968. *Automatic Information Organization and Retrieval.* McGraw Hill Text.

[212] Gerard Salton. 1989. Automatic Text Processing : The Transformation, Analysis, and Retrieval of Information by Computer. *Reading: Addison-Wesley* 169 (1989).

[213] Gerard Salton and Christopher Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.* 24, 5 (aug 1988), 513–523.

[214] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (nov 1975), 613–620.

[215] Mark Sanderson. 1994. Word sense disambiguation and information retrieval. In *SIGIR'94*. Springer, 142–151.

[216] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[217] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488* (2021).

[218] Tefko Saracevic. 1970. *On the concept of relevance in information science*. Case Western Reserve University.

[219] Tefko Saracevic. 2016. The Notion of Relevance in Information Science: Everybody knows what relevance is. But, what is it really? *Synthesis lectures on information concepts, retrieval, and services* 8, 3 (2016), i–109.

[220] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1715–1725.

[221] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. 2008. Nearest-neighbor methods in learning and vision. *IEEE Trans. Neural Networks* 19, 2 (2008), 377.

[222] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. 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* (Shanghai, China) *(CIKM '14)*. Association for Computing Machinery, New York, NY, USA, 101–110.

[223] Amit Singhal and Fernando Pereira. 1999. Document Expansion for Speech Retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, California, USA) *(SIGIR '99)*. Association for Computing Machinery, New York, NY, USA, 34–41.

[224] Martin Szummer and Emine Yilmaz. 2011. Semi-Supervised Learning to Rank with Preference Regularization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (Glasgow, Scotland, UK) *(CIKM '11)*. Association for Computing Machinery, New York, NY, USA, 269–278.

[225] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136* (2019).

[226] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732* (2020).

[227] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991* (2022).

[228] Robert S Taylor. 1962. The process of asking questions. *American documentation* 13, 4 (1962), 391–396.

[229] Wilson L Taylor. 1953. "Cloze procedure": A new tool for measuring readability. *Journalism quarterly* 30, 4 (1953), 415–433.

[230] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4593–4601.

[231] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

[232] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[233] Paul Thomas, Gabriella Kazai, Ryen White, and Nick Craswell. 2022. The Crowd is Made of People: Observations from Large-Scale Crowd Labelling. In *ACM SIGIR Conference on Human Information Interaction and Retrieval* (Regensburg, Germany) *(CHIIR '22)*. Association for Computing Machinery, New York, NY, USA, 25–35.

[234] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *Proceedings of the 2014 Australasian Document Computing Symposium* (Melbourne, VIC, Australia) *(ADCS '14)*. Association for Computing Machinery, New York, NY, USA, 58–65.

[235] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv:1908.08962* (2019).

[236] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[237] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5797–5808.

[238] Ellen M. Voorhees. 1993. Using WordNet to Disambiguate Word Senses for Text Retrieval. In *Proceedings of the 16th Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval (Pittsburgh, Pennsylvania, USA) *(SIGIR '93)*. Association for Computing Machinery, New York, NY, USA, 171–180.

[239] Ellen M. Voorhees. 1994. Query Expansion Using Lexical-Semantic Relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '94)*. Springer-Verlag, Berlin, Heidelberg, 61–69.

[240] Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004 (NIST Special Publication)*, Ellen M. Voorhees and Lori P. Buckland (Eds.), Vol. 500-261. National Institute of Standards and Technology (NIST).

[241] Ellen M Voorhees and Donna K Harman. 2005. *TREC: Experiment and evaluation in information retrieval*. MIT press.

[242] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 515–524.

[243] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding. In *International Conference on Learning Representations*.

[244] Yile Wang, Leyang Cui, and Yue Zhang. 2019. How Can BERT Help Lexical Semantics Tasks? *arXiv preprint arXiv:1911.02929* (2019).

[245] Xing Wei and W. Bruce Croft. 2006. LDA-Based Document Models for Ad-Hoc Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 178–185.

[246] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430* (2019).

[247] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan

Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45.

[248] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144* (2016).

[249] Zhijing Wu, Jiaxin Mao, Yiqun Liu, Jingtao Zhan, Yukun Zheng, Min Zhang, and Shaoping Ma. 2020. *Leveraging Passage-Level Cumulative Gain for Document Ranking*. Association for Computing Machinery, New York, NY, USA, 2421–2431.

[250] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 55–64.

[251] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[252] Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient Passage Retrieval with Hashing for Open-domain Question Answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Online, 979–986.

[253] Ming Yan, Chenliang Li, Jiangnan Xia, and Wei Wang. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling. In *TREC*.

[254] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 287–296.

[255] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1253–1256.

[256] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1129–1132.

[257] Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple Applications of BERT for Ad Hoc Document Retrieval. *arXiv:1903.10972* (2019).

[258] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Curran Associates Inc., Red Hook, NY, USA.

[259] Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. 2020. *Capreolus: A Toolkit for End-to-End Neural Ad Hoc Retrieval*. Association for Computing Machinery, New York, NY, USA, 861–864.

[260] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 497–506.

[261] Hansi Zeng, Hamed Zamani, and Vishwa Vinay. 2022. Curriculum Learning for Dense Retrieval Distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1979–1983.

[262] ChengXiang Zhai. 2008. Statistical Language Models for Information Retrieval A Critical Review. *Foundations and Trends in Information Retrieval* 2, 3 (mar 2008), 137–213.

[263] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. *Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance*. Association for Computing Machinery, New York, NY, USA, 2487–2496.

[264] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1503–1512.

[265] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 1328–1336.

[266] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. *An Analysis of BERT in Document Ranking*. Association for Computing Machinery, New York, NY, USA, 1941–1944.

[267] Kaitao Zhang, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2020. *Selective Weak Supervision for Neural Information Retrieval*. Association for Computing Machinery, 474–485.

[268] Wangshu Zhang, Junhong Liu, Zujie Wen, Yafang Wang, and Gerard de Melo. 2020. Query Distillation: BERT-based Distillation for Ensemble Ranking. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*. International Committee on Computational Linguistics, Online, 33–43.

[269] Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2021. Comparing Score Aggregation Approaches for Document Retrieval with Pretrained Transformers. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*. Springer-Verlag, Berlin, Heidelberg, 150–163.

[270] Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. 2020. BERT-QE: Contextualized Query Expansion for Document Re-ranking. In *Findings of the Association for Computational Linguistics:*

EMNLP 2020. Association for Computational Linguistics, Online, 4718–4728.

[271] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, and Ji-Rong Wen. 2022. DynamicRetriever: A Pre-training Model-based IR System with Neither Sparse nor Dense Index. *arXiv preprint arXiv:2203.00537* (2022).

[272] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM computing surveys (CSUR)* 38, 2 (2006), 6–es.